EY-0019E-SG-0001

# VAX/VMS System Programmer

## Student Guide

Prepared by Educational Services
of
Digital Equipment Corporation

Printed in U.S.A.

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may not be used or copied except in accordance with the terms of such license.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Digital.

# STUDENT GUIDE

# STUDENT GUIDE

## COURSE DESCRIPTION

This course is designed for specialists who will be doing consulting work on VAX/VMS at an advanced level. It provides students with considerable laboratory time to practice writing system-level code that will interface to the operating system.

This course is a 'how to' course. To illustrate system-level code, and to provide an opportunity to practice writing such code, the following topics will be discussed:

- General considerations for writing system-level code
- Adding a system service
- Writing a command language interpreter (CLI)
- Writing a symbiont (spooler)
- Writing an application migration executive (AME)

## PREREQUISITES

Fluency in the VAX-11 MACRO language, and successful completion of the VAX/VMS Internals/Data Structures course.

## COURSE GOALS

- Identify restrictions imposed on system-level code, and which VMS features have access mode characteristics.

- Explain the organization of VMS source code, how to call commonly used system routines from a program, and how to work generally with the VMS source kit.

- Given a task involving writing privileged code, select the appropriate technique(s) to use to solve the problem from the following list, and implement the solution:

  - Procedure called in kernel mode ($CMKRNL)
  - Adding a system service (privileged shareable image)
  - Placing code in system buffer shared by all users
  - Using special kernel ASTs to access process context

- Given a task involving changing a user's interface to the operating system, determine when it is appropriate to write the following types of privileged code, and implement the solution:

  - Symbiont (and other communication with the job controller)
  - Command language interpreter (CLI)
  - Compatibility mode operating system emulator (AME)

## NON-GOALS

- Writing device drivers
- Writing ancillary control processors (ACPs)

# STUDENT GUIDE

## RESOURCES

- VAX-11 PATCH Utility Reference Manual

In addition, the following material should be available for your reference:

- VAX/VMS Documentation Set
- VAX/VMS Internals and Data Structures
- VAX/VMS Hardware/Handbook
- VAX/VMS Architecture Handbook
- VAX/VMS Microfiche and Projector
- (Optional) VAX/VMS Source Kit
- VAX-11 Programming Card

# STUDENT GUIDE

## COURSE ORGANIZATION

This course is presented in a lecture/lab format. The instructor will reference the materials in this course handout. Lectures may consist of instructor presentation, class discussions, or directed individual study. The lab time will be used for demonstrations by the instructor, hands-on experience for the students, and the working of exercises and tests.

The course material is structured within modules. Each module is a unique lesson on one or more of the skills required to write a particular type of system program. In many cases, existing system programs are studied as a basis for writing new or altered versions of those programs.

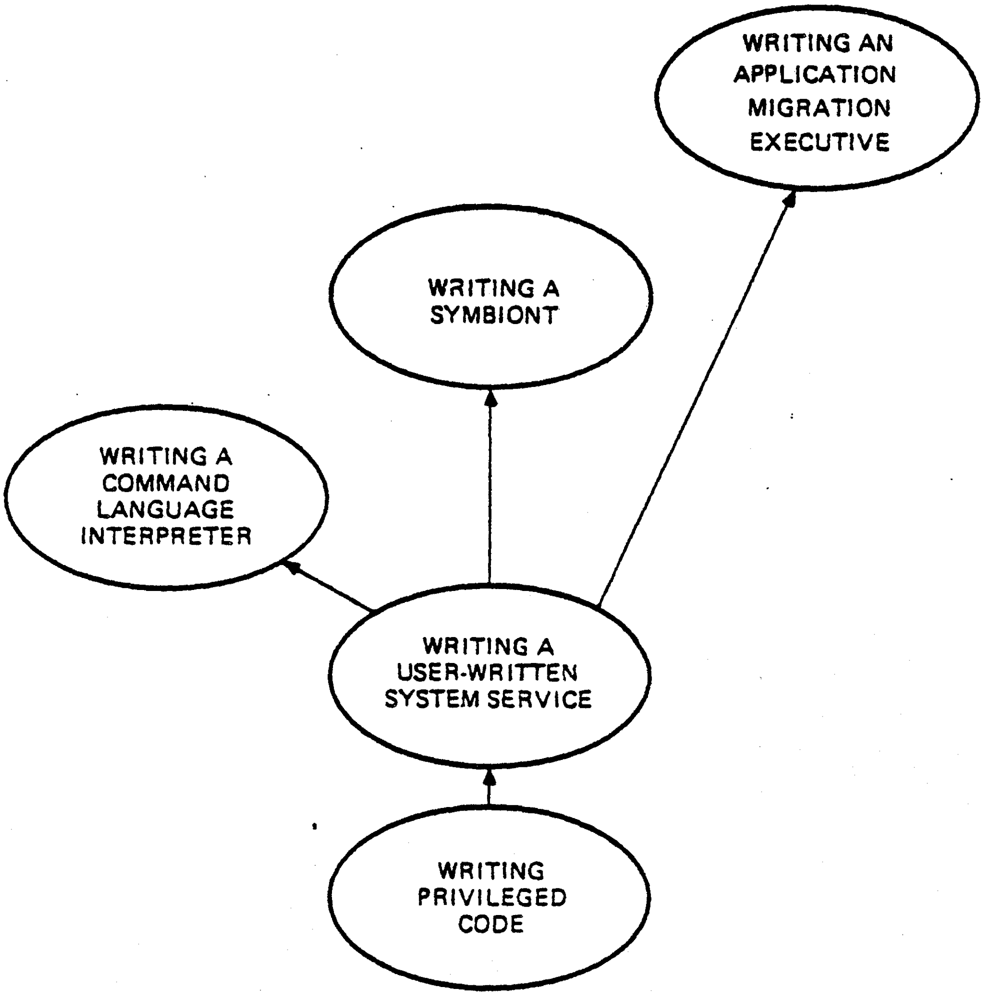
A module consists of:

- An introduction describing the purpose of the lesson.
- At least one objective which states what you will know or be able to do when you complete the module.
- Additional resources that provide supplementary reading and/or reference material for the module.
- The module text consists of examples, reference notes, and copies of any visuals used by the instructor. In terminal printouts, user input is underlined.
- A module test, which may be paper-and-pencil, lab-oriented, or both. By comparing your responses with the answers supplied, you can determine whether or not you have met the objective(s) of the module. If you cannot pass the test, you should consult with your instructor for additional help.

## STUDENT GUIDE

## COURSE MAP

The course map shows the relationship among the various modules. Those modules having arrows leading into other modules are defined as prerequisites for that module. You should complete all the prerequisites for a module before you begin studying its material.


WRITING AN APPLICATION MIGRATION EXECUTIVE
WRITING A SYMBIONT
WRITING A COMMAND LANGUAGE INTERPRETER
WRITING A USER-WRITTEN SYSTEM SERVICE
WRITING PRIVILEGED CODE

TK-9214

# WRITING PRIVILEGED CODE

## INTRODUCTION

There are two types of software programmers, application programmers and system programmers. They produce two types of programs, application programs and system programs. System programs are intended to help programmers write application programs that solve user problems. This course focuses on writing system programs.

When writing system programs, there are two general approaches that can be taken. The programmer can either write a program that solves a particular problem without altering the operating system, or the programmer can try to interface with (or modify) existing operating system components. In either case, the system programmer has to write privileged code.

When trying to solve individual problems, the system programmer may often:

- Write a procedure that will be called in kernel mode (using $CMKRNL)
- Add a system service (privileged shareable image)
- Place code in a system buffer that can be accessed (shared) by many users
- Use special kernel ASTs to access process context

Examples of operating system components that can be modified or replaced include a symbiont, command language interpreter, and compatibility mode operating system emulator (or application migration executive, AME).

One of the most important issues in writing privileged code is synchronization. System programs must observe various conventions to synchronize access to data, and to prevent events from occurring at inopportune times. It is important to synchronize access to data so that shared data structures are protected from being modified simultaneously by several routines. It is equally important that some sequences of instructions be allowed to execute without interruption.

# WRITING PRIVILEGED CODE

Special interrupt priority values (IPLs) can be used to block interrupts from being serviced during the execution of critical code paths. Care must be taken, however, to minimize the use of these special values so that the reporting and handling of important system events can still take place.

This module will focus on synchronization issues for system programs. It will also discuss the following topics related to writing privileged code:

- Restrictions placed on privileged code
- Techniques used for writing privileged code
- VMS features having access mode characteristics
- Commonly used system routines
- Commonly used system macros
- Commonly referenced system locations
- Development tools available for writing system programs

The other modules in this course will build on the concepts presented here. They will be used to demonstrate applications of the rules and principles presented in various areas related to system programming.

## OBJECTIVES

- Identify restrictions imposed on system-level code
- Identify VMS features that have the characteristic of access mode
- Explain the organization of the VMS source code and source kit
- Explain how to reference commonly used system routines and system macros

# WRITING PRIVILEGED CODE

## RESOURCES

- VAX/VMS Internals and Data Structures
- VAX Source Code Listings
- VAX/VMS Guide to Writing Device Drivers (for DELTA)
- VAX/VMS System Dump Analyzer Reference Manual (for SDA)
- VAX-11 PATCH Utility Reference Manual

## Scope of System Programming

- Operating system exists to serve application programs and users
- Application programming utilizes the existing operating system
- System programming modifies or extends the existing operating system to better serve the application programs and users

APPLICATION SYSTEM

<------------------------------------------------->

| Utilize the existing operating system | Alter the existing operating system | Write a new operating system |
|---|---|---|
| - Application programming in VAX/VMS | - Place code in buffer in non-paged pool | - VAX/xyz |
| - Tune the existing operating system | - User written system service | |
| | - Add command language interpreter | |
| | - Add symbiont | |
| | - Add application migration executive | |

# WRITING PRIVILEGED CODE

## System Programming

## vs.

## Application Programming

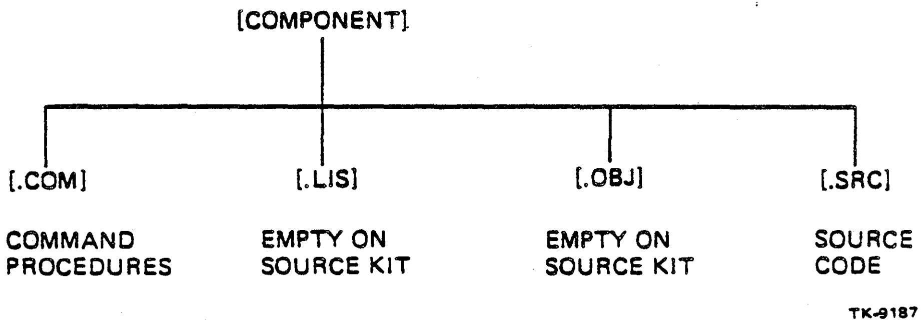
- System programs are intended to help programmers write application programs
- System programs possess a degree of generality not found in typical application programs
- System programs are very concerned with system and user data security, as well as recovery from all possible error conditions
- Application programs are first and foremost problem solvers
- For application programs, system programs are a means to an end

## Approaches to System Programming

- Individual problem solving
  - With procedure called in kernel mode ($CMKRNL)
  - Adding a system service (privileged shareable image)
  - Placing code in system buffer shared by all users
  - Using special kernel ASTs to access process context
- Changing operating system component:
  - Writing a symbiont (or other communication with Job Controller)
  - Writing a command language interpreter
  - Writing a compatibility mode operating system emulator

## Organization of VMS Source Code

- System map
  - SYS$SYSTEM:SYS.MAP
  - Roadmap to source code
- Microfiche
  - Source code listings
  - Use index card to locate source code modules
- Source kit
  - Source code files
  - Requires separate license
  - Organized by system component



## System Programming Tools

- Program Development Tools
  - Powerful instruction set
  - System routines
  - System macros
  - System symbols
- Debugging Tools
  - System Dump Analyzer (SDA)
  - DELTA debugger
  - PATCH utility
  - Console command language
  - MONITOR
- Programming Aids
  - System map
  - System symbol table
  - System macro library
  - System examples directory

## Frequently Used Instructions

### Queue Instructions

| | |
|---|---|
| INSQUE | Insert entry in queue |
| REMQUE | Remove entry from queue |

### Procedure Call and Return Instructions

| | |
|---|---|
| CHMx | Change mode |
| REI | Return from exception or interrupt |
| CALLx | Call procedure |
| RET | Return from procedure |

### Subroutine Call and Return Instructions

| | |
|---|---|
| JSB | Jump to subroutine |
| BSB | Branch to subroutine |
| RSB | Return from subroutine |

### Privileged Processor Register Control Instructions

| | |
|---|---|
| SVPCTX | Save process context |
| LDPCTX | Load process context |
| MTPR | Move to processor register |
| MFPR | Move from processor register |

### Address Manip. Instructions

| | |
|---|---|
| MOVAx | Move address |
| PUSHAx | Push address on stack |

### General Register Manipulation Instructions

| | |
|---|---|
| PUSHL | Push longword |
| PUSHR | Push register(s) |
| POPR | Pop register(s) |
| MOVPSL | Move from PSL |
| BISPSW | Set bit(s) in PSW |
| BICPSW | Clear bit(s) in PSW |

### Unconditional Branch and Jump Instructions

| | |
|---|---|
| BRx | Branch |
| JMP | Jump |

### Loop and Case Instructions

| | |
|---|---|
| ACBx | Add, compare, and branch |
| AOBLEQ | Add one and branch if LEQ |
| AOBLSS | Add one and branch if LT |
| SOBGEQ | Subtract one and branch if GEQ |
| SOBGTR | Subtract one and branch if GT |
| CASE | Case using operand |

## Commonly Used System Macros

- Defined in SYS$LIBRARY:LIB.MLB
  - Can list/extract macros using LIBRARY command
  - Must assemble programs with this library

```
$ MACRO program+SYS$LIBRARY:LIB/LIBRARY
```

- IPL control macros:

```
-  SETIPL   [IPL]        (default = #31)
-  DSBINT   [IPL, DST]   (default = #31, stack)
-  ENBINT   [SRC]        (default = stack)
-  SOFTINT IPL
```

- Address probing macros:

  Arguments are SIZE, ADDRESS, DESTINATION, MODE
  - IFRD
  - IFNORD
  - IFWRT
  - IFNOWRT

- Privilege checking macros:

  Arguments are PRIV, DEST, PCBREG
  - IFPRIV
  - IFNPRIV

# WRITING PRIVILEGED CODE

- Others:
  - ASSUME (Tests assumptions at assembly time)
  - BUG_CHECK (Halts system)
  - RPTEVT (Report system event)
  - FORK (Create fork process)

# WRITING PRIVILEGED CODE

## Commonly Used Definition Macros

- Data Structure Formats ($xyzDEF)
  - $PCBDEF
  - $JIBDEF
  - $IRPDEF
  - $TQEDEF
  - $PHDDEF
- Constants
  - $IPLDEF
  - $SSDEF
  - $PRDEF
  - $IODEF
  - $DYNDEF
- Symbol Definitions
  - $DEFINI
  - $DEF
  - $DEFEND
  - $VIELD

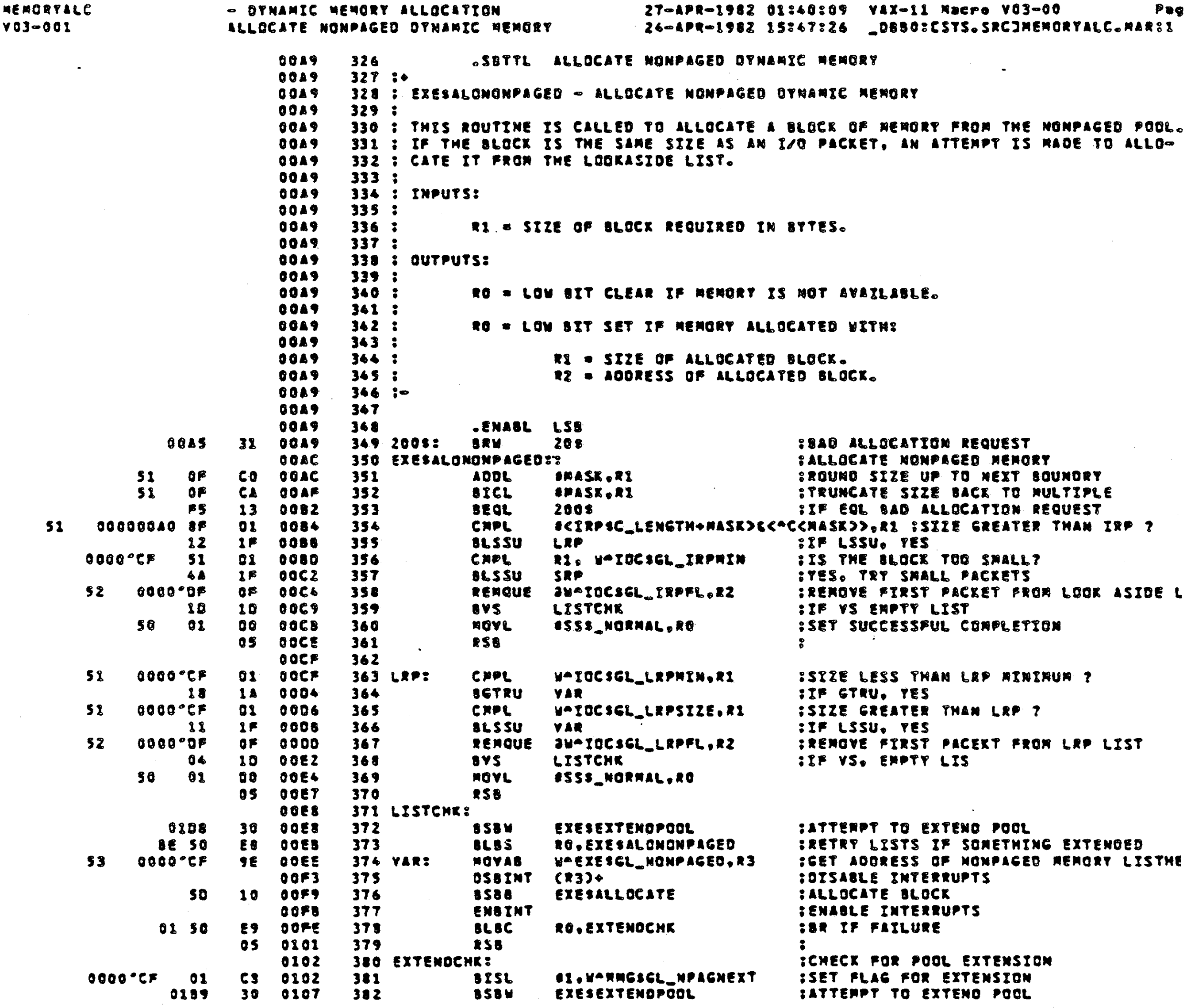
## Commonly Used System Routines

- To access from a program:
  - JSB G^routine
  - Must link program with system symbol table

    ```
    $ LINK program,SYS$SYSTEM:SYS.STB/SELECTIVE
    ```
  - Must relink program with each major release of VMS
- To find inputs/outputs/side effects
  - Look in system map for defining module
  - Find module on fiche, read comments, check code
  - Examine modules that call routine
- Obtaining/Releasing pool space
  - EXE$ALONONPAGED
  - EXE$ALLOCxyz
  - EXE$DEANONPAGED
- Obtaining/Releasing mutex
  - SCH$LOCKR
  - SCH$LOCKW
  - SCH$UNLOCK
- Handling event flags
  - SCH$CLREF
  - SCH$POSTEF

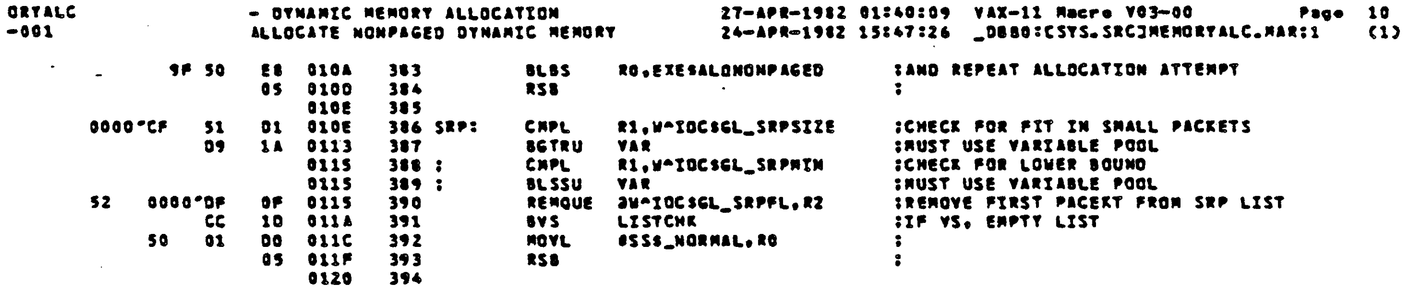
- Others
  - SCH$QAST (queue AST to process)
  - EXE$SNGLEQUOTA (check single quota)
  - EXE$BUFFRQUOTA (check buffered byte count quota)
  - EXE$NAMPID (convert process name to PID)

Listing 1-1: Sample System Routine (EXE$ALONONPAGED) (Page 1 of 2)



```
MEMORYALC                - DYNAMIC MEMORY ALLOCATION          27-APR-1982 01:40:09  VAX-11 Macro V03-00          Pag
V03-001                  ALLOCATE NONPAGED DYNAMIC MEMORY     26-APR-1982 15:47:26  _DBB0:[SYS.SRC]MEMORYALC.MAR;1

                                 00A9   326         .SBTTL  ALLOCATE NONPAGED DYNAMIC MEMORY
                                 00A9   327 ;+
                                 00A9   328 ; EXE$ALONONPAGED - ALLOCATE NONPAGED DYNAMIC MEMORY
                                 00A9   329 ;
                                 00A9   330 ; THIS ROUTINE IS CALLED TO ALLOCATE A BLOCK OF MEMORY FROM THE NONPAGED POOL.
                                 00A9   331 ; IF THE BLOCK IS THE SAME SIZE AS AN I/O PACKET, AN ATTEMPT IS MADE TO ALLO-
                                 00A9   332 ; CATE IT FROM THE LOOKASIDE LIST.
                                 00A9   333 ;
                                 00A9   334 ; INPUTS:
                                 00A9   335 ;
                                 00A9   336 ;       R1 = SIZE OF BLOCK REQUIRED IN BYTES.
                                 00A9   337 ;
                                 00A9   338 ; OUTPUTS:
                                 00A9   339 ;
                                 00A9   340 ;       R0 = LOW BIT CLEAR IF MEMORY IS NOT AVAILABLE.
                                 00A9   341 ;
                                 00A9   342 ;       R0 = LOW BIT SET IF MEMORY ALLOCATED WITH:
                                 00A9   343 ;
                                 00A9   344 ;               R1 = SIZE OF ALLOCATED BLOCK.
                                 00A9   345 ;               R2 = ADDRESS OF ALLOCATED BLOCK.
                                 00A9   346 ;-
                                 00A9   347
                                 00A9   348         .ENABL  LSB
                  00A5     31    00A9   349 200$:   BRW     20$                         ;BAD ALLOCATION REQUEST
                                 00AC   350 EXE$ALONONPAGED::                           ;ALLOCATE NONPAGED MEMORY
            51    0F       C0    00AC   351         ADDL    #MASK,R1                    ;ROUND SIZE UP TO NEXT BOUNDRY
            51    0F       CA    00AF   352         BICL    #MASK,R1                    ;TRUNCATE SIZE BACK TO MULTIPLE
                  F5       13    00B2   353         BEQL    200$                        ;IF EQL BAD ALLOCATION REQUEST
   51   000000A0  8F       D1    00B4   354         CMPL    #<IRP$C_LENGTH+MASK>&<^C<MASK>>,R1 ;SIZE GREATER THAN IRP ?
                  12       1F    00B8   355         BLSSU   LRP                         ;IF LSSU, YES
      0000'CF     51       D1    00BD   356         CMPL    R1, W^IOC$GL_IRPMIN         ;IS THE BLOCK TOO SMALL?
                  4A       1F    00C2   357         BLSSU   SRP                         ;YES, TRY SMALL PACKETS
     52   0000'DF          0F    00C4   358         REMQUE  @W^IOC$GL_IRPFL,R2          ;REMOVE FIRST PACKET FROM LOOK ASIDE L
                  1D       1D    00C9   359         BVS     LISTCHK                     ;IF VS EMPTY LIST
            50    01       D0    00CB   360         MOVL    #SS$_NORMAL,R0              ;SET SUCCESSFUL COMPLETION
                           05    00CE   361         RSB                                 ;
                                 00CF   362
     51   0000'CF          D1    00CF   363 LRP:    CMPL    W^IOC$GL_LRPMIN,R1          ;SIZE LESS THAN LRP MINIMUM ?
                  18       1A    00D4   364         BGTRU   VAR                         ;IF GTRU, YES
     51   0000'CF          D1    00D6   365         CMPL    W^IOC$GL_LRPSIZE,R1         ;SIZE GREATER THAN LRP ?
                  11       1F    00DB   366         BLSSU   VAR                         ;IF LSSU, YES
     52   0000'DF          0F    00DD   367         REMQUE  @W^IOC$GL_LRPFL,R2          ;REMOVE FIRST PACEKT FROM LRP LIST
                  04       1D    00E2   368         BVS     LISTCHK                     ;IF VS, EMPTY LIS
            50    01       D0    00E4   369         MOVL    #SS$_NORMAL,R0
                           05    00E7   370         RSB
                                 00E8   371 LISTCHK:
                  01D8     30    00E8   372         BSBW    EXE$EXTENDPOOL              ;ATTEMPT TO EXTEND POOL
                BE 50      E8    00EB   373         BLBS    R0,EXE$ALONONPAGED          ;RETRY LISTS IF SOMETHING EXTENDED
     53   0000'CF          9E    00EE   374 VAR:    MOVAB   W^EXE$GL_NONPAGED,R3        ;GET ADDRESS OF NONPAGED MEMORY LISTHE
                                 00F3   375         DSBINT  (R3)+                       ;DISABLE INTERRUPTS
                  5D       10    00F9   376         BSBB    EXE$ALLOCATE                ;ALLOCATE BLOCK
                                 00FB   377         ENBINT                              ;ENABLE INTERRUPTS
                01 50      E9    00FE   378         BLBC    R0,EXTENDCHK                ;BR IF FAILURE
                           05    0101   379         RSB                                 ;
                                 0102   380 EXTENDCHK:                                  ;CHECK FOR POOL EXTENSION
      0000'CF     01       C8    0102   381         BISL    #1,W^MMG$GL_NPAGNEXT        ;SET FLAG FOR EXTENSION
                  01B9     30    0107   382         BSBW    EXE$EXTENDPOOL              ;ATTEMPT TO EXTEND POOL
```

Listing 1-1: Sample System Routine (EXE$ALONONPAGED) (Page 2 of 2)



```
ORYALC                  - DYNAMIC MEMORY ALLOCATION             27-APR-1982 01:40:09  VAX-11 Macro V03-00         Page  10
-001                    ALLOCATE NONPAGED DYNAMIC MEMORY        24-APR-1982 15:47:26  _DBB0:[SYS.SRC]MEMORYALC.MAR;1   (1)

     -      9F 50  E8  010A   383          BLBS    R0,EXE$ALONONPAGED      ;AND REPEAT ALLOCATION ATTEMPT
                   05  010D   384          RSB                             ;
                       010E   385
    0000'CF   51   D1  010E   386 SRP:     CMPL    R1,W^IOC$GL_SRPSIZE     ;CHECK FOR FIT IN SMALL PACKETS
              09   1A  0113   387          BGTRU   VAR                     ;MUST USE VARIABLE POOL
                       0115   388 ;        CMPL    R1,W^IOC$GL_SRPMIN      ;CHECK FOR LOWER BOUND
                       0115   389 ;        BLSSU   VAR                     ;MUST USE VARIABLE POOL
    52   0000'DF   0F  0115   390          REMQUE  @W^IOC$GL_SRPFL,R2      ;REMOVE FIRST PACEKT FROM SRP LIST
              CC   1D  011A   391          BVS     LISTCHK                 ;IF VS, EMPTY LIST
         50   01   D0  011C   392          MOVL    #SS$_NORMAL,R0          ;
                   05  011F   393          RSB                             ;
                       0120   394
```

## Commonly Referenced System Symbols

- Many symbols defined in modules SYSCOMMON and SYSPARAM
- If reference in program, need to link with system symbol table
- Precede symbol names in program with G^

Symbols used by most components in executive (EXE$...)

| | |
|---|---|
| EXE$GL_SITESPEC | Can be used for any purpose you choose |
| EXE$GL_ABSTIM | System absolute time, in seconds |
| EXE$GQ_SYSTIME | System absolute time, in nanoseconds |
| EXE$GL_NONPAGED+4 | Address of first free block of nonpaged pool |
| EXE$GL_PAGED | Address of first free block of paged pool |
| EXE$GL_SCB | Virtual address of system control block |
| EXE$GL_TQFL | Forward link to timer queue elements |

Control region (P1 space) locations (CTL$...)

| | |
|---|---|
| CTL$GL_PHD | Base of window to process header |

I/O data base symbols (IOC$...)

| | |
|---|---|
| IOC$GL_IRPBL | Backward link to last used I/O request packet |
| IOC$GL_MUTEX | I/O data base mutex |

Memory management symbols (MMG$...)

| | |
|---|---|
| MMG$GL_SBR | System page table base register |
| MMG$GL_SPTBASE | Base address of system page table (virtual) |

Scheduler data base symbols (SCH$...)

| | |
|---|---|
| SCH$GB_PRI | Software priority of CURRENT process |
| SCH$GL_CURPCB | CURRENT process PCB address |
| SCH$GL_PCBVEC | Base of address of PCB vector table |

## Commonly Used System Programming Techniques

- Calling procedures in kernel mode
  - Requires process context
  - Could be used to implement system service

- Place code in buffer from non-paged pool
  - Process context not required
  - Code could be invoked by TQE
  - Accessible to all processes at same virtual address
  - Code must be position-independent (PIC)

- Build AST control block and queue to another process
  - To gain access to another process's context
  - AST code often part of AST control block
  - Frequently queues AST back to original process when done
  - For example, $GETJPI

- Write program that gets mapped into a process's address space
  - Alternatives to VMS supplied components (for example):
    - Command language interpreter (P1 space)
    - Application migration executive (PØ space)

- Write program that runs as separate process
  - For example, symbionts

## Considerations in Writing System Programs

- Addressing region implications
  - Process or system wide access
  - Speed (SØ address translation faster)
  - Paged or non-paged pool (if in system space)
- Calling sequence
  - JSB/RSB faster than CALL/RET
  - No need to worry about stack with CALL/RET
  - Need to explicitly save/restore registers with JSB/RSB
- VMS features having access mode implications
  - Logical names
  - I/O channels
  - Mailboxes
  - Pages of memory
    - Data structures
    - Virtual address space
  - Condition handling routines
  - Exit handling routines
  - ASTs
  - Timer requests

# WRITING PRIVILEGED CODE

## DO's in System Programming

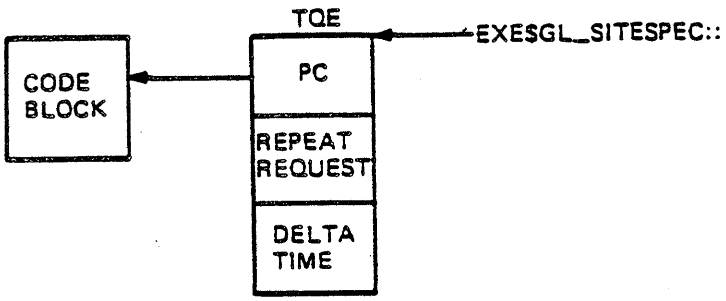
- Run in privileged access mode (typically kernel)
- Write code that is re-entrant
- Write code that is position independent
- Check appropriate assumptions about user
  - Privileges
  - Quotas
  - Access to buffers
- Ensure process not deleted at inopportune times (elevate IPL to 2)
  - While have mutex
  - While have buffer from non-paged pool
- Check for and respond to all possible error conditions
  - Unlink data structures from queues
  - Deallocate buffers previously allocated
  - Restore modified fields in other data structures
- Always be concerned about synchronization problems
  - When accessing data structure
  - When altering IPL
  - When testing specific locations for values
- Always RAISE IPL to synchronize access to data structures or to report system events
- Generally exit a routine at the same IPL at which it was entered

## DONT's in System Programming

- Avoid exceptions in executive and kernel mode
  - Last chance handler for kernel mode issues fatal BUG_CHECK
  - Unhandled executive mode exceptions result in process deletion
- Only call system services at IPL=0
  - Most system services raise and lower IPL (back to 0)
    - REI to higher IPL will cause reserved operand exception
- Use $service macros, not CALLx G^EXE$service
  - Services that place a process in wait state make assumptions about state of stack (and update return PC)
- Cannot take page faults above IPL 2
  - Take care to lock code in working set
- Don't overuse limited system resources
  - Memory
- Don't assume anything about current process
  - P0, P1 address space
  - Access PHD through P1 pointer
- Don't do anything in a privileged mode that can be done in a less privileged mode
- Don't stay at an elevated IPL longer than absolutely necessary for synchronization

## Sample System Programs

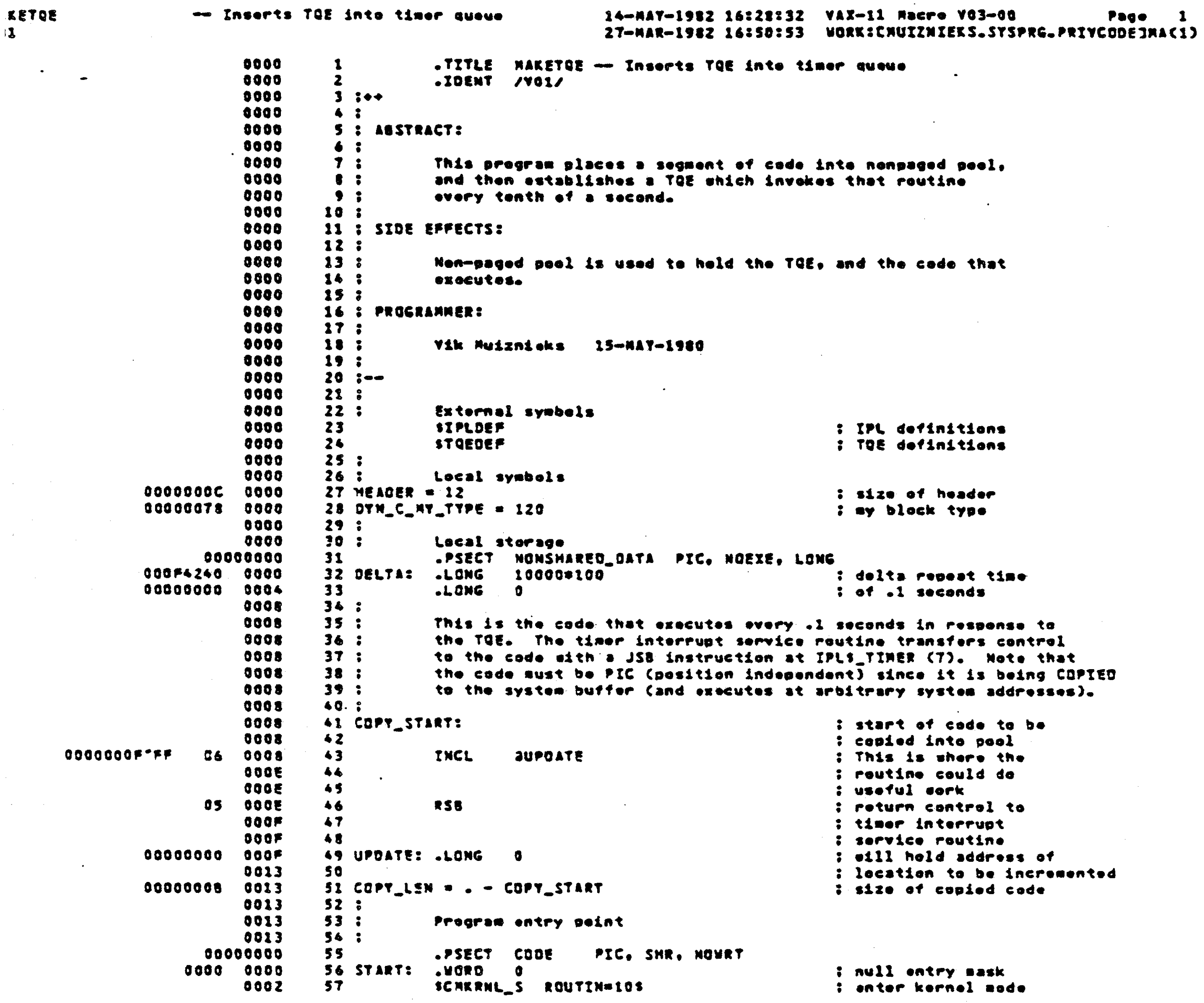
- MAKETQE
  - Allocates two blocks from non-paged pool
  - Places code to execute periodically in first block
  - Makes second block TQE that invokes code in first block
  - Records address of TQE block in site-specific longword
  - After program run, user can log out
    - Code will still be executed periodically
    - No process overhead involved
    - Independent of CURRENT process



TK-9188

- STOPTQE
  - Removes TQE from queue
  - Deallocates TQE and code block
  - Clears site-specific longword

## Listing 1-2: MAKETQE Example (Page 1 of 3)

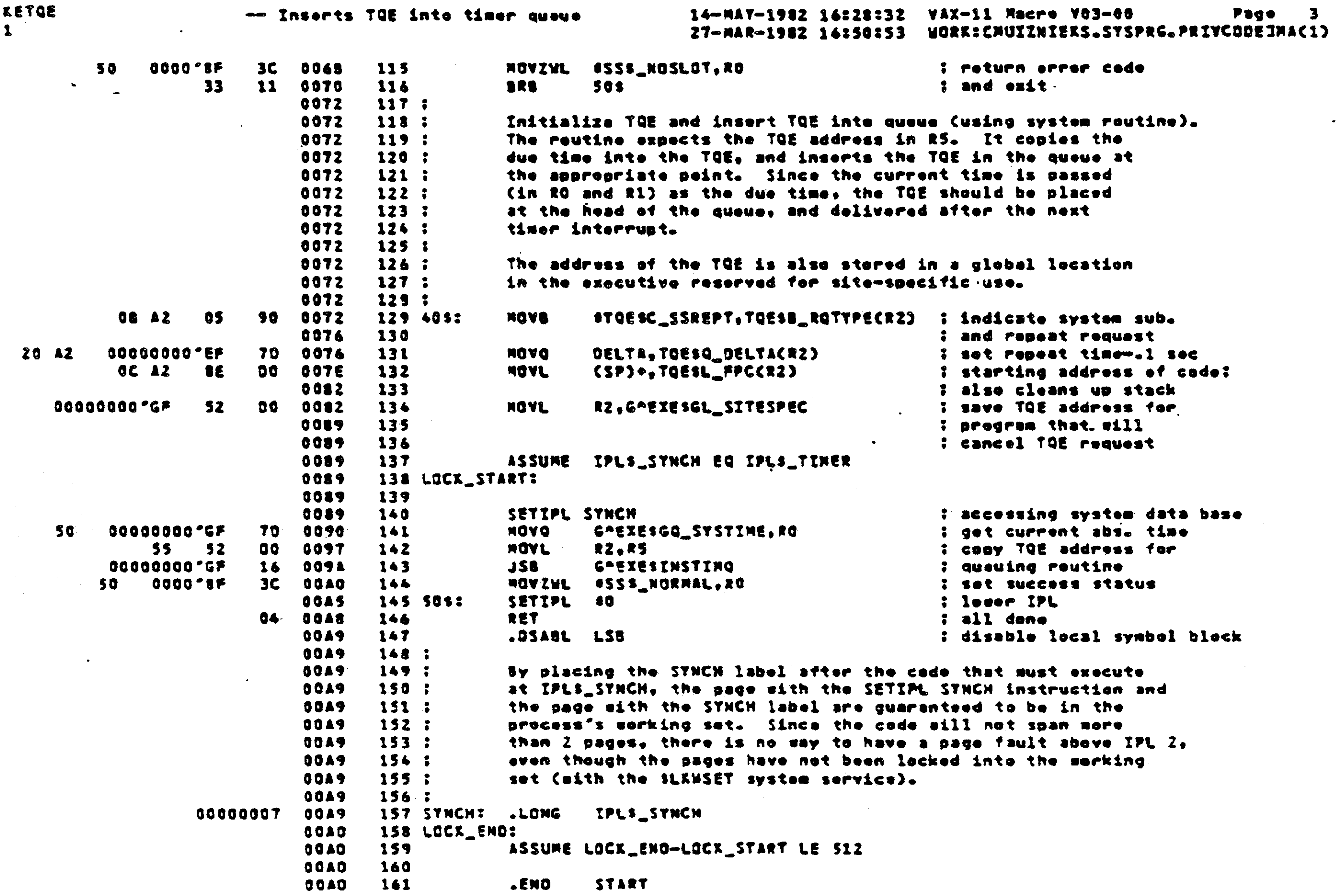


```
KETQE              -- Inserts TQE into timer queue        14-MAY-1982 16:28:32  VAX-11 Macro V03-00         Page   1
1                                                         27-MAR-1982 16:50:53  WORK:[MUIZNIEKS.SYSPRG.PRIVCODE]MA(1)

                     0000          .TITLE  MAKETQE -- Inserts TQE into timer queue
                     0000          .IDENT  /V01/
                     0000  ;++
                     0000  ;
                     0000  ; ABSTRACT:
                     0000  ;
                     0000  ;       This program places a segment of code into nonpaged pool,
                     0000  ;       and then establishes a TQE which invokes that routine
                     0000  ;       every tenth of a second.
                     0000  ;
                     0000  ; SIDE EFFECTS:
                     0000  ;
                     0000  ;       Non-paged pool is used to hold the TQE, and the code that
                     0000  ;       executes.
                     0000  ;
                     0000  ; PROGRAMMER:
                     0000  ;
                     0000  ;       Vik Muiznieks   15-MAY-1980
                     0000  ;
                     0000  ;--
                     0000  ;
                     0000  ;       External symbols
                     0000          $IPLDEF                                    ; IPL definitions
                     0000          $TQEDEF                                    ; TQE definitions
                     0000  ;
                     0000  ;       Local symbols
           0000000C  0000  HEADER = 12                                        ; size of header
           00000078  0000  DYN_C_MY_TYPE = 120                                ; my block type
                     0000  ;
                     0000  ;       Local storage
                 00000000          .PSECT  NONSHARED_DATA  PIC, NOEXE, LONG
           000F4240  0000  DELTA:  .LONG   10000*100                          ; delta repeat time
           00000000  0004          .LONG   0                                  ; of .1 seconds
                     0008  ;
                     0008  ;       This is the code that executes every .1 seconds in response to
                     0008  ;       the TQE.  The timer interrupt service routine transfers control
                     0008  ;       to the code with a JSB instruction at IPL$_TIMER (7).  Note that
                     0008  ;       the code must be PIC (position independent) since it is being COPIED
                     0008  ;       to the system buffer (and executes at arbitrary system addresses).
                     0008  ;
                     0008  COPY_START:                                        ; start of code to be
                     0008                                                     ; copied into pool
     0000000F'FF  D6  0008          INCL    @UPDATE                           ; This is where the
                     000E                                                     ; routine could do
                     000E                                                     ; useful work
                 05  000E          RSB                                        ; return control to
                     000F                                                     ; timer interrupt
                     000F                                                     ; service routine
           00000000  000F  UPDATE: .LONG   0                                  ; will hold address of
                     0013                                                     ; location to be incremented
           00000008  0013  COPY_LEN = . - COPY_START                          ; size of copied code
                     0013  ;
                     0013  ;       Program entry point
                     0013  ;
                 00000000          .PSECT  CODE    PIC, SHR, NOWRT
               0000  0000  START:  .WORD   0                                  ; null entry mask
                     0002          $CMKRNL_S  ROUTIN=10$                      ; enter kernel mode
```

## Listing 1-2: MAKETQE Example (Page 2 of 3)

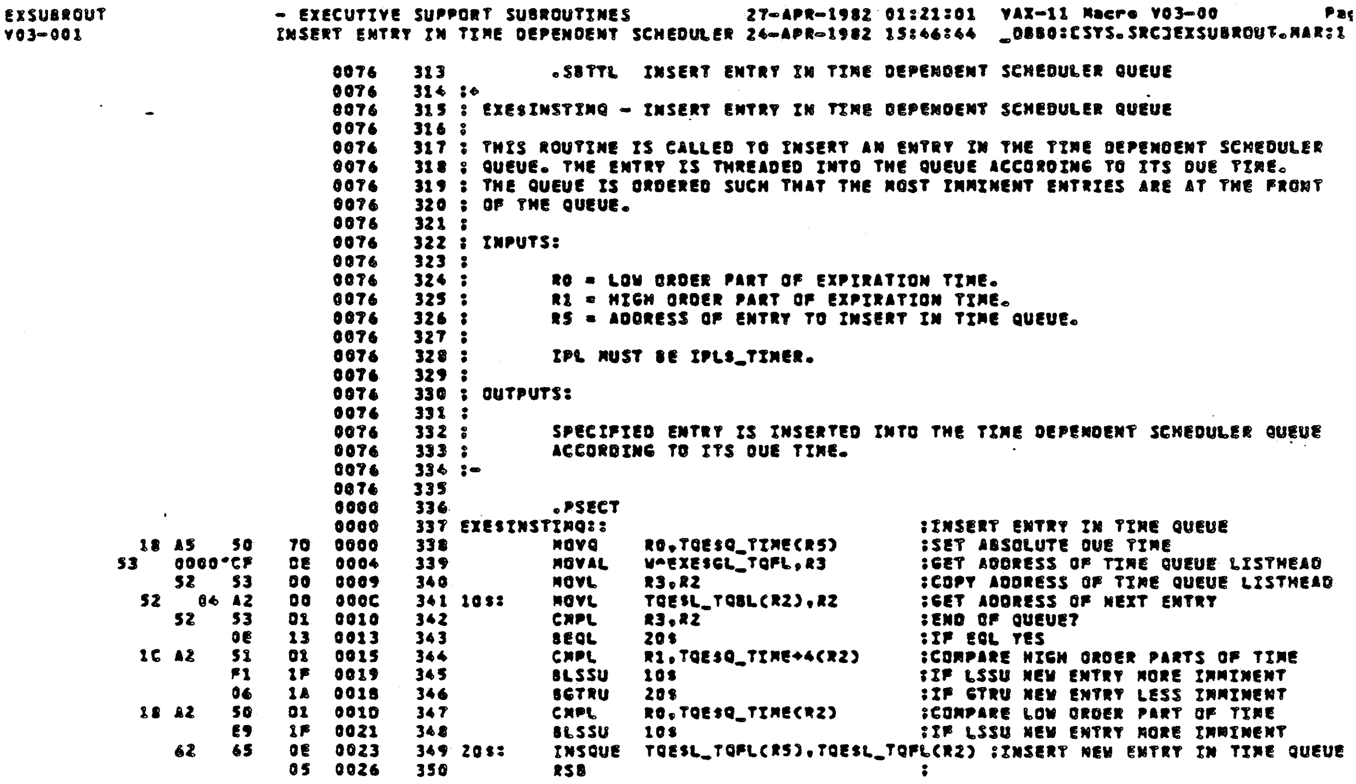
```
MAKETQE                      -- Inserts TQE into timer queue          16-MAY-1982 16:28:32  VAX-11 Macro V03-00
V01                                                                   27-MAR-1982 16:50:53  WORK:CMUIZNIEKS.SYSPRG.PRIVC

                      04  0011          RET                                   ; all done
                          0012
                    003C  0012     10$: .WORD   ^M<R2,R3,R4,R5>               ; save registers used
                          0014          .ENABL  LSB                           ; enable local symbol block
       00000000'GF    D5  0014          TSTL    G^EXE$GL_SITESPEC             ; if in use, error
                08    13  001A          BEQLU   15$
  50   00000000'8F    D0  001C          MOVL    #SS$_IVMODE,R0
                      04  0023          RET
                          0024     ;
                          0024     ;    Allocate pool to hold code.  Code must be placed in system
                          0024     ;    space so that it can execute in ANY process context.  HEADER extra
                          0024     ;    bytes will be allocated for a header (since the code block may
                          0024     ;    later be deleted by running program STOPTQE).  The program will
                          0024     ;    use the first word in the third longword to store the size of
                          0024     ;    the block.  Normally the system uses the first two longwords
                          0024     ;    for forward and backward links.  In this case, the first
                          0024     ;    longword will be incremented each time the routine specified
                          0024     ;    by the TQE executes.  The second longword will not be used.
                          0024     ;    Note that IPL is raised to IPL$_ASTDEL before the block of pool
                          0024     ;    is allocated.  This is done so that the process can not be
                          0024     ;    deleted while it has the address of the block in a register
                          0024     ;    (and no other record of the block is maintained elsewhere in
                          0024     ;    the system).
                          0024     ;
            51    17  00  0024     15$: MOVL    #COPY_LEN+HEADER,R1           ; size of pool needed
                          0027          SETIPL  #IPL$_ASTDEL                  ; so process not deleted
       00000000'GF    16  002A          JSB     G^EXE$ALONONPAGED             ; allocate pool
                          0030     ;
                          0030     ;    The above routine destroys R0-R3, and returns in R2 the
                          0030     ;    address of the allocated block of pool.
                          0030     ;
             09 50    E8  0030          BLBS    R0,20$                        ; proceed if no error
                          0033          SETIPL  #0                            ; lower IPL before exiting
        50   0000'8F  3C  0036          MOVZWL  #SS$_INSFMEM,R0               ; indicate error
                      04  0038          RET                                   ; return error code
  0000000F'EF    52   D0  003C     20$: MOVL    R2,UPDATE                     ; save address of block
                 82   7C  0043          CLRQ    (R2)+                         ; clear location to be upd
                          0045                                                ; point R2 to 3rd longword
            82   51   B0  0045          MOVW    R1,(R2)+                      ; fill in size field
        82    78 8F   9B  0048          MOVZBW  #DYN_C_MY_TYPE,(R2)+          ; fill in type field and
                          004C                                                ; point R2 to start of cod
                 52   DD  004C          PUSHL   R2                            ; save address of code
62   00000008'EF 08   28  004E          MOVC3   #COPY_LEN,COPY_START,(R2)     ; copy code to buffer
                          0056                                                ; NOTE -- R0-R5 altered
                          0056     ;
                          0056     ;    Allocate a TQE.  Note that the routine allocates the TQE at
                          0056     ;    IPL$_SYNCH, but returns control at IPL$_ASTDEL (so process
                          0056     ;    cannot be deleted before it can deallocate pool used for TQE).
                          0056     ;    The routine destroys R0-R4, and returns the address of the TQE
                          0056     ;    block in R2.
                          0056     ;
       00000000'GF    16  0056          JSB     G^EXE$ALLOCTQE                ; allocate TQE block
             13 50    E8  005C          BLBS    R0,40$                        ; continue if no error
            50   8E   D0  005F          MOVL    (SP)+,R0                      ; else, get code address
                          0062                                                ; and clean up stack
            50   0C   C2  0062          SUBL    #HEADER,R0                    ; account for header
       00000000'GF    16  0065          JSB     G^EXE$DEANONPAGED             ; deallocate code block
```

## Listing 1-2: MAKETQE Example (Page 3 of 3)



```
KETQE                  -- Inserts TQE into timer queue        14-MAY-1982 16:28:32  VAX-11 Macro V03-00         Page   3
1                                                             27-MAR-1982 16:50:53  WORK:[MUIZNIEKS.SYSPRG.PRIVCODE]MA(1)

        50   0000'8F    3C  0068    115           MOVZWL  #SS$_NOSLOT,R0                   ; return error code
                  33    11  0070    116           BRB     50$                              ; and exit
                            0072    117 ;
                            0072    118 ;         Initialize TQE and insert TQE into queue (using system routine).
                            0072    119 ;         The routine expects the TQE address in R5.  It copies the
                            0072    120 ;         due time into the TQE, and inserts the TQE in the queue at
                            0072    121 ;         the appropriate point.  Since the current time is passed
                            0072    122 ;         (in R0 and R1) as the due time, the TQE should be placed
                            0072    123 ;         at the head of the queue, and delivered after the next
                            0072    124 ;         timer interrupt.
                            0072    125 ;
                            0072    126 ;         The address of the TQE is also stored in a global location
                            0072    127 ;         in the executive reserved for site-specific use.
                            0072    128 ;
          08 A2    05   90  0072    129 40$:      MOVB    #TQE$C_SSREPT,TQE$B_RQTYPE(R2)  ; indicate system sub.
                            0076    130                                                    ; and repeat request
   20 A2   00000000'EF  7D  0076    131           MOVQ    DELTA,TQE$Q_DELTA(R2)            ; set repeat time=.1 sec
          0C A2    8E   D0  007E    132           MOVL    (SP)+,TQE$L_FPC(R2)              ; starting address of code:
                            0082    133                                                    ; also cleans up stack
     00000000'GF   52   D0  0082    134           MOVL    R2,G^EXE$GL_SITESPEC             ; save TQE address for
                            0089    135                                                    ; program that will
                            0089    136                                                    ; cancel TQE request
                            0089    137           ASSUME  IPL$_SYNCH EQ IPL$_TIMER
                            0089    138 LOCK_START:
                            0089    139
                            0089    140           SETIPL  SYNCH                            ; accessing system data base
      50   00000000'GF  7D  0090    141           MOVQ    G^EXE$GQ_SYSTIME,R0              ; get current abs. time
                  55    52  D0  0097    142       MOVL    R2,R5                            ; copy TQE address for
        00000000'GF     16  009A    143           JSB     G^EXE$INSTIMQ                    ; queuing routine
        50   0000'8F    3C  00A0    144           MOVZWL  #SS$_NORMAL,R0                   ; set success status
                            00A5    145 50$:      SETIPL  #0                               ; lower IPL
                        04  00A8    146           RET                                      ; all done
                            00A9    147           .DSABL  LSB                              ; disable local symbol block
                            00A9    148 ;
                            00A9    149 ;         By placing the SYNCH label after the code that must execute
                            00A9    150 ;         at IPL$_SYNCH, the page with the SETIPL SYNCH instruction and
                            00A9    151 ;         the page with the SYNCH label are guaranteed to be in the
                            00A9    152 ;         process's working set.  Since the code will not span more
                            00A9    153 ;         than 2 pages, there is no way to have a page fault above IPL 2,
                            00A9    154 ;         even though the pages have not been locked into the working
                            00A9    155 ;         set (with the $LKWSET system service).
                            00A9    156 ;
                 00000007   00A9    157 SYNCH:    .LONG   IPL$_SYNCH
                            00AD    158 LOCK_END:
                            00AD    159           ASSUME LOCK_END-LOCK_START LE 512
                            00AD    160
                            00AD    161           .END    START
```

## Listing 1-3: EXE$INSTIMQ (from module EXSUBROUT)

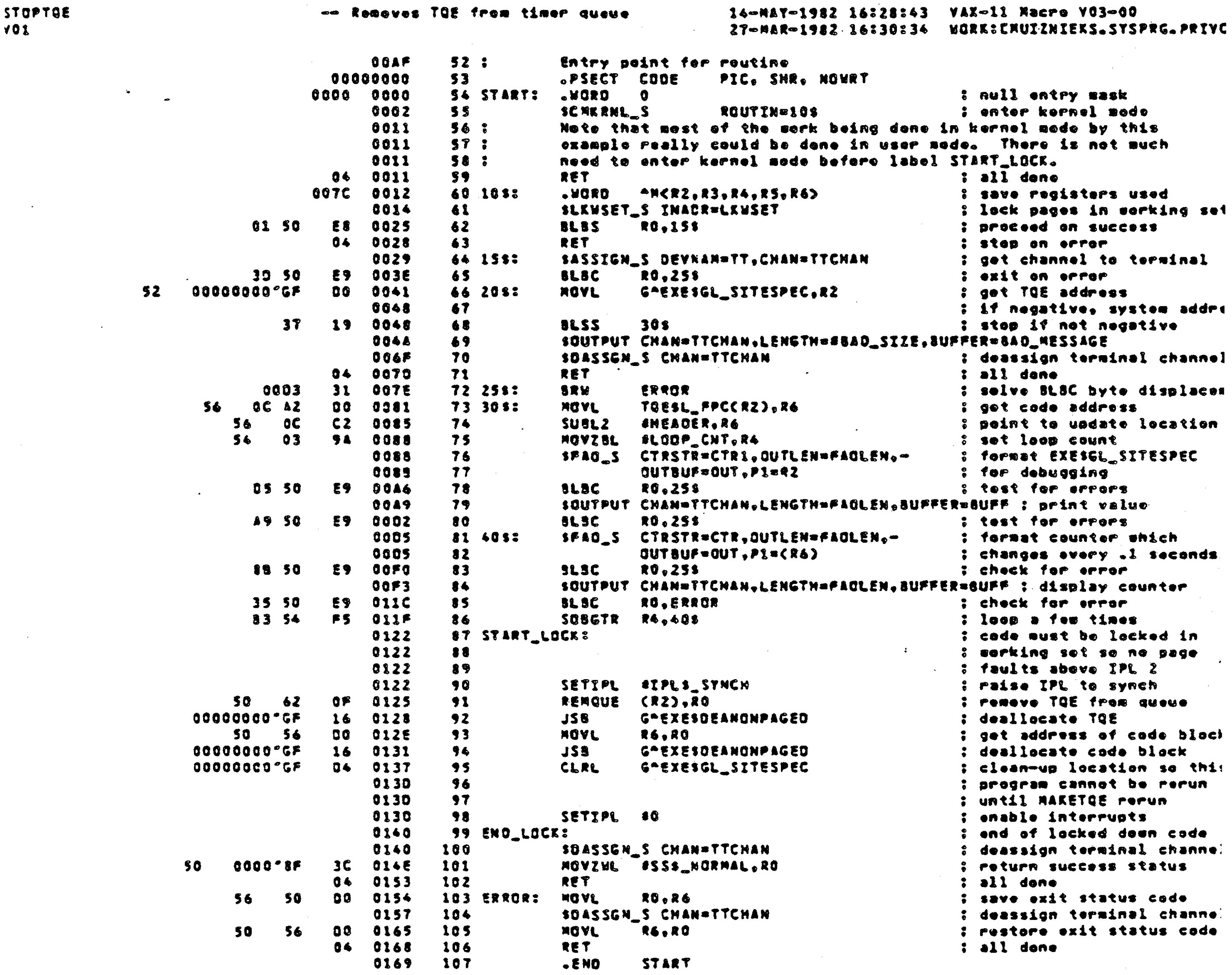


```
EXSUBROUT            - EXECUTIVE SUPPORT SUBROUTINES         27-APR-1982 01:21:01  VAX-11 Macro V03-00         Pag
V03-001              INSERT ENTRY IN TIME DEPENDENT SCHEDULER 24-APR-1982 15:46:44  _DBB0:[SYS.SRC]EXSUBROUT.MAR;1

                             0076   313          .SBTTL  INSERT ENTRY IN TIME DEPENDENT SCHEDULER QUEUE
                             0076   314 ;+
                             0076   315 ; EXE$INSTIMQ - INSERT ENTRY IN TIME DEPENDENT SCHEDULER QUEUE
                             0076   316 ;
                             0076   317 ; THIS ROUTINE IS CALLED TO INSERT AN ENTRY IN THE TIME DEPENDENT SCHEDULER
                             0076   318 ; QUEUE. THE ENTRY IS THREADED INTO THE QUEUE ACCORDING TO ITS DUE TIME.
                             0076   319 ; THE QUEUE IS ORDERED SUCH THAT THE MOST IMMINENT ENTRIES ARE AT THE FRONT
                             0076   320 ; OF THE QUEUE.
                             0076   321 ;
                             0076   322 ; INPUTS:
                             0076   323 ;
                             0076   324 ;        R0 = LOW ORDER PART OF EXPIRATION TIME.
                             0076   325 ;        R1 = HIGH ORDER PART OF EXPIRATION TIME.
                             0076   326 ;        R5 = ADDRESS OF ENTRY TO INSERT IN TIME QUEUE.
                             0076   327 ;
                             0076   328 ;        IPL MUST BE IPL$_TIMER.
                             0076   329 ;
                             0076   330 ; OUTPUTS:
                             0076   331 ;
                             0076   332 ;        SPECIFIED ENTRY IS INSERTED INTO THE TIME DEPENDENT SCHEDULER QUEUE
                             0076   333 ;        ACCORDING TO ITS DUE TIME.
                             0076   334 ;-
                             0076   335
                             0000   336          .PSECT
                             0000   337 EXE$INSTIMQ::                               ;INSERT ENTRY IN TIME QUEUE
  18 A5     50    70         0000   338          MOVQ    R0,TQE$Q_TIME(R5)          ;SET ABSOLUTE DUE TIME
53    0000'CF     DE         0004   339          MOVAL   W^EXE$GL_TQFL,R3           ;GET ADDRESS OF TIME QUEUE LISTHEAD
      52    53    D0         0009   340          MOVL    R3,R2                      ;COPY ADDRESS OF TIME QUEUE LISTHEAD
  52    04 A2     D0         000C   341 10$:     MOVL    TQE$L_TQBL(R2),R2          ;GET ADDRESS OF NEXT ENTRY
      52    53    D1         0010   342          CMPL    R3,R2                      ;END OF QUEUE?
            0E    13         0013   343          BEQL    20$                        ;IF EQL YES
  1C A2     51    D1         0015   344          CMPL    R1,TQE$Q_TIME+4(R2)        ;COMPARE HIGH ORDER PARTS OF TIME
            F1    1F         0019   345          BLSSU   10$                        ;IF LSSU NEW ENTRY MORE IMMINENT
            06    1A         001B   346          BGTRU   20$                        ;IF GTRU NEW ENTRY LESS IMMINENT
  18 A2     50    D1         001D   347          CMPL    R0,TQE$Q_TIME(R2)          ;COMPARE LOW ORDER PART OF TIME
            E9    1F         0021   348          BLSSU   10$                        ;IF LSSU NEW ENTRY MORE IMMINENT
      62    65    0E         0023   349 20$:     INSQUE  TQE$L_TQFL(R5),TQE$L_TQFL(R2) ;INSERT NEW ENTRY IN TIME QUEUE
                  05         0026   350          RSB                                ;
```

## Listing 1-4: STOPTQE Example (Page 1 of 2)

```
                               -- Removes TQE from timer queue        14-MAY-1982 16:28:43  VAX-11 Macro V03-00          Page   1
                                                                      27-MAR-1982 16:30:34  WORK:[MUIZNIEKS.SYSPRG.PRIVCODE]ST(1)

                                    0000     1         .TITLE  STOPTCE -- Removes TQE from timer queue
                                    0000     2         .IDENT  /V01/
                                    0000     3 ;++
                                    0000     4 ;
                                    0000     5 ; ABSTARCT:
                                    0000     6 ;
                                    0000     7 ;       This program displays the contents of the location being updated
                                    0000     8 ;       by the routine specified in a TQE (thrice).  It then cancels the
                                    0000     9 ;       TQE request, and deallocates the block of pool being used to
                                    0000    10 ;       contain the TCE routine.
                                    0000    11 ;
                                    0000    12 ; SIDE EFFECTS:
                                    0000    13 ;
                                    0000    14 ;       Non-paged pool is returned to the system.
                                    0000    15 ;
                                    0000    16 ; PROGRAMMER:
                                    0000    17 ;
                                    0000    18 ;       Vik Muiznieks   15-MAY-1980
                                    0000    19 ;
                                    0000    20 ;--
                                    0000    21 ;
                                    0000    22 ;       External symbols
                                    0000    23         $IPLDEF                                 ; IPL definitions
                                    0000    24         $TQEDEF                                 ; TQE definitions
                                    0000    25 ;
                                    0000    26 ;       Local symbols
                          0000000C  0000    27 HEADER = 12                                     ; header size for code block
                          00000003  0000    28 LOOP_CNT = 3                                    ; loop counter
                                    0000    29 ;
                                    0000    30 ;       Local storage
                               00000000     31         .PSECT  NONSHARED_DATA  PIC, NOEXE, LONG
                          00000122' 0000    32 LKWSET: .ADDRESS START_LOCK                     ; starting address
                          00000140' 0004    33         .ADDRESS END_LOCK                       ; ending address
                              0000  0008    34 TTCHAN: .WORD   0                               ; TT channel
  24 53 59 53 00000012'010E0000'  000A    35 TT:     .ASCID  /SYS$COMMAND/                   ; descriptor for terminal
                    44 4E 41 4D 4D  0018
                          00000014' 001D    36 CTR:    .LONG   STR_END - STRING                ; $FAO control string
                          00000048' 0021    37         .ADDRESS STRING                         ; descriptor
                          0000001E' 0025    38 CTR1:   .LONG   STR1_END - STR                  ; $FAO control string
                          0000002D' 0029    39         .ADDRESS STR                            ; descriptor
8 45 20 6E 69 20 65 75 6C 61 56   002D    40 STR:    .ASCII  *Value in EXE$GL_SITESPEC = !XL*; converts to hexadecimal
5 50 53 45 54 49 53 5F 4C 47 24   0039
                4C 58 21 20 3D 20   0045
                                    0048    41 STR1_END:
9 66 20 6E 69 20 65 75 6C 61 56   0048    42 STRING: .ASCII  *Value in field = !XL*          ; converts to hexadecimal
       4C 58 21 20 3D 20 64 6C   0057
                                    005F    43 STR_END:
                          00000000  005F    44 FAOLEN: .LONG                                   ; $FAO output length
                          00000023  0063    45 OUT:    .LONG   35                              ; Output string desc.
                          0000006B' 0067    46         .ADDRESS BUFF
                          0000008E  006B    47 BUFF:   .BLKB   35                              ; Actual output string
                                    008E    48 BAD_MESSAGE:                                    ; used in case MAKETQE
6F 72 70 20 45 51 54 45 4B 41 4D  008E    49         .ASCII  /MAKETQE program has not been run./ ; not yet run
74 6F 6E 20 73 61 68 20 6D 61 72  009A
    2E 6E 75 72 20 6E 65 65 62   00A6
                          00000021  00AF    50 BAD_SIZE = . - BAD_MESSAGE
                                    00AF    51 ;
```

## Listing 1-4: STOPTQE Example (Page 2 of 2)



```
STOPTQE                          -- Removes TQE from timer queue         14-MAY-1982 16:28:43  VAX-11 Macro V03-00
V01                                                                      27-MAR-1982 16:30:34  WORK:[MUIZNIEKS.SYSPRG.PRIVC

                         00AF     52 :        Entry point for routine
                     00000000     53          .PSECT  CODE     PIC, SHR, NOWRT
                    0000 0000     54 START:   .WORD   0                           ; null entry mask
                         0002     55          $CMKRNL_S        ROUTIN=10$         ; enter kernel mode
                         0011     56 :        Note that most of the work being done in kernel mode by this
                         0011     57 :        example really could be done in user mode.  There is not much
                         0011     58 :        need to enter kernel mode before label START_LOCK.
                      04 0011     59          RET                                 ; all done
                    007C 0012     60 10$:     .WORD   ^M<R2,R3,R4,R5,R6>          ; save registers used
                         0014     61          $LKWSET_S INADR=LKWSET              ; lock pages in working se
             01 50    E8 0025     62          BLBS    R0,15$                      ; proceed on success
                      04 0028     63          RET                                 ; stop on error
                         0029     64 15$:     $ASSIGN_S DEVNAM=TT,CHAN=TTCHAN     ; get channel to terminal
             3D 50    E9 003E     65          BLBC    R0,25$                      ; exit on error
  52   00000000'GF    D0 0041     66 20$:     MOVL    G^EXE$GL_SITESPEC,R2        ; get TQE address
                         0048     67                                              ; if negative, system addr
                37    19 0048     68          BLSS    30$                         ; stop if not negative
                         004A     69          $OUTPUT CHAN=TTCHAN,LENGTH=#BAD_SIZE,BUFFER=BAD_MESSAGE
                         006F     70          $DASSGN_S CHAN=TTCHAN               ; deassign terminal channel
                      04 0070     71          RET                                 ; all done
              0003    31 007E     72 25$:     BRW     ERROR                       ; solve BLBC byte displacem
         56  0C A2    D0 0381     73 30$:     MOVL    TQE$L_FPC(R2),R6            ; get code address
           56    0C   C2 0085     74          SUBL2   #HEADER,R6                  ; point to update location
           56    03   9A 0088     75          MOVZBL  #LOOP_CNT,R4                ; set loop count
                         0088     76          $FAO_S  CTRSTR=CTR1,OUTLEN=FAOLEN,- ; format EXE$GL_SITESPEC
                         0089     77                  OUTBUF=OUT,P1=R2            ; for debugging
             D5 50    E9 00A6     78          BLBC    R0,25$                      ; test for errors
                         00A9     79          $OUTPUT CHAN=TTCHAN,LENGTH=FAOLEN,BUFFER=BUFF ; print value
             A9 50    E9 0002     80          BLBC    R0,25$                      ; test for errors
                         0005     81 40$:     $FAO_S  CTRSTR=CTR,OUTLEN=FAOLEN,-  ; format counter which
                         0005     82                  OUTBUF=OUT,P1=(R6)          ; changes every .1 seconds
             8B 50    E9 00F0     83          BLBC    R0,25$                      ; check for error
                         00F3     84          $OUTPUT CHAN=TTCHAN,LENGTH=FAOLEN,BUFFER=BUFF ; display counter
             35 50    E9 011C     85          BLBC    R0,ERROR                    ; check for error
             83 54    F5 011F     86          SOBGTR  R4,40$                      ; loop a few times
                         0122     87 START_LOCK:                                  ; code must be locked in
                         0122     88                                              ; working set so no page
                         0122     89                                              ; faults above IPL 2
                         0122     90          SETIPL  #IPL$_SYNCH                 ; raise IPL to synch
           50    62   0F 0125     91          REMQUE  (R2),R0                     ; remove TQE from queue
       00000000'GF    16 0128     92          JSB     G^EXE$DEANONPAGED           ; deallocate TQE
           50    56   D0 012E     93          MOVL    R6,R0                       ; get address of code bloc
       00000000'GF    16 0131     94          JSB     G^EXE$DEANONPAGED           ; deallocate code block
       000000C0'GF    D4 0137     95          CLRL    G^EXE$GL_SITESPEC           ; clean-up location so thi
                         013D     96                                              ; program cannot be rerun
                         013D     97                                              ; until MAKETQE rerun
                         013D     98          SETIPL  #0                          ; enable interrupts
                         0140     99 END_LOCK:                                    ; end of locked down code
                         0140    100          $DASSGN_S CHAN=TTCHAN               ; deassign terminal channe
  50      0000'8F     3C 014E    101          MOVZWL  #SS$_NORMAL,R0              ; return success status
                      04 0153    102          RET                                 ; all done
           56    50   D0 0154    103 ERROR:   MOVL    R0,R6                       ; save exit status code
                         0157    104          $DASSGN_S CHAN=TTCHAN               ; deassign terminal channe
           50    56   D0 0165    105          MOVL    R6,R0                       ; restore exit status code
                      04 0168    106          RET                                 ; all done
                         0169    107          .END    START
```

## Listing 1-5: MAKETQE.COM

```
$!                                              MAKETQE.COM
$
$!      This command procedure assembles and links the files
$!      needed for the example that builds a TQE.
$
$!      The debugger is included as well.
$
$ SET VERIFY
$
$ MAC/LIST/ENABLE=DBG MAKETQE + SYS$LIBRARY:LIB/LIB
$ LINK/MAP/FULL/DEBUG MAKETQE, SYS$SYSTEM:SYS.STB/SELECTIVE
$ MAC/LIST STOPTQE + SYS$LIBRARY:LIB/LIB
$ LINK/MAP/FULL STOPTQE, SYS$SYSTEM:SYS.STB/SELECTIVE
$
$!      The TQEDEF.OBJ file can be useful when debugging with
$!      SDA, since it can be read in, and used to FORMAT the
$!      TQE block.
$
$ MAC TQEDEF + SYS$LIBRARY:LIB/LIB
$
$!      Prepare for running/debugging the programs
$
$ SET PROCESS/PRIV=(CMKRNL)
$ DEFINE LIB$DEBUG DELTA
$
$ SET NOVERIFY
```

## SAMPLE RUN

```
$ SET PROCESS/PRIV=(CMKRNL)
$
$ RUN/NODEBUG MAKETQE
$
$ RUN/NODEBUG MAKETQE
%SYSTEM-F-IVMODE, invalid mode for requested function
$
$ RUN/NODEBUG STOPTQE
Value in EXE$GL_SITESPEC = 80112FE0
Value in field = 000000D9
Value in field = 000000DC
Value in field = 000000DE
$
$ RUN/NODEBUG STOPTQE
MAKETQE program has not been run.
$
$ RUN/NODEBUG MAKETQE
$
$ RUN/NODEBUG STOPTQE
Value in EXE$GL_SITESPEC = 80110760
Value in field = 00000047
Value in field = 00000049
Value in field = 0000004C
$
```

## SDA Command Summary

| Command | Function |
|---|---|
| COPY file | Copies the dump file |
| DEFINE sym = exp | Defines symbols and their values |
| EVALUATE exp | Performs computations |
| EXAMINE loc[:loc]<br>loc[;len]<br>/P0 /P1 /SYSTEM /ALL /INSTRUCTION | Examines memory locations |
| EXIT | Exits from the display or from SDA |
| FORMAT<br>/TYPE=block | Formats data blocks |
| HELP | Prints help files |
| READ file | Copies object module symbols |
| REPEAT or <ESC> | Repeats the last command |
| SET OUTPUT file | Directs output to file |
| SET PROCESS name<br>/INDEX=nn /SYSTEM | Sets process context to specific process |
| SHOW CRASH | Displays crash information |
| SHOW DEVICE devnam | Displays I/O data base structures |
| SHOW PAGE_TABLE<br>/GLOBAL /SYSTEM /ALL | Displays system page table |
| SHOW PFN_DATA<br>/FREE /MODIFIED /BAD<br>/SYSTEM /ALL | Displays the PFN data base |
| SHOW POOL<br>/IRP /NONPAGED /PAGED<br>/SUMMARY /ALL | Displays dynamic memory |
| SHOW PROCESS name<br>/INDEX=nn /SYSTEM<br>other qualifiers | Displays specific process information |
| SHOW STACK<br>/INTERRUPT /KERNEL<br>/EXECUTIVE /USER<br>/SUPERVISOR /ALL | Displays process/interrupt stacks |
| SHOW SUMMARY | Displays a summary of all processes |
| SHOW SYMBOL symbol<br>/ALL | Displays the symbol table |
| Operators | + - * / @ (shift) |

| | | |
|---|---|---|
| Symbols | . | Current location |
| | G | 8ØØØØØØØ (hex) |
| | H | 7FFEØØØØ (hex) |
| | RØ-R11 | General registers |
| | AP | Argument pointer |
| | FP | Frame pointer |
| | KSP, ESP, SSP, USP | Stack pointers |
| | PxBR, PxLR | Base/Length registers |
| | PC | Program counter |
| | PSL | Processor status longword |

## Debugger Comparison

- Symbolic Debugger used only for user mode code
- XDELTA/DELTA used for any access mode code
  - Same command syntax
  - No visible prompt
  - Non-symbolic
  - Only error message is EH?

| XDELTA | DELTA |
|---|---|
| Debug operating system or device drivers | Debug user images |
| Used only at console | Used from any terminal |
| Can debug code at any IPL | Can only debug code at IPL=0 |
| Must be specifically requested on boot | Assembled (compiled) and linked with image |
| | Included at run time using $DEFINE LIB$DEBUG DELTA |

## Using PATCH

- When to use PATCH
  - On executable or shareable image files
  - Source program not available
  - Takes too long to reassemble and relink large application
- When NOT to use patch
  - On DIGITAL-supplied software (invalidates warranty)
- Inputs to PATCH
  - Name of image file to be modified
  - PATCH commands to be executed
    - From terminal
    - From command procedure
- Outputs from PATCH
  - Journal file, containing a record of PATCH session
  - Updated image file, if issue UPDATE command
  - Command procedure containing PATCH commands used, if issue CREATE command
- General PATCH use
  1. Invoke PATCH ($ PATCH)
  2. SET ECO level (recommended)
  3. Issue PATCH commands to change image file
  4. Apply patches with UPDATE command
  5. Exit from PATCH

# WRITING A USER-WRITTEN SYSTEM SERVICE

# WRITING A USER-WRITTEN SYSTEM SERVICE

## INTRODUCTION

VAX/VMS provides a set of more than 100 system services and RMS services, implemented as procedures, for non-privileged users to perform privileged functions. Most of the system services execute in kernel mode, while the RMS services execute in executive mode. User-written system services are a set of site-specific procedures that perform privileged functions for non-privileged users, to achieve some site-specific purpose.

This module examines how user-written system services are written, how they can be called by non-privileged users, and how they can be integrated into the operating system. The discussion will focus on the use of various template files provided on the system. These files can be modified to incorporate site-specific user-written system services.

Throughout the discussion, many of the issues and techniques described in the previous module related to system programming will be applied to specific examples.

# WRITING A USER-WRITTEN SYSTEM SERVICE

## OBJECTIVES

- Assemble, link, and install a user-written system service.
- Modify the system-supplied dispatcher to include a user-written system service.
- Write a user-written system service.
- Write a T-BIT dispatcher.

## RESOURCES

VAX/VMS Real-Time User's Guide

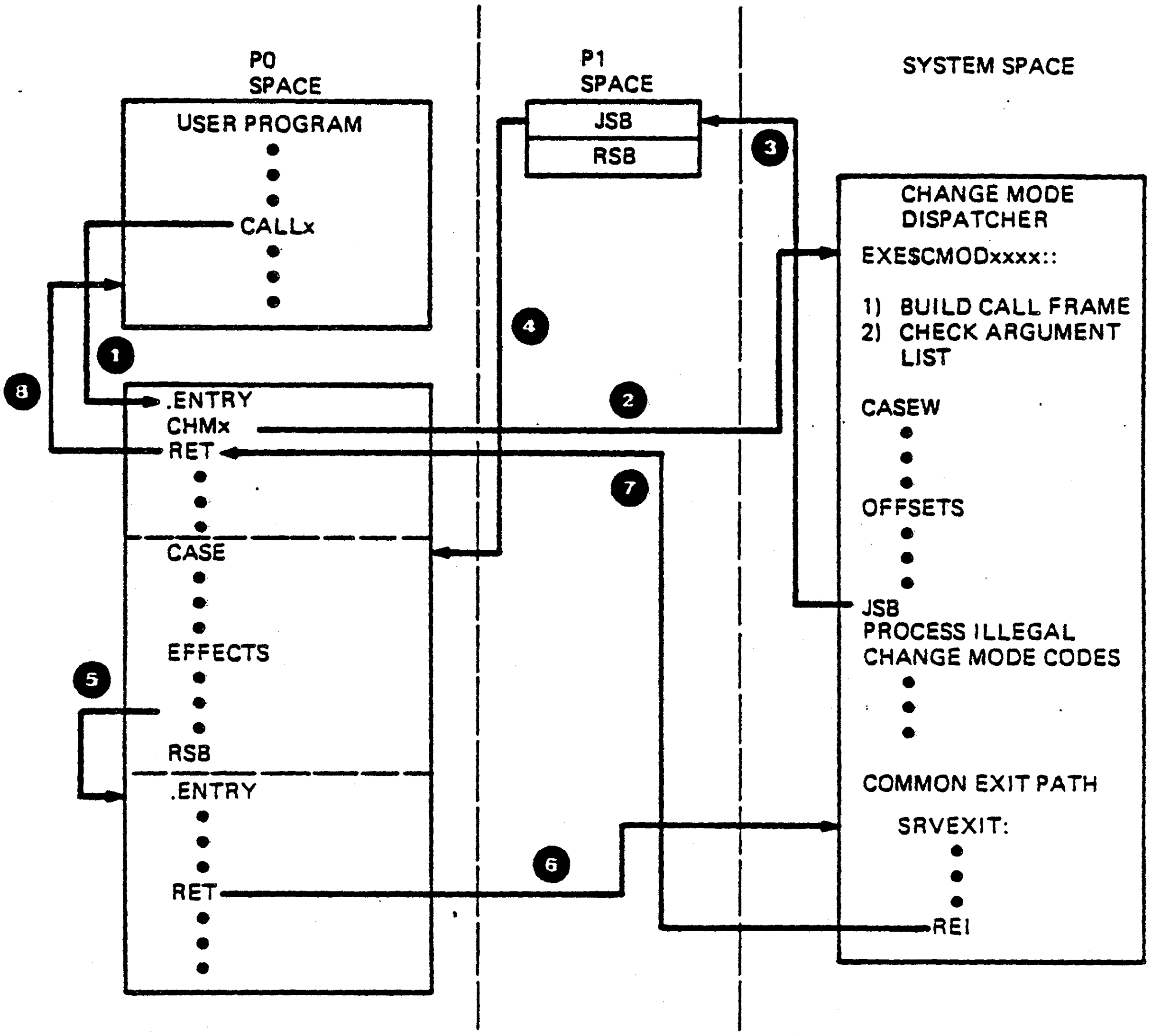
VAX-11 Linker Reference Manual

USS* files in SYS$EXAMPLES:

## Components of User-Written System Services

- Provided in SYS$EXAMPLES:USSDISP.MAR
- Transfer vector (system-supplied)
  - Contains all entry points for system services
  - Located at lowest virtual address in shareable image
    - Allows revision of system service without relinking images that use it
  - Built by DEFINE_SERVICE macro
- Privileged library vector (system-supplied)
  - In PSECT with VEC attribute
  - Contains offsets to executive and kernel mode dispatchers
  - Contains offset to routine that should be called at image rundown time
  - Contains information for validation purposes (e.g., system version number)
  - Used by image activator to connect system service to VMS change mode dispatcher
- Kernel and executive mode dispatchers (system-supplied)
  - Called by VMS change mode dispatcher
  - Decide whether system service valid
  - Verify correct number of arguments for service
  - Transfer control to system service code
- System service code (user-supplied)
  - Usually written in MACRO or BLISS
  - High level languages not recommended, since sometimes:
    - Require run time support routines
    - Make excessive use of stack
    - Unable to generate PIC code
- Rundown routine (user-supplied, optional)
  - Entered in kernel mode with JSB

## Review of User-Written System Service Dispatching



TK-9166

- Mulitple dispatchers can be linked to image
- Dispatchers are searched in order linked
- Negative CHMx codes identify user-written system services
- Duplicate CHMx codes allowed, only first occurrence recognized

## Building a User-Written System Service

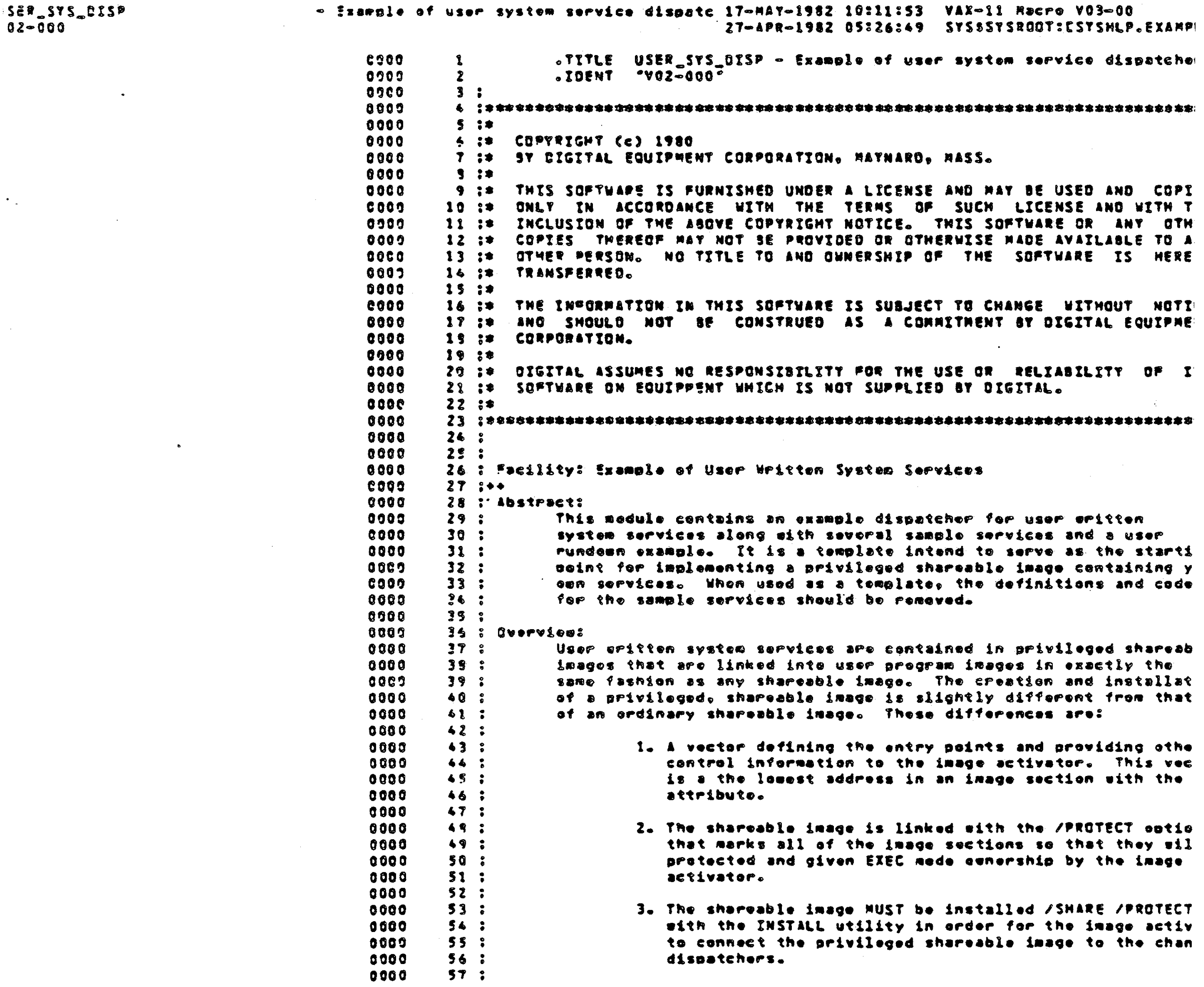
- Start with SYS$EXAMPLES:USSDISP.MAR
  - Edit out unneeded portions, such as:
    - Executive mode dispatcher
    - Example services and their definitions
    - Sample rundown routine
  - Add DEFINE_SERVICE entry for each new system service
  - Add code (if MACRO) for each system service
  - Maybe change base value of -1Ø24 for KCODE_BASE (o ECODE_BASE) to avoid conflicts with other services
- Assembling user-written system service dispatcher file
  - Need to include SYS$LIBRARY:LIB/LIB
  - System supplied dispatcher has .LIBRARY directive
- Linking user-written system service dispatcher file
  - Must use /PROTECT qualifier so all image sections hav EXEC mode page ownership
  - Create separate clusters for transfer vector
  - Do not include run-time library (/NOSYSSHR)
  - Usually include system symbol table
  - Can edit SYS$EXAMPLES:USSLNK.COM

## Building a User-Written System Service

- Linking programs that call user-written system services
  - Must include OPTIONS file
  - OPTIONS file must specify name of file containing user-written system service with /SHARE qualifier

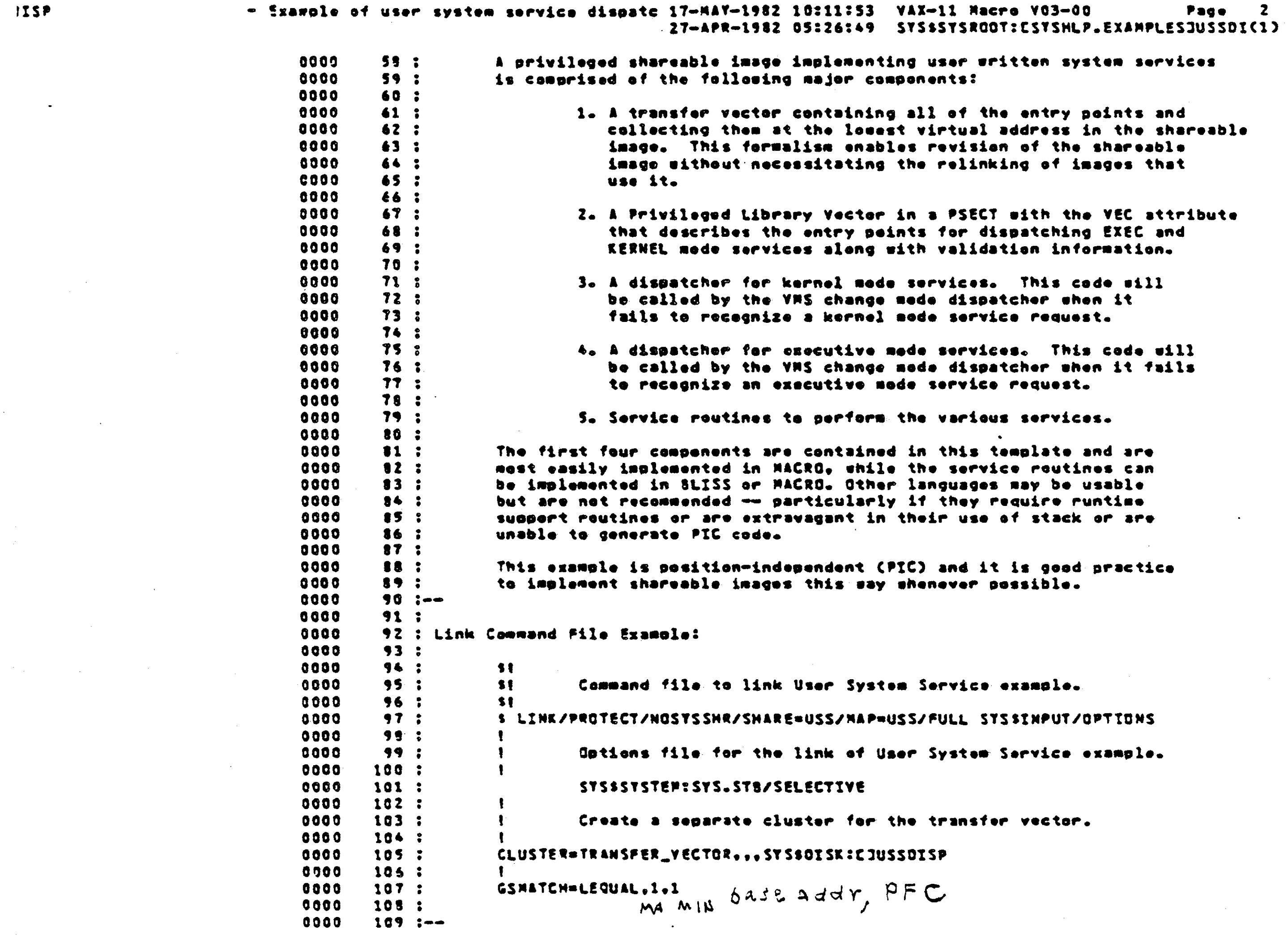
- Installing system service
  - Before programs can be run, system service file must be INSTALLed
  - Allows image activator to connect system service to change mode dispatchers
  - Copy system service file to SYS$SHARE directory
  - INSTALL file /SHARE/PROTECT from SYS$SHARE directory

Listing 2-1: System Supplied Dispatcher File (Page 1 of 12)



```
SER_SYS_DISP                  - Example of user system service dispatc 17-MAY-1982 10:11:53  VAX-11 Macro V03-00
02-000                                                                  27-APR-1982 05:26:49  SYS$SYSROOT:[SYSHLP.EXAMP

                    0000              .TITLE  USER_SYS_DISP - Example of user system service dispatche
                    0000              .IDENT  'V02-000'
                    0000      ;
                    0000      ;*************************************************************************
                    0000      ;*
                    0000      ;*  COPYRIGHT (c) 1980
                    0000      ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                    0000      ;*
                    0000      ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPI
                    0000      ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH T
                    0000      ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTH
                    0000      ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO A
                    0000      ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HERE
                    0000      ;*  TRANSFERRED.
                    0000      ;*
                    0000      ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTI
                    0000      ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPME
                    0000      ;*  CORPORATION.
                    0000      ;*
                    0000      ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  I
                    0000      ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                    0000      ;*
                    0000      ;*************************************************************************
                    0000      ;
                    0000      ;
                    0000      ; Facility: Example of User Written System Services
                    0000      ;++
                    0000      ; Abstract:
                    0000      ;       This module contains an example dispatcher for user written
                    0000      ;       system services along with several sample services and a user
                    0000      ;       rundown example.  It is a template intend to serve as the starti
                    0000      ;       point for implementing a privileged shareable image containing y
                    0000      ;       own services.  When used as a template, the definitions and code
                    0000      ;       for the sample services should be removed.
                    0000      ;
                    0000      ; Overview:
                    0000      ;       User written system services are contained in privileged shareab
                    0000      ;       images that are linked into user program images in exactly the
                    0000      ;       same fashion as any shareable image.  The creation and installat
                    0000      ;       of a privileged, shareable image is slightly different from that
                    0000      ;       of an ordinary shareable image.  These differences are:
                    0000      ;
                    0000      ;               1. A vector defining the entry points and providing othe
                    0000      ;                  control information to the image activator.  This vec
                    0000      ;                  is a the lowest address in an image section with the
                    0000      ;                  attribute.
                    0000      ;
                    0000      ;               2. The shareable image is linked with the /PROTECT optio
                    0000      ;                  that marks all of the image sections so that they wil
                    0000      ;                  protected and given EXEC mode ownership by the image
                    0000      ;                  activator.
                    0000      ;
                    0000      ;               3. The shareable image MUST be installed /SHARE /PROTECT
                    0000      ;                  with the INSTALL utility in order for the image activ
                    0000      ;                  to connect the privileged shareable image to the chan
                    0000      ;                  dispatchers.
                    0000      ;
```

## Listing 2-1: System Supplied Dispatcher File (Page 2 of 12)

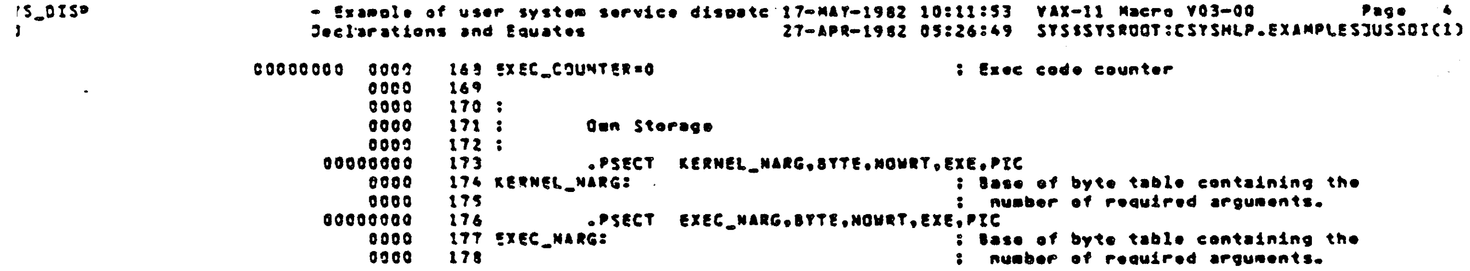


```
ISP                  - Example of user system service dispatc 17-MAY-1982 10:11:53  VAX-11 Macro V03-00        Page   2
                                                               27-APR-1982 05:26:49  SYS$SYSROOT:[SYSHLP.EXAMPLES]USSDI(1)

              0000  :       A privileged shareable image implementing user written system services
              0000  :       is comprised of the following major components:
              0000  :
              0000  :               1. A transfer vector containing all of the entry points and
              0000  :                  collecting them at the lowest virtual address in the shareable
              0000  :                  image.  This formalism enables revision of the shareable
              0000  :                  image without necessitating the relinking of images that
              0000  :                  use it.
              0000  :
              0000  :               2. A Privileged Library Vector in a PSECT with the VEC attribute
              0000  :                  that describes the entry points for dispatching EXEC and
              0000  :                  KERNEL mode services along with validation information.
              0000  :
              0000  :               3. A dispatcher for kernel mode services.  This code will
              0000  :                  be called by the VMS change mode dispatcher when it
              0000  :                  fails to recognize a kernel mode service request.
              0000  :
              0000  :               4. A dispatcher for executive mode services.  This code will
              0000  :                  be called by the VMS change mode dispatcher when it fails
              0000  :                  to recognize an executive mode service request.
              0000  :
              0000  :               5. Service routines to perform the various services.
              0000  :
              0000  :       The first four components are contained in this template and are
              0000  :       most easily implemented in MACRO, while the service routines can
              0000  :       be implemented in BLISS or MACRO. Other languages may be usable
              0000  :       but are not recommended -- particularly if they require runtime
              0000  :       support routines or are extravagant in their use of stack or are
              0000  :       unable to generate PIC code.
              0000  :
              0000  :       This example is position-independent (PIC) and it is good practice
              0000  :       to implement shareable images this way whenever possible.
              0000  :--
              0000  :
              0000  : Link Command File Example:
              0000  :
              0000  :       $!
              0000  :       $!      Command file to link User System Service example.
              0000  :       $!
              0000  :       $ LINK/PROTECT/NOSYSSHR/SHARE=USS/MAP=USS/FULL SYS$INPUT/OPTIONS
              0000  :       !
              0000  :       !       Options file for the link of User System Service example.
              0000  :       !
              0000  :               SYS$SYSTEM:SYS.STB/SELECTIVE
              0000  :       !
              0000  :       !       Create a separate cluster for the transfer vector.
              0000  :       !
              0000  :       CLUSTER=TRANSFER_VECTOR,,,SYS$DISK:[]USSDISP
              0000  :       !
              0000  :       GSMATCH=LEQUAL,1,1
              0000  :                       MA MIN base addr, PFC
              0000  :--
```

## Listing 2-1: System Supplied Dispatcher File (Page 3 of 12)

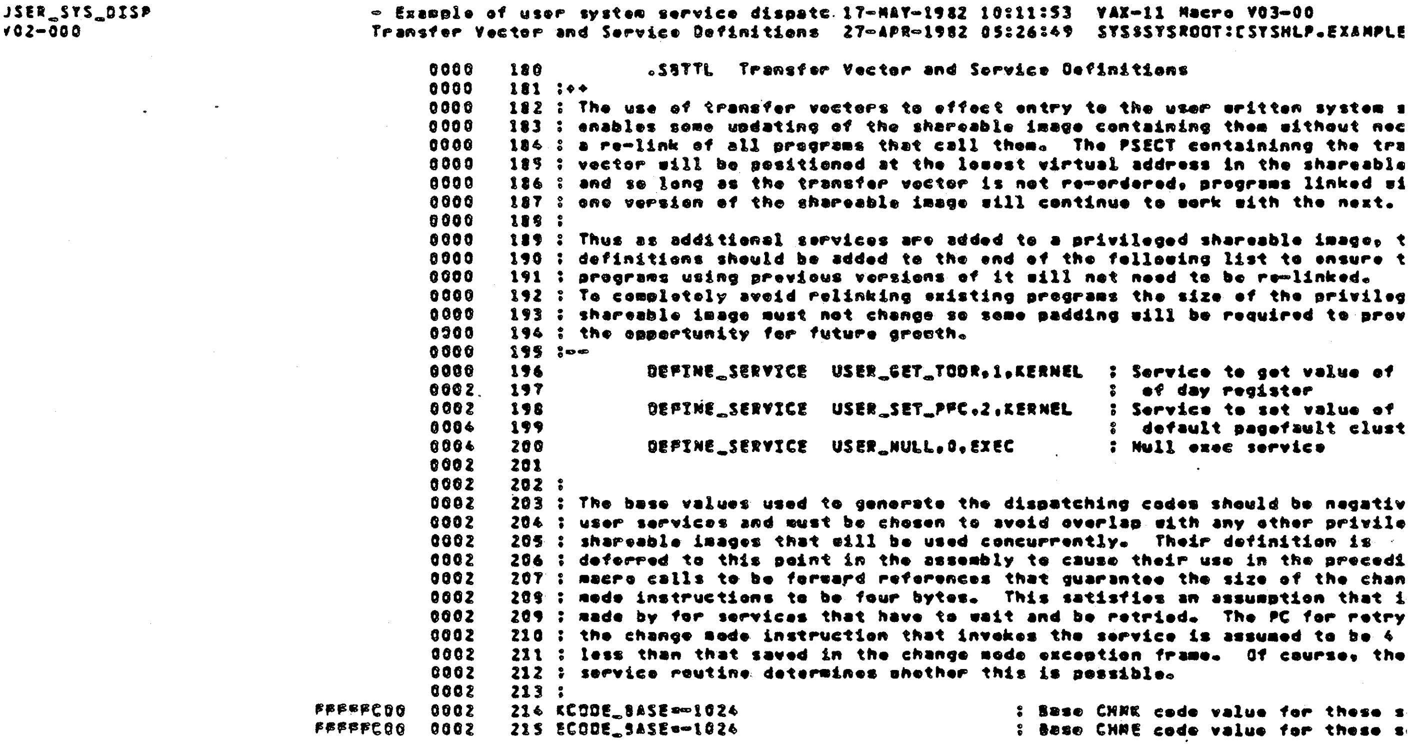
```
:SER_SYS_DIS                - Example of user system service dispatc 17-MAY-1982 10:11:53  VAX-11 Macro V03-00
02-000                      Declarations and Equates                 27-APR-1982 05:26:49  SYS$SYSROOT:[SYSHLP.EXAMPLE

                     0000             .SBTTL  Declarations and Equates
                     0000       ;
                     0000       ;       Include Files
                     0000       ;
                     0000
                     0000             .LIBRARY "SYS$LIBRARY:LIB.MLB"  ; Macro library for system structu
                     0000                                             ;  definitions
                     0000       ;
                     0000       ;       Macro Definitions
                     0000       ;
                     0000       ;       DEFINE_SERVICE - A macro to make the appropriate entries in severa
                     0000       ;                        different PSECTs required to define an EXEC or KE
                     0000       ;                        mode service.  These include the transfer vector,
                     0000       ;                        the case table for dispatching, and a table conta
                     0000       ;                        the number of required arguments.
                     0000       ;
                     0000       ;       DEFINE_SERVICE Name,Number_of_Arguments,Mode
                     0000       ;
                     0000             .MACRO  DEFINE_SERVICE,NAME,NARG=0,MODE=KERNEL
                     0000             .PSECT  $$$TRANSFER_VECTOR,PAGE,NOWRT,EXE,PIC
                     0000             .ALIGN  QUAD                    ; Align entry points for speed and
                     0000             .TRANSFER       NAME            ; Define name as universal symbol
                     0000             .MASK   NAME                    ; Use entry mask defined in main r
                     0000             .IF     IDN MODE,KERNEL
                     0000             CHMK    #<KCODE_BASE+KERNEL_COUNTER> ; Change to kernel mode and e
                     0000             RET                             ; Return
                     0000             KERNEL_COUNTER=KERNEL_COUNTER+1 ; Advance counter
                     0000
                     0000             .PSECT  KERNEL_NARG,BYTE,NOWRT,EXE,PIC
                     0000             .BYTE   NARG                    ; Define number of required argume
                     0000
                     0000             .PSECT  USER_KERNEL_DISP1,BYTE,NOWRT,EXE,PIC
                     0000             .WORD   2+NAME-KCASE_BASE       ; Make entry in kernel mode CASE t
                     0000
                     0000             .IFF
                     0000             CHME    #<ECODE_BASE+EXEC_COUNTER> ; Change to executive mode and
                     0000             RET                             ; Return
                     0000             EXEC_COUNTER=EXEC_COUNTER+1     ; Advance counter
                     0000
                     0000             .PSECT  EXEC_NARG,BYTE,NOWRT,EXE,PIC
                     0000             .BYTE   NARG                    ; Define number of required argume
                     0000
                     0000             .PSECT  USER_EXEC_DISP1,BYTE,NOWRT,EXE,PIC
                     0000             .WORD   2+NAME-ECASE_BASE       ; Make entry in exec mode CASE tab
                     0000             .ENDC                           ;
                     0000             .ENDM   DEFINE_SERVICE          ;
                     0000       ;
                     0000       ;       Equated Symbols
                     0000       ;
                     0000
                     0000             $PHDDEF                         ; Define process header offsets
                     0000             $PLVDEF                         ; Define PLV offsets and values
                     0000             $PRDEF                          ; Define processor register number
                     0000       ;
                     0000       ;       Initialize counters for change mode dispatching codes
                     0000       ;
           00000000  0000       KERNEL_COUNTER=0                      ; Kernel code counter
```

Listing 2-1: System Supplied Dispatcher File (Page 4 of 12)



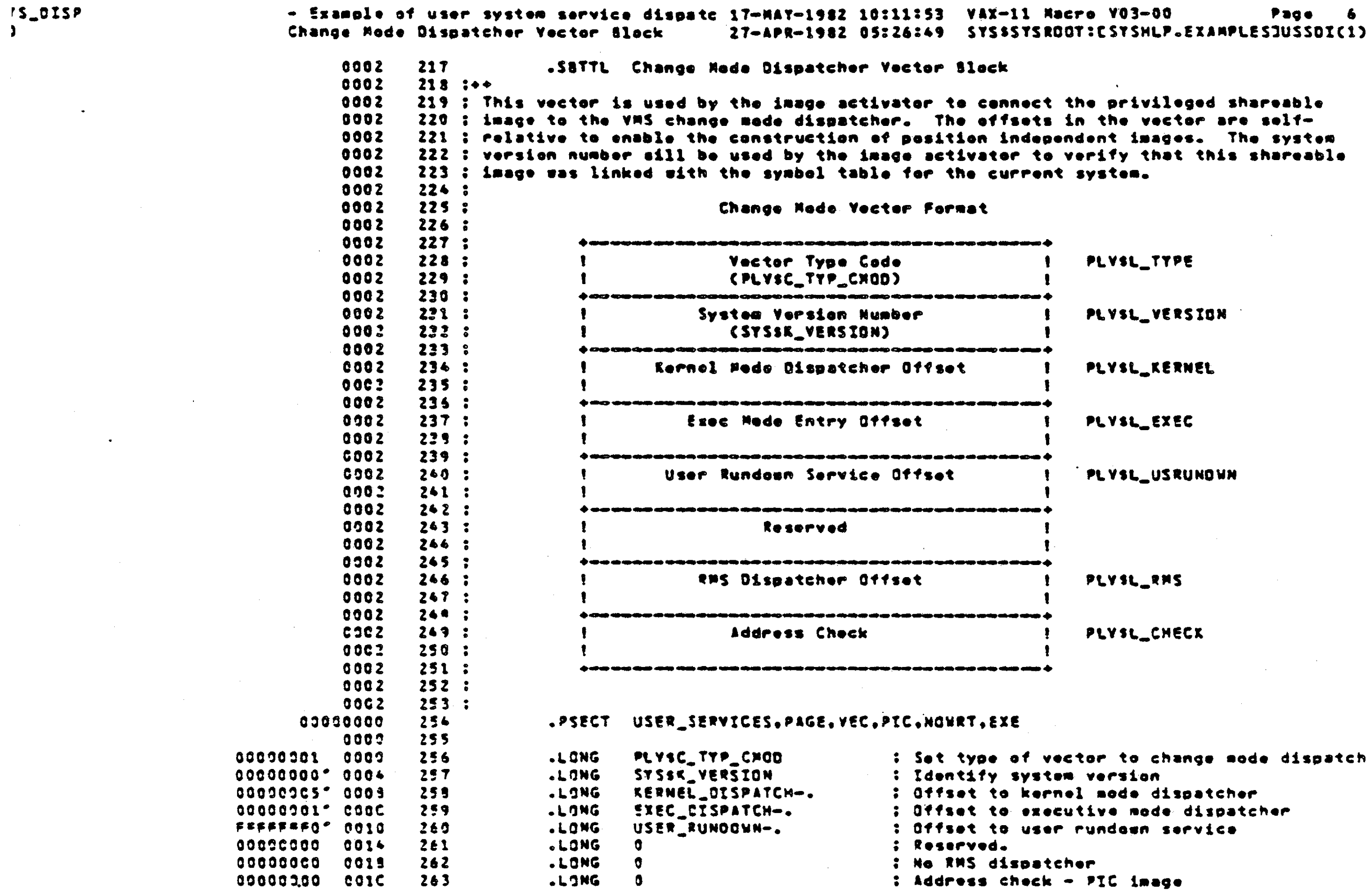
```
'S_DISP                 - Example of user system service dispatc 17-MAY-1982 10:11:53  VAX-11 Macro V03-00          Page    4
)                       Declarations and Equates                 27-APR-1982 05:26:49  SYS$SYSROOT:[SYSHLP.EXAMPLES]USSDI(1)

              00000000  0000    168 EXEC_COUNTER=0                              ; Exec code counter
                        0000    169
                        0000    170 ;
                        0000    171 ;       Own Storage
                        0000    172 ;
                    00000000    173         .PSECT  KERNEL_NARG,BYTE,NOWRT,EXE,PIC
                        0000    174 KERNEL_NARG:                                ; Base of byte table containing the
                        0000    175                                             ;  number of required arguments.
                    00000000    176         .PSECT  EXEC_NARG,BYTE,NOWRT,EXE,PIC
                        0000    177 EXEC_NARG:                                  ; Base of byte table containing the
                        0000    178                                             ;  number of required arguments.
```

## Listing 2-1: System Supplied Dispatcher File (Page 5 of 12)



```
JSER_SYS_DISP                  - Example of user system service dispatc 17-MAY-1982 10:11:53  VAX-11 Macro V03-00
V02-000                        Transfer Vector and Service Definitions  27-APR-1982 05:26:49  SYS$SYSROOT:[SYSHLP.EXAMPLE

                           0000              .SBTTL  Transfer Vector and Service Definitions
                           0000        ;++
                           0000        ; The use of transfer vectors to effect entry to the user written system s
                           0000        ; enables some updating of the shareable image containing them without nec
                           0000        ; a re-link of all programs that call them.  The PSECT containinng the tra
                           0000        ; vector will be positioned at the lowest virtual address in the shareable
                           0000        ; and so long as the transfer vector is not re-ordered, programs linked wi
                           0000        ; one version of the shareable image will continue to work with the next.
                           0000        ;
                           0000        ; Thus as additional services are added to a privileged shareable image, t
                           0000        ; definitions should be added to the end of the following list to ensure t
                           0000        ; programs using previous versions of it will not need to be re-linked.
                           0000        ; To completely avoid relinking existing programs the size of the privileg
                           0000        ; shareable image must not change so some padding will be required to prov
                           0300        ; the opportunity for future growth.
                           0000        ;--
                           0000              DEFINE_SERVICE  USER_GET_TODR,1,KERNEL  ; Service to get value of
                           0002                                                      ;  of day register
                           0002              DEFINE_SERVICE  USER_SET_PFC,2,KERNEL   ; Service to set value of
                           0004                                                      ;  default pagefault clust
                           0004              DEFINE_SERVICE  USER_NULL,0,EXEC        ; Null exec service
                           0002
                           0002        ;
                           0002        ; The base values used to generate the dispatching codes should be negativ
                           0002        ; user services and must be chosen to avoid overlap with any other privile
                           0002        ; shareable images that will be used concurrently.  Their definition is
                           0002        ; deferred to this point in the assembly to cause their use in the precedi
                           0002        ; macro calls to be forward references that guarantee the size of the chan
                           0002        ; mode instructions to be four bytes.  This satisfies an assumption that i
                           0002        ; made by for services that have to wait and be retried.  The PC for retry
                           0002        ; the change mode instruction that invokes the service is assumed to be 4
                           0002        ; less than that saved in the change mode exception frame.  Of course, the
                           0002        ; service routine determines whether this is possible.
                           0002        ;
                FFFFFC00   0002        KCODE_BASE=-1024                      ; Base CHMK code value for these s
                FFFFFC00   0002        ECODE_BASE=-1024                      ; Base CHME code value for these s
```

Listing 2-1: System Supplied Dispatcher File (Page 6 of 12)

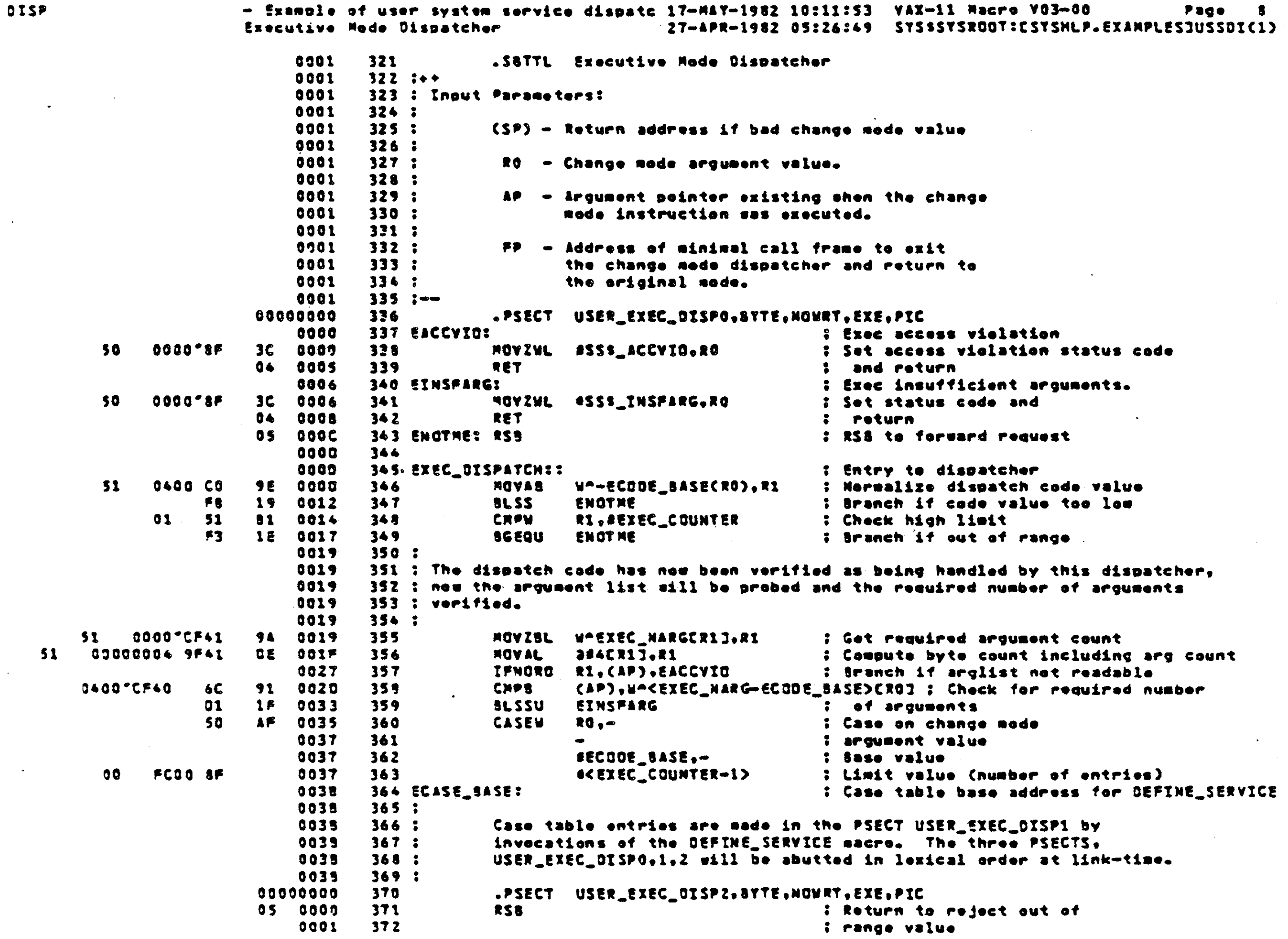


```
'S_DISP                     - Example of user system service dispatc 17-MAY-1982 10:11:53  VAX-11 Macro V03-00          Page   6
)                           Change Mode Dispatcher Vector Block      27-APR-1982 05:26:49  SYS$SYSROOT:[SYSHLP.EXAMPLES]USSDI(1)

                     0002   217         .SBTTL  Change Mode Dispatcher Vector Block
                     0002   218 ;++
                     0002   219 ; This vector is used by the image activator to connect the privileged shareable
                     0002   220 ; image to the VMS change mode dispatcher.  The offsets in the vector are self-
                     0002   221 ; relative to enable the construction of position independent images.  The system
                     0002   222 ; version number will be used by the image activator to verify that this shareable
                     0002   223 ; image was linked with the symbol table for the current system.
                     0002   224 ;
                     0002   225 ;                           Change Mode Vector Format
                     0002   226 ;
                     0002   227 ;              +------------------------------------------+
                     0002   228 ;              !             Vector Type Code             !   PLV$L_TYPE
                     0002   229 ;              !             (PLV$C_TYP_CMOD)             !
                     0002   230 ;              +------------------------------------------+
                     0002   231 ;              !          System Version Number           !   PLV$L_VERSION
                     0002   232 ;              !             (SYS$K_VERSION)              !
                     0002   233 ;              +------------------------------------------+
                     0002   234 ;              !      Kernel Mode Dispatcher Offset       !   PLV$L_KERNEL
                     0002   235 ;              !                                          !
                     0002   236 ;              +------------------------------------------+
                     0002   237 ;              !          Exec Mode Entry Offset          !   PLV$L_EXEC
                     0002   238 ;              !                                          !
                     0002   239 ;              +------------------------------------------+
                     0002   240 ;              !       User Rundown Service Offset        !   PLV$L_USRUNDWN
                     0002   241 ;              !                                          !
                     0002   242 ;              +------------------------------------------+
                     0002   243 ;              !                 Reserved                 !
                     0002   244 ;              !                                          !
                     0002   245 ;              +------------------------------------------+
                     0002   246 ;              !          RMS Dispatcher Offset           !   PLV$L_RMS
                     0002   247 ;              !                                          !
                     0002   248 ;              +------------------------------------------+
                     0002   249 ;              !              Address Check               !   PLV$L_CHECK
                     0002   250 ;              !                                          !
                     0002   251 ;              +------------------------------------------+
                     0002   252 ;
                     0002   253 ;
                 00000000   254         .PSECT  USER_SERVICES,PAGE,VEC,PIC,NOWRT,EXE
                     0000   255
        00000001     0000   256         .LONG   PLV$C_TYP_CMOD          ; Set type of vector to change mode dispatch
        00000000'    0004   257         .LONG   SYS$K_VERSION           ; Identify system version
        000000C5'    0008   258         .LONG   KERNEL_DISPATCH-.       ; Offset to kernel mode dispatcher
        00000001'    000C   259         .LONG   EXEC_DISPATCH-.         ; Offset to executive mode dispatcher
        FFFFFFF0'    0010   260         .LONG   USER_RUNDOWN-.          ; Offset to user rundown service
        00000000     0014   261         .LONG   0                       ; Reserved.
        00000000     0018   262         .LONG   0                       ; No RMS dispatcher
        00000000     001C   263         .LONG   0                       ; Address check - PIC image
```

## Listing 2-1: System Supplied Dispatcher File (Page 7 of 12)

```
SER_SYS_DISP                    - Example of user system service dispatc 17-MAY-1982 10:11:53  VAX-11 Macro V03-00
02-003                          Kernel Mode Dispatcher                   27-APR-1982 05:26:49  SYS$SYSROOT:[SYSHLP.EXAMP

                               0020    265          .SBTTL  Kernel Mode Dispatcher
                               0020    266 ;++
                               0020    267 ; Input Parameters:
                               0020    268 ;
                               0020    269 ;        (SP) - Return address if bad change mode value
                               0020    270 ;
                               0020    271 ;         R0  - Change mode argument value.
                               0020    272 ;
                               0020    273 ;         R4  - Current PCB Address. (Therefore R4 must be specified in a
                               0020    274 ;                register save masks for kernel routines.)
                               0020    275 ;
                               0020    276 ;         AP  - Argument pointer existing when the change
                               0020    277 ;               mode instruction was executed.
                               0020    278 ;
                               0020    279 ;         FP  - Address of minimal call frame to exit
                               0020    280 ;               the change mode dispatcher and return to
                               0020    281 ;               the original mode.
                               0020    282 ;--
                           00000000    283          .PSECT  USER_KERNEL_DISP0,BYTE,NOWRT,EXE,PIC
                               0000    284 KACCVIO:                                ; Kernel access violation
         50    0000'8F    3C   0000    285          MOVZWL  #SS$_ACCVIO,R0          ; Set access violation status co
                          04   0005    286          RET                             ;  and return
                               0006    287 KINSFARG:                               ; Kernel insufficient arguments.
         50    0000'8F    3C   0006    288          MOVZWL  #SS$_INSFARG,R0         ; Set status code and
                          04   000B    289          RET                             ;  return
                          05   000C    290 KNOTME: RSB                             ; RSB to forward request
                               000D    291
                               000D    292 KERNEL_DISPATCH::                       ; Entry to dispatcher
         51    0400 C0    9E   000D    293          MOVAB   W^-KCODE_BASE(R0),R1    ; Normalize dispatch code value
                    F8    19   0012    294          BLSS    KNOTME                  ; Branch if code value too low
               02   51    B1   0014    295          CMPW    R1,#KERNEL_COUNTER      ; Check high limit
                    F3    1E   0017    296          BGEQU   KNOTME                  ; Branch if out of range
                               0019    297 ;
                               0019    298 ; The dispatch code has now been verified as being handled by this dispa
                               0019    299 ; now the argument list will be probed and the required number of argume
                               0019    300 ; verified.
                               0019    301 ;
      51   0000'CF41      9A   0019    302          MOVZBL  W^KERNEL_NARG[R1],R1    ; Get required argument count
   51   00000004 9F61     DE   001F    303          MOVAL   @#4[R1],R1              ; Compute byte count including a
                               0027    304          IFNORD  R1,(AP),KACCVIO         ; Branch if arglist not readable
     0400'CF40      6C    91   002D    305          CMPB    (AP),W^<KERNEL_NARG-KCODE_BASE>[R0] ; Check for required
                    01    1F   0033    306          BLSSU   KINSFARG                ;  of arguments
                    50    AF   0035    307          CASEW   R0,-                    ; Case on change mode
                               0037    308                  -                       ; argument value
                               0037    309                  #KCODE_BASE,-           ; Base value
         01    FC00 8F         0037    310                  #<KERNEL_COUNTER-1>     ; Limit value (number of entries
                               003B    311 KCASE_BASE:                             ; Case table base address for DE
                               003B    312 ;
                               003B    313 ;        Case table entries are made in the PSECT USER_KERNEL_DISP1 by
                               003B    314 ;        invocations of the DEFINE_SERVICE macro.  The three PSECTS,
                               003B    315 ;        USER_KERNEL_DISP0,1,2 will be abutted in lexical order at link-t
                               003B    316 ;
                           00000000    317          .PSECT  USER_KERNEL_DISP2,BYTE,NOWRT,EXE,PIC
                          05   0000    318          RSB                             ; Return to reject out of
                               0001    319                                          ; range value
```

Listing 2-1: System Supplied Dispatcher File (Page 8 of 12)

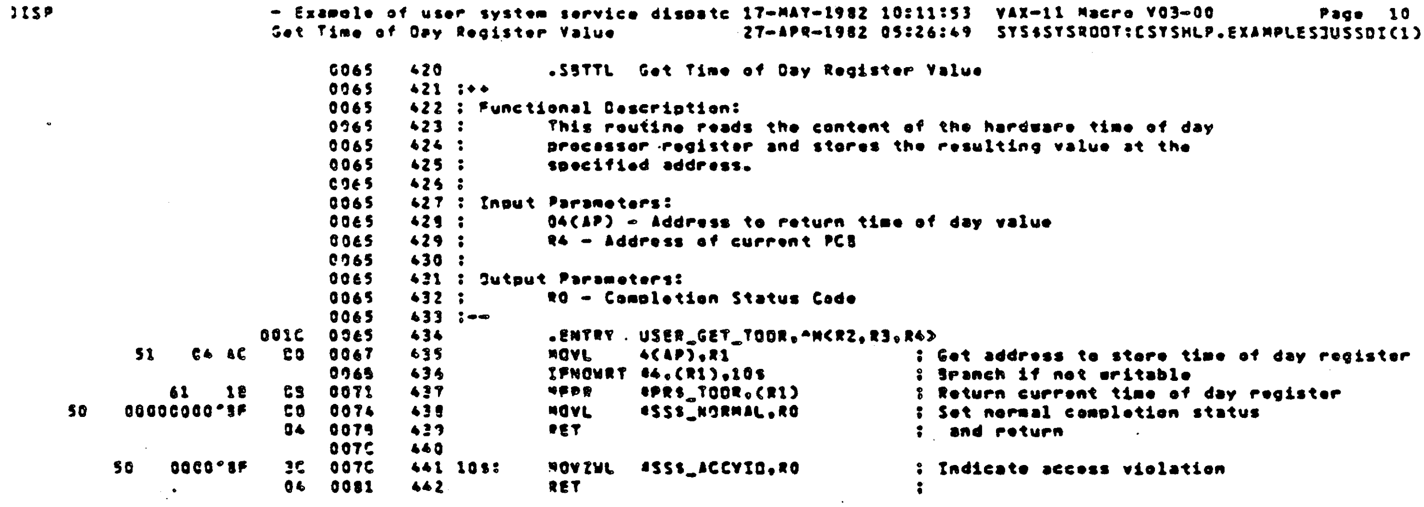


```
DISP                    - Example of user system service dispatc 17-MAY-1982 10:11:53  VAX-11 Macro V03-00          Page   8
                        Executive Mode Dispatcher                27-APR-1982 05:26:49  SYS$SYSROOT:[SYSHLP.EXAMPLES]USSDI(1)

                            0001          .SBTTL  Executive Mode Dispatcher
                            0001     ;++
                            0001     ; Input Parameters:
                            0001     ;
                            0001     ;        (SP) - Return address if bad change mode value
                            0001     ;
                            0001     ;         R0  - Change mode argument value.
                            0001     ;
                            0001     ;         AP  - Argument pointer existing when the change
                            0001     ;               mode instruction was executed.
                            0001     ;
                            0001     ;         FP  - Address of minimal call frame to exit
                            0001     ;               the change mode dispatcher and return to
                            0001     ;               the original mode.
                            0001     ;--
                        00000000          .PSECT  USER_EXEC_DISP0,BYTE,NOWRT,EXE,PIC
                            0000     EACCVIO:                               ; Exec access violation
      50    0000'8F     3C  0000          MOVZWL  #SS$_ACCVIO,R0            ; Set access violation status code
                        04  0005          RET                               ;  and return
                            0006     EINSFARG:                              ; Exec insufficient arguments.
      50    0000'8F     3C  0006          MOVZWL  #SS$_INSFARG,R0           ; Set status code and
                        04  000B          RET                               ;  return
                        05  000C     ENOTME: RSB                            ; RSB to forward request
                            000D
                            000D     EXEC_DISPATCH::                        ; Entry to dispatcher
      51    0400 C0     9E  000D          MOVAB   W^-ECODE_BASE(R0),R1      ; Normalize dispatch code value
                 F8     19  0012          BLSS    ENOTME                    ; Branch if code value too low
            01   51     B1  0014          CMPW    R1,#EXEC_COUNTER          ; Check high limit
                 F3     1E  0017          BGEQU   ENOTME                    ; Branch if out of range
                            0019     ;
                            0019     ; The dispatch code has now been verified as being handled by this dispatcher,
                            0019     ; now the argument list will be probed and the required number of arguments
                            0019     ; verified.
                            0019     ;
    51   0000'CF41      9A  0019          MOVZBL  W^EXEC_NARG[R1],R1        ; Get required argument count
 51   00000004 9F41     DE  001F          MOVAL   @#4[R1],R1                ; Compute byte count including arg count
                            0027          IFNORD  R1,(AP),EACCVIO           ; Branch if arglist not readable
   0400'CF40      6C    91  002D          CMPB    (AP),W^<EXEC_NARG-ECODE_BASE>[R0] ; Check for required number
                 01     1F  0033          BLSSU   EINSFARG                  ;  of arguments
                 50     AF  0035          CASEW   R0,-                      ; Case on change mode
                            0037                  -                         ; argument value
                            0037                  #ECODE_BASE,-             ; Base value
      00    FC00 8F         0037                  #<EXEC_COUNTER-1>         ; Limit value (number of entries)
                            003D     ECASE_BASE:                            ; Case table base address for DEFINE_SERVICE
                            003D     ;
                            003D     ;        Case table entries are made in the PSECT USER_EXEC_DISP1 by
                            003D     ;        invocations of the DEFINE_SERVICE macro.  The three PSECTS,
                            003D     ;        USER_EXEC_DISP0,1,2 will be abutted in lexical order at link-time.
                            003D     ;
                        00000000          .PSECT  USER_EXEC_DISP2,BYTE,NOWRT,EXE,PIC
                        05  0000          RSB                               ; Return to reject out of
                            0001                                            ; range value
```

## Listing 2-1: System Supplied Dispatcher File (Page 9 of 12)

```
US   .YS_DISP                    - Example of user system service dispate 17-MAY-1982 10:11:53  VAX-11 Macro V03-00
V02-000                          User Rundown Service                     27-APR-1982 05:26:49  SYS$SYSROOT:[SYSHLP.EXAMPLE

                                     0001   374          .SBTTL  User Rundown Service
                                     0001   375 ;++
                                     0001   376 ; Functional description:
                                     0001   377 ;        This service is invoked from within the kernel mode system service
                                     0001   378 ;        that performs image rundown.  It is invoked before any system
                                     0001   379 ;        rundown functions (i.e. deassign channels, release memory) are
                                     0001   380 ;        performed.  User code should not invoked any RMS services or RTL
                                     0001   381 ;        routines, must not signal any exceptions.  User code can invoke
                                     0001   382 ;        most system services except those that use RMS (e.g. $PUTMSG).
                                     0001   383 ;
                                     0001   384 ; Calling sequence:
                                     0001   385 ;        JSB     USER_RUNDOWN
                                     0001   386 ;        Entered at IPL=0 and must leave at IPL=0.
                                     0001   387 ;
                                     0001   388 ; Input Parameters:
                                     0001   389 ;          R4  - Current PCB Address. (Therefore R4 must be specified in all
                                     0001   390 ;                  register save masks for kernel routines.)
                                     0001   391 ;
                                     0001   392 ;          R7  - Access mode parameter to $RUNDOWN maximized with previous mo
                                     0001   393 ;
                                     0001   394 ;          AP  - Argument pointer existing when the $RUNDOWN system
                                     0001   395 ;                 service was invoked.
                                     0001   396 ;
                                     0001   397 ;          4(AP) - Access mode parameter to $RUNDOWN
                                     0001   398 ;
                                     0001   399 ;--
                             00000000   400          .PSECT  USER_CODE,BYTE,NOWRT,EXE,PIC
                                 0000   401
                                 0000   402 USER_RUNDOWN::                                     ; Entry point for service
                       52   DD   0000   403          PUSHL   R2                                ; Save a register
                    4A'AF   9F   0002   404          PUSHAB  B^SYSOUT                          ; Set up address of descriptor
                      06'   DD   0005   405          PUSHL   S^#SYS_LEN                        ; Set up length
                  52   7E   DE   0007   406          MOVAL   -(SP), R2                         ; Grab some temporary storage
                                 000A   407          $ASSIGN_S 4(R2), (R2)                     ; Assign a channel to operator con
                   29 50    E9   0018   408          BLBC    R0, 10$                           ; Error
                                 001B   409          $OUTPUT   (R2), S^#MSG_LEN, B^MSG              ; Print the message on ope
                                 0039   410          $DASSGN_S (R2)                            ; Get rid of the channel
                  5E   0C   C0   0043   411 10$:     ADDL2   #12, SP                           ; Clean up
                  52   8E   D0   0046   412          MOVL    (SP)+, R2                         ; Restore register
                            05   0049   413          RSB
                                 004A   414 ;
             3A 30 41 50 4F 5F   004A   415 SYSOUT: .ASCII  /_OPA0:/
                     00000006    0050   416 SYS_LEN=.-SYSOUT
78 65 20 65 67 61 6D 49 20 2A 2A 2A   0050   417 MSG:     .ASCII  /*** Image exiting ***/
      2A 2A 2A 20 67 6E 69 74 69   005C
                     00000015    0065   418 MSG_LEN=.-MSG
```

Listing 2-1: System Supplied Dispatcher File (Page 1Ø of 12)

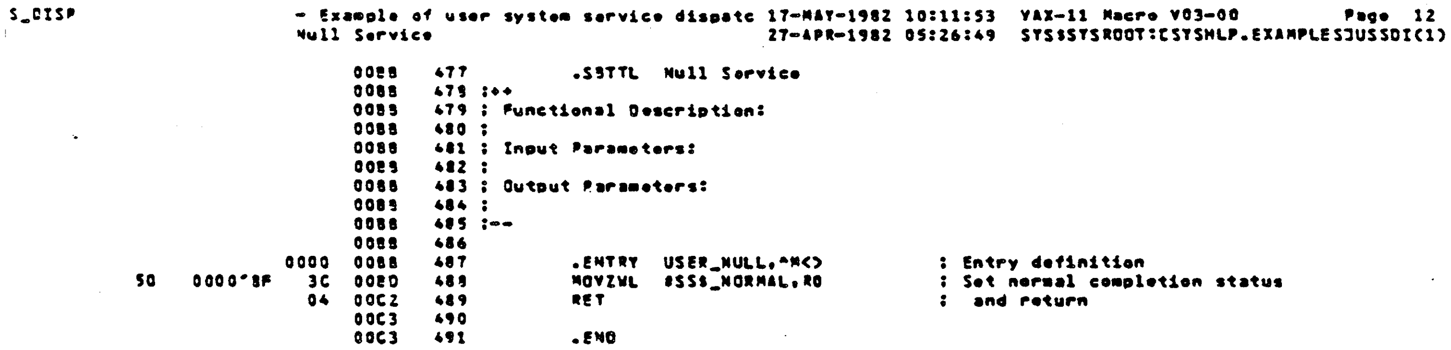


```
DISP                     - Example of user system service dispatc 17-MAY-1982 10:11:53  VAX-11 Macro V03-00        Page  10
                         Get Time of Day Register Value           27-APR-1982 05:26:49  SYS$SYSROOT:[SYSHLP.EXAMPLES]USSDI(1)

                             0065   420          .SBTTL  Get Time of Day Register Value
                             0065   421 ;++
                             0065   422 ; Functional Description:
                             0065   423 ;        This routine reads the content of the hardware time of day
                             0065   424 ;        processor register and stores the resulting value at the
                             0065   425 ;        specified address.
                             0065   426 ;
                             0065   427 ; Input Parameters:
                             0065   428 ;        04(AP) - Address to return time of day value
                             0065   429 ;        R4 - Address of current PCB
                             0065   430 ;
                             0065   431 ; Output Parameters:
                             0065   432 ;        R0 - Completion Status Code
                             0065   433 ;--
                        001C 0065   434          .ENTRY  USER_GET_TODR,^M<R2,R3,R4>
          51    04 AC   D0   0067   435          MOVL    4(AP),R1                ; Get address to store time of day register
                             0069   436          IFNOWRT #4,(R1),10$             ; Branch if not writable
            61   1B     DB   0071   437          MFPR    #PR$_TODR,(R1)          ; Return current time of day register
   50   00000000'8F     D0   0074   438          MOVL    #SS$_NORMAL,R0          ; Set normal completion status
                        04   0079   439          RET                             ;  and return
                             007C   440
        50   0000'8F    3C   007C   441 10$:     MOVZWL  #SS$_ACCVIO,R0          ; Indicate access violation
                        04   0081   442          RET                             ;
```

## Listing 2-1: System Supplied Dispatcher File (Page 11 of 12)

```
SER_SYS_DISP                      - Example of user system service dispatc 17-MAY-1982 10:11:53  VAX-11 Macro V03-00
02-000                            Set Page Fault Cluster Factor            27-APR-1982 05:26:49  SYS$SYSROOT:[SYSHLP.EXAMP

                            0082            .SBTTL  Set Page Fault Cluster Factor
                            0082      ;++
                            0082      ; Functional Description:
                            0082      ;         This routine sets the page fault cluster to the specified value
                            0082      ;         and returns the previous value.
                            0082      ;
                            0082      ; Input Parameters:
                            0082      ;         04(AP) - New value for Page Fault Cluster factor
                            0082      ;         08(AP) - Address to return previous value
                            0082      ;                         (0 means none)
                            0082      ;         R4 - PCB address of current process
                            0082      ;
                            0082      ; Output Parameters:
                            0082      ;         R0 - Completion Status code
                            0082      ;--
                      0030  0082            .ENTRY   USER_SET_PFC,^M<R4,R5>
55   00000000'9F      D0    0084            MOVL     @#CTL$GL_PHD,R5            ; Get address of process header
  51     08 AC        D0    008B            MOVL     8(AP),R1                   ; Get address to store previous
             0A       13    008F            BEQL     10$                        ; Branch if none
                            0091            IFNOWRT  #4,(R1),30$                ; Branch if not writable
  61     34 A5        9A    0097            MOVZBL   PHD$B_DFPFC(R5),(R1)       ; Return current value
  50     04 AC        90    009B      10$:  MOVB     4(AP),R0                   ; Get new value for PFC
  7F 8F     50        91    009F            CMPB     R0,#127                    ; Check for legal value
             04       1B    00A3            BLEQU    20$                        ; Branch if legal
  50     7F 8F        90    00A5            MOVB     #127,R0                    ; Set to maximum value
  34 A5     50        90    00A9      20$:  MOVB     R0,PHD$B_DFPFC(R5)         ; Set new value into PHD
50   00000000'8F      D0    00AD            MOVL     #SS$_NORMAL,R0             ; Set normal completion status
                      04    00B4            RET                                 ;  and return
                            00B5
    50    0000'8F     3C    00B5      30$:  MOVZWL   #SS$_ACCVIO,R0             ; Indicate access violation
                      04    00BA            RET                                 ;
                            00BB
```

## Listing 2-1: System Supplied Dispatcher File (Page 12 of 12)

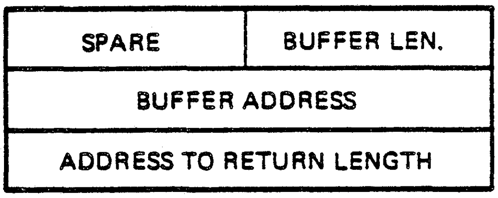


```
S_DISP                       - Example of user system service dispatc 17-MAY-1982 10:11:53  VAX-11 Macro V03-00         Page  12
                             Null Service                             27-APR-1982 05:26:49  SYS$SYSROOT:[SYSHLP.EXAMPLES]USSDI(1)

                                  00B8   477          .SBTTL  Null Service
                                  00B8   478 ;++
                                  00B8   479 ; Functional Description:
                                  00B8   480 ;
                                  00B8   481 ; Input Parameters:
                                  00B8   482 ;
                                  00B8   483 ; Output Parameters:
                                  00B8   484 ;
                                  00B8   485 ;--
                                  00B8   486
                            0000  00B8   487          .ENTRY  USER_NULL,^M<>          ; Entry definition
        50    0000'8F         3C  00BD   488          MOVZWL  #SS$_NORMAL,R0          ; Set normal completion status
                              04  00C2   489          RET                             ;  and return
                                  00C3   490
                                  00C3   491          .END
```

## Sample User-Written System Service

- Allows suitably privileged (GROUP, WORLD) caller to obtain the default directory of any process in the system
- Implemented using ASTs
  - Similar to $GETJPI system service and $SETDDIR RMS call
  - One AST executes in context of target process, and loads default directory string from P1 space into part of AST control block
  - Another AST executes in context of original caller, and returns the default directory string
- Argument list for system service

| | |
|---|---|
| EFN | Event flag number to set when done |
| PIDADR | Address of PID for target process |
| PRCNAM | Address of process name for target process |
| DDDESC | Address of 3-longword descriptor for data |



TK-9185

| | |
|---|---|
| IOSB | Address of longword for final status |
| ASTADR | AST address for notification |
| ASTPRM | AST parameter |

## System Programming Techniques Illustrated

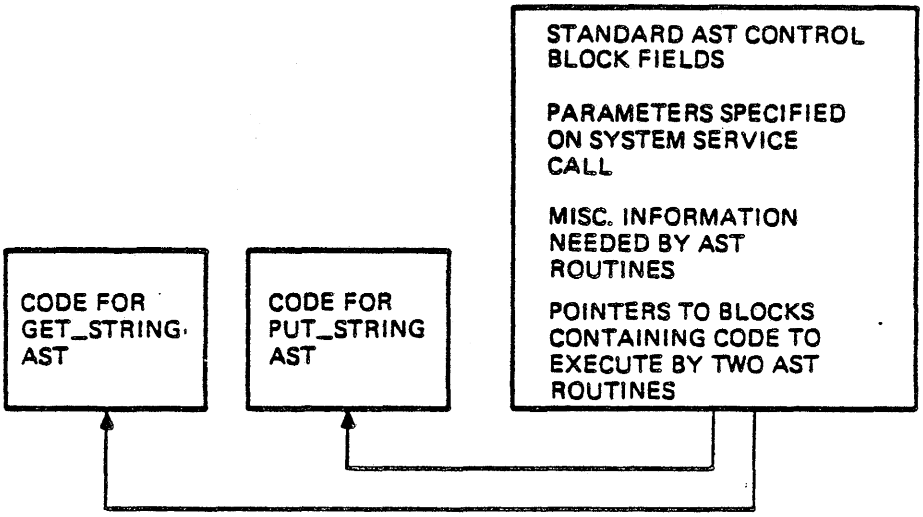
- Making privilege checks
- Making quota checks
- Making memory accessibility checks
- Allocating nonpaged pool
- Using P1 mapping of process header
- Queuing ASTs
- Defining symbolic offsets in data structures
- Accessing system data structures observing synchronization rules
- Converting process name to process id
- Unique features of special kernel ASTs
- Recovering from error conditions
- Guarding against errors in kernel mode for asynchronous operations
  - Access violations
  - Page protection changes
  - Image exits

## Flow of Control in Sample System Service

-1- User program calls USSGETDD

USSGETDD System Service

-2- Allocates and builds ACB data structure


STANDARD AST CONTROL BLOCK FIELDS
PARAMETERS SPECIFIED ON SYSTEM SERVICE CALL
MISC. INFORMATION NEEDED BY AST ROUTINES
POINTERS TO BLOCKS CONTAINING CODE TO EXECUTE BY TWO AST ROUTINES
CODE FOR GET_STRING AST
CODE FOR PUT_STRING AST

TK-9186

-3- Queues up GET_STRING AST

- Executes in context of target process to get default directory string
- Queues up PUT_STRING AST

-4- PUT_STRING AST routine

- Executes in context of caller process
- Returns default directory string to caller
- Queues up AST requested by user (if any)

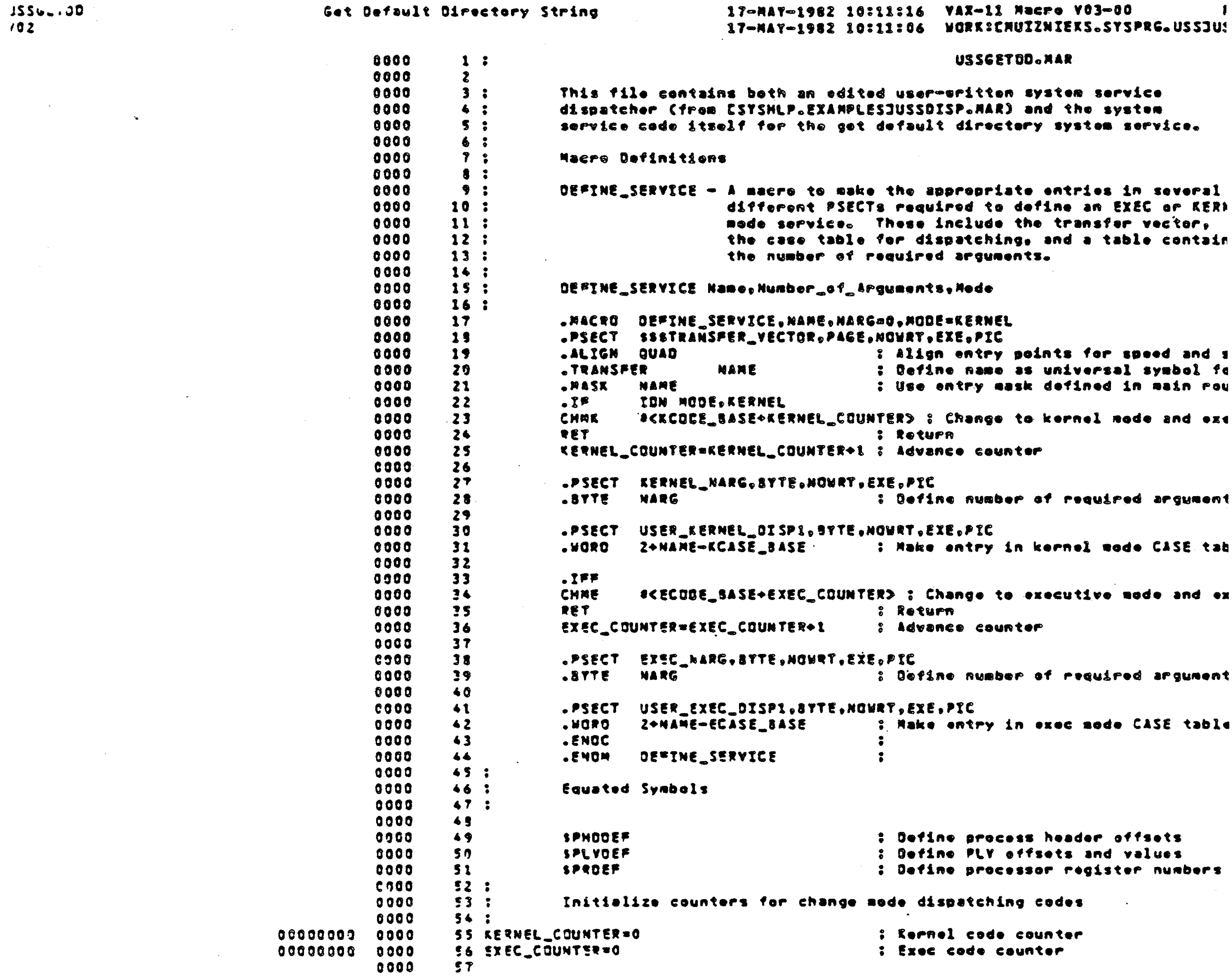
-5- User AST executes (if requested)

-6- Control returned to user program

## The USS_GETDD Procedure

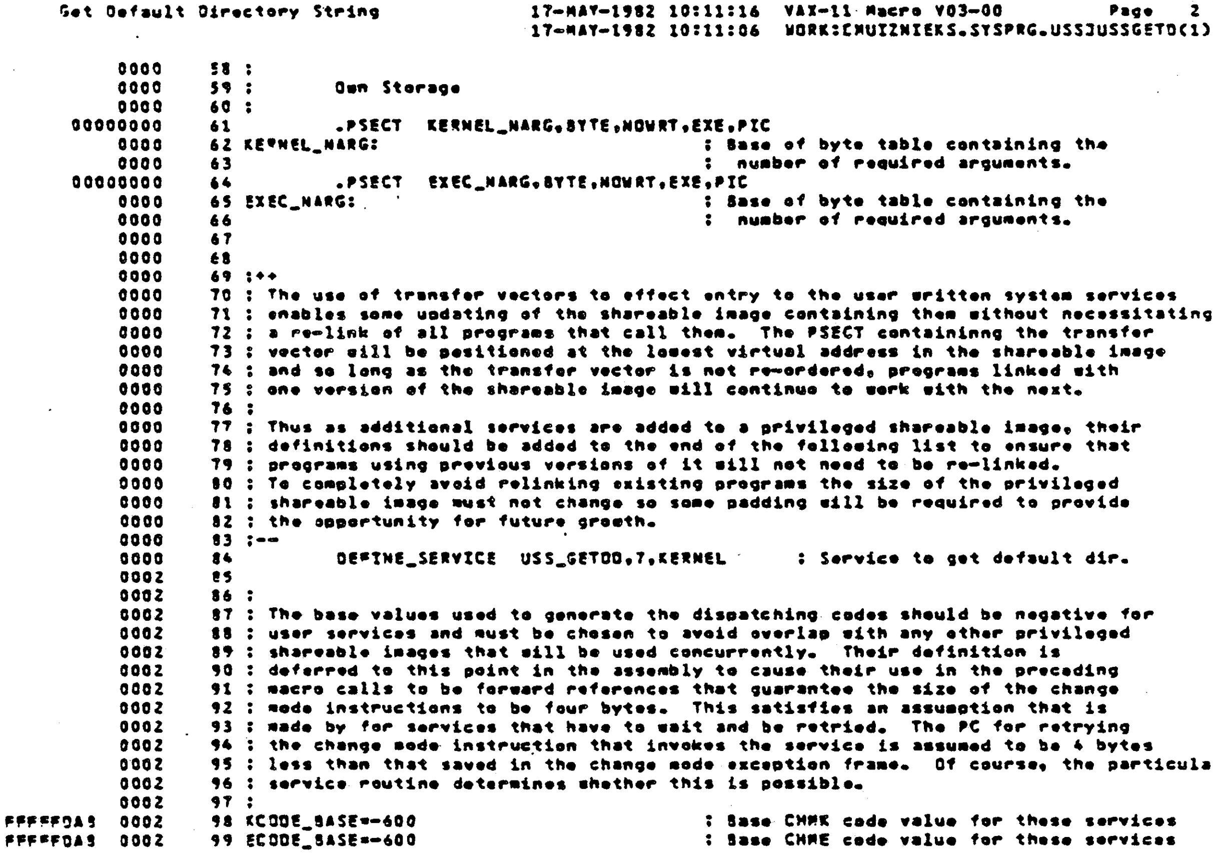
- Entry point for system service
  - Makes various checks to insure appropriate parameters on call
  - Allocates nonpaged pool space for AST control block to queue to target process
  - Allocates nonpaged pool space to hold two segments of code
    - GET string procedure
    - PUT string procedure
- AST control block for target process
  - Contains parameters from system service call
  - Space reserved for default directory string
  - Contains addresses of GET and PUT string procedures
  - Specifies GET string procedure as AST code to execute in context of target process
  - Specifies that AST is a special kernel AST (and cannot be disabled)
- Before queueing AST
  - Raises IPL to SYNCH
  - Checks to see if target process was deleted while allocating nonpaged pool
- Possible optimizations
  - Could check if target process is caller
  - Could allocate one large block of nonpaged pool and store AST block as well as GET and PUT string procedures in it
  - Not done to avoid making complicated example even more complicated

Listing 2-2: USSGETDD Procedure (Page 1 of 12)



```
JSS..DD                 Get Default Directory String         17-MAY-1982 10:11:16  VAX-11 Macro V03-00         I
/02                                                          17-MAY-1982 10:11:06  WORK:[MUIZNIEKS.SYSPRG.USS]U

                 0000    ;                                                   USSGETDD.MAR
                 0000
                 0000    ;         This file contains both an edited user-written system service
                 0000    ;         dispatcher (from [SYSHLP.EXAMPLES]USSDISP.MAR) and the system
                 0000    ;         service code itself for the get default directory system service.
                 0000    ;
                 0000    ;         Macro Definitions
                 0000    ;
                 0000    ;         DEFINE_SERVICE - A macro to make the appropriate entries in several
                 0000    ;                          different PSECTs required to define an EXEC or KER
                 0000    ;                          mode service.  These include the transfer vector,
                 0000    ;                          the case table for dispatching, and a table contain
                 0000    ;                          the number of required arguments.
                 0000    ;
                 0000    ;         DEFINE_SERVICE Name,Number_of_Arguments,Mode
                 0000    ;
                 0000              .MACRO  DEFINE_SERVICE,NAME,NARG=0,MODE=KERNEL
                 0000              .PSECT  $$$TRANSFER_VECTOR,PAGE,NOWRT,EXE,PIC
                 0000              .ALIGN  QUAD                         ; Align entry points for speed and s
                 0000              .TRANSFER       NAME                 ; Define name as universal symbol fo
                 0000              .MASK   NAME                         ; Use entry mask defined in main rou
                 0000              .IF     IDN MODE,KERNEL
                 0000              CHMK    #<KCODE_BASE+KERNEL_COUNTER> ; Change to kernel mode and exe
                 0000              RET                                  ; Return
                 0000              KERNEL_COUNTER=KERNEL_COUNTER+1 ; Advance counter
                 0000
                 0000              .PSECT  KERNEL_NARG,BYTE,NOWRT,EXE,PIC
                 0000              .BYTE   NARG                         ; Define number of required argument
                 0000
                 0000              .PSECT  USER_KERNEL_DISP1,BYTE,NOWRT,EXE,PIC
                 0000              .WORD   2+NAME-KCASE_BASE            ; Make entry in kernel mode CASE tab
                 0000
                 0000              .IFF
                 0000              CHME    #<ECODE_BASE+EXEC_COUNTER> ; Change to executive mode and ex
                 0000              RET                                  ; Return
                 0000              EXEC_COUNTER=EXEC_COUNTER+1          ; Advance counter
                 0000
                 0000              .PSECT  EXEC_NARG,BYTE,NOWRT,EXE,PIC
                 0000              .BYTE   NARG                         ; Define number of required argument
                 0000
                 0000              .PSECT  USER_EXEC_DISP1,BYTE,NOWRT,EXE,PIC
                 0000              .WORD   2+NAME-ECASE_BASE            ; Make entry in exec mode CASE table
                 0000              .ENDC                                ;
                 0000              .ENDM   DEFINE_SERVICE               ;
                 0000    ;
                 0000    ;         Equated Symbols
                 0000    ;
                 0000
                 0000              $PHDDEF                              ; Define process header offsets
                 0000              $PLVDEF                              ; Define PLV offsets and values
                 0000              $PRDEF                               ; Define processor register numbers
                 0000    ;
                 0000    ;         Initialize counters for change mode dispatching codes
                 0000    ;
       00000000  0000    KERNEL_COUNTER=0                               ; Kernel code counter
       00000000  0000    EXEC_COUNTER=0                                 ; Exec code counter
                 0000
```

## Listing 2-2: USSGETDD Procedure (Page 2 of 12)

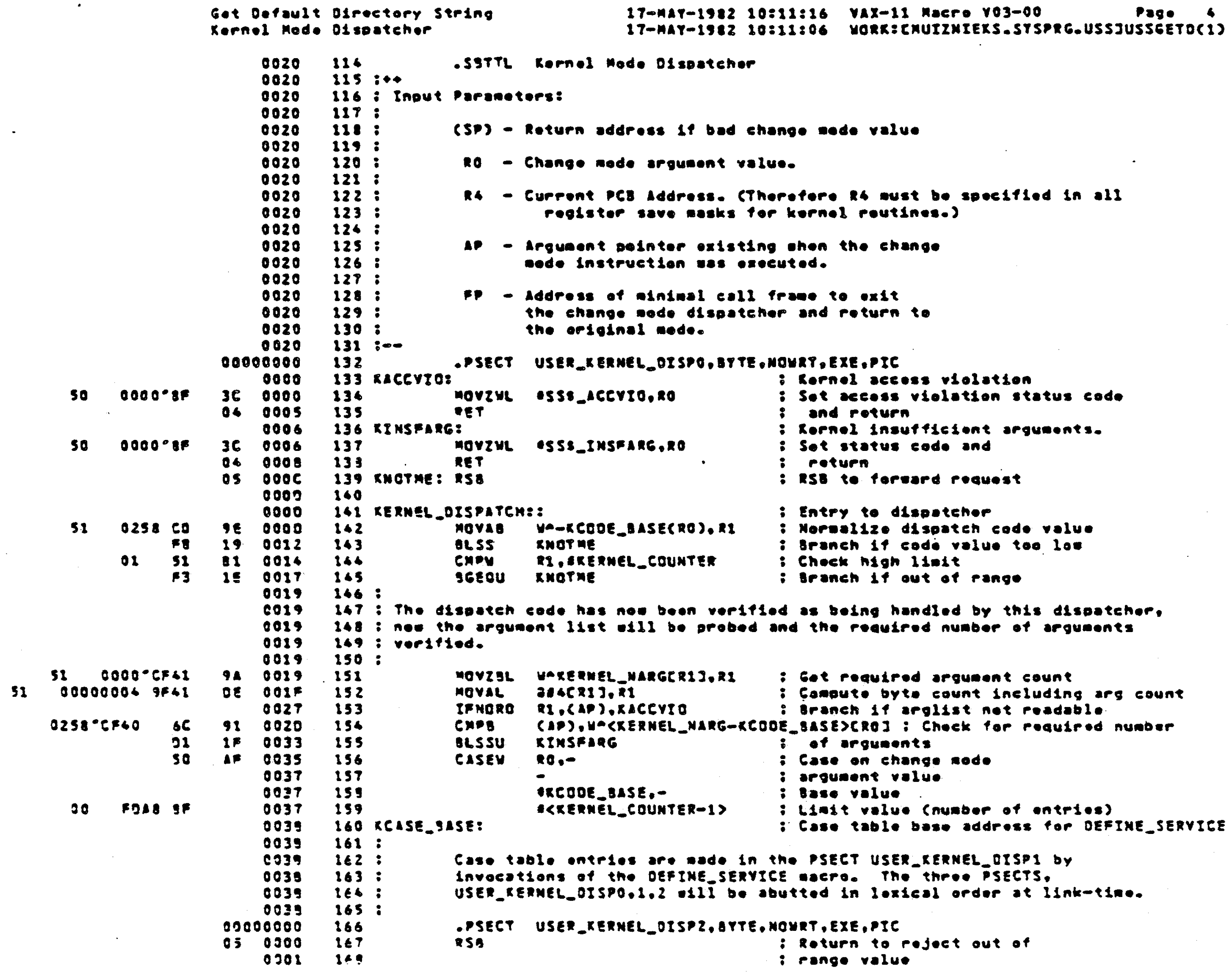


```
Get Default Directory String            17-MAY-1982 10:11:16  VAX-11 Macro V03-00         Page   2
                                        17-MAY-1982 10:11:06  WORK:[MUIZNIEKS.SYSPRG.USS]USSGETD(1)

             0000     ;
             0000     ;       Own Storage
             0000     ;
         00000000             .PSECT  KERNEL_NARG,BYTE,NOWRT,EXE,PIC
             0000     KERNEL_NARG:                            ; Base of byte table containing the
             0000                                             ;  number of required arguments.
         00000000             .PSECT  EXEC_NARG,BYTE,NOWRT,EXE,PIC
             0000     EXEC_NARG:                              ; Base of byte table containing the
             0000                                             ;  number of required arguments.
             0000
             0000
             0000     ;++
             0000     ; The use of transfer vectors to effect entry to the user written system services
             0000     ; enables some updating of the shareable image containing them without necessitating
             0000     ; a re-link of all programs that call them.  The PSECT containinng the transfer
             0000     ; vector will be positioned at the lowest virtual address in the shareable image
             0000     ; and so long as the transfer vector is not re-ordered, programs linked with
             0000     ; one version of the shareable image will continue to work with the next.
             0000     ;
             0000     ; Thus as additional services are added to a privileged shareable image, their
             0000     ; definitions should be added to the end of the following list to ensure that
             0000     ; programs using previous versions of it will not need to be re-linked.
             0000     ; To completely avoid relinking existing programs the size of the privileged
             0000     ; shareable image must not change so some padding will be required to provide
             0000     ; the opportunity for future growth.
             0000     ;--
             0000             DEFINE_SERVICE  USS_GETDD,7,KERNEL        ; Service to get default dir.
             0002
             0002     ;
             0002     ; The base values used to generate the dispatching codes should be negative for
             0002     ; user services and must be chosen to avoid overlap with any other privileged
             0002     ; shareable images that will be used concurrently.  Their definition is
             0002     ; deferred to this point in the assembly to cause their use in the preceding
             0002     ; macro calls to be forward references that guarantee the size of the change
             0002     ; mode instructions to be four bytes.  This satisfies an assumption that is
             0002     ; made by for services that have to wait and be retried.  The PC for retrying
             0002     ; the change mode instruction that invokes the service is assumed to be 4 bytes
             0002     ; less than that saved in the change mode exception frame.  Of course, the particula
             0002     ; service routine determines whether this is possible.
             0002     ;
FFFFFDA8     0002     KCODE_BASE=-600                         ; Base CHMK code value for these services
FFFFFDA8     0002     ECODE_BASE=-600                         ; Base CHME code value for these services
```

## Listing 2-2: USSGETDD Procedure (Page 3 of 12)

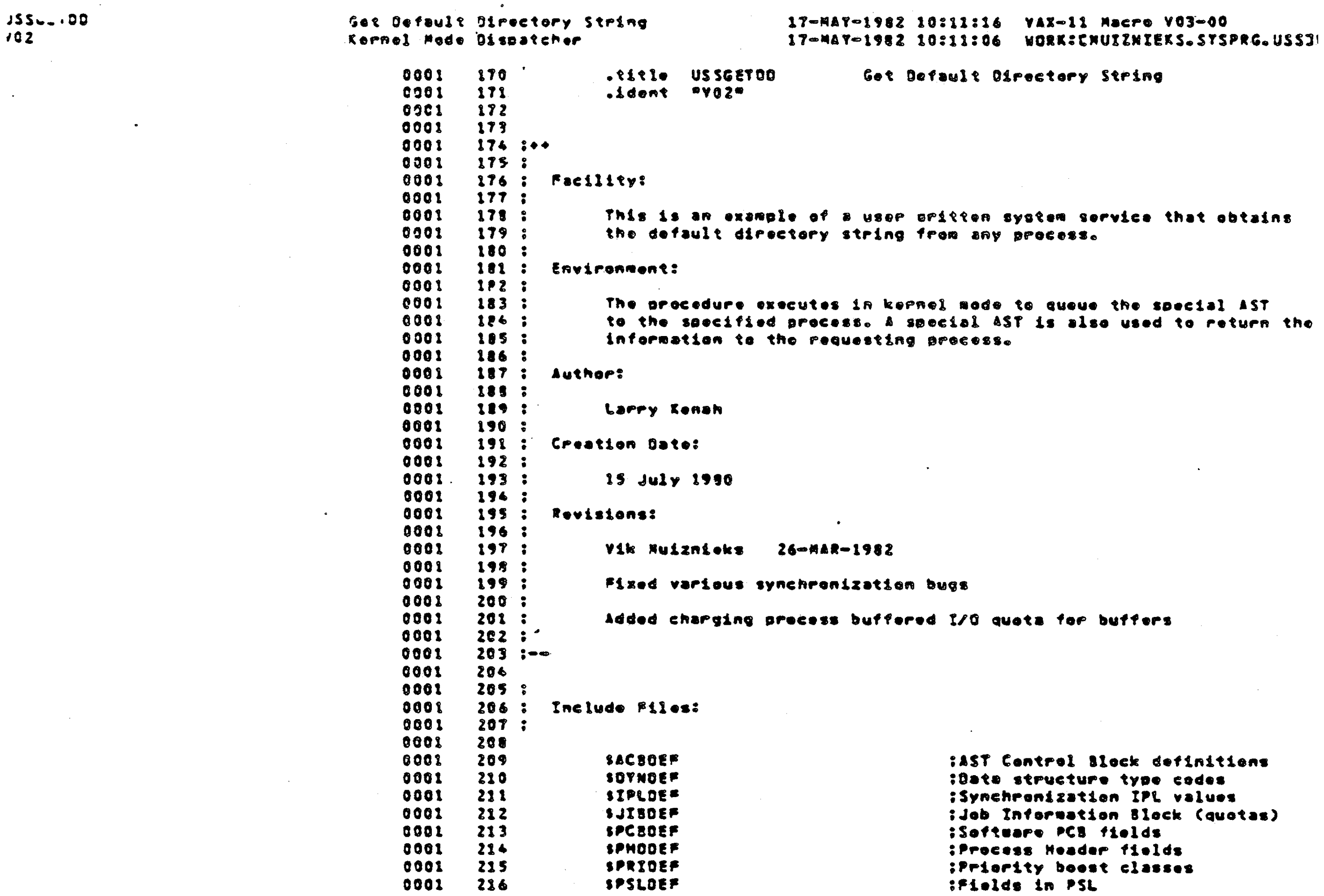
```
USSGETDD                     Get Default Directory String           17-MAY-1982 10:11:16  VAX-11 Macro V03-00
V02                          Change Mode Dispatcher Vector Block    17-MAY-1982 10:11:06  WORK:[MUIZNIEKS.SYSPRG.USS

                    0002   101          .SBTTL  Change Mode Dispatcher Vector Block
                    0002   102
                00000000   103          .PSECT  USER_SERVICES,PAGE,VEC,PIC,NOWRT,EXE
                    0000   104
           00000001 0000   105          .LONG   PLV$C_TYP_CMOD          ; Set type of vector to change mod
           00000000'0004   106          .LONG   SYS$K_VERSION           ; Identify system version
           00000005'0008   107          .LONG   KERNEL_DISPATCH-.       ; Offset to kernel mode dispatcher
           00000000 000C   108          .LONG   0                       ; Offset to executive mode dispatc
           00000000 0010   109          .LONG   0                       ; Reserved.
           00000000 0014   110          .LONG   0                       ; Reserved.
           00000000 0018   111          .LONG   0                       ; No RMS dispatcher
           00000000 001C   112          .LONG   0                       ; Address check - PIC image
```

## Listing 2-2: USSGETDD Procedure (Page 4 of 12)



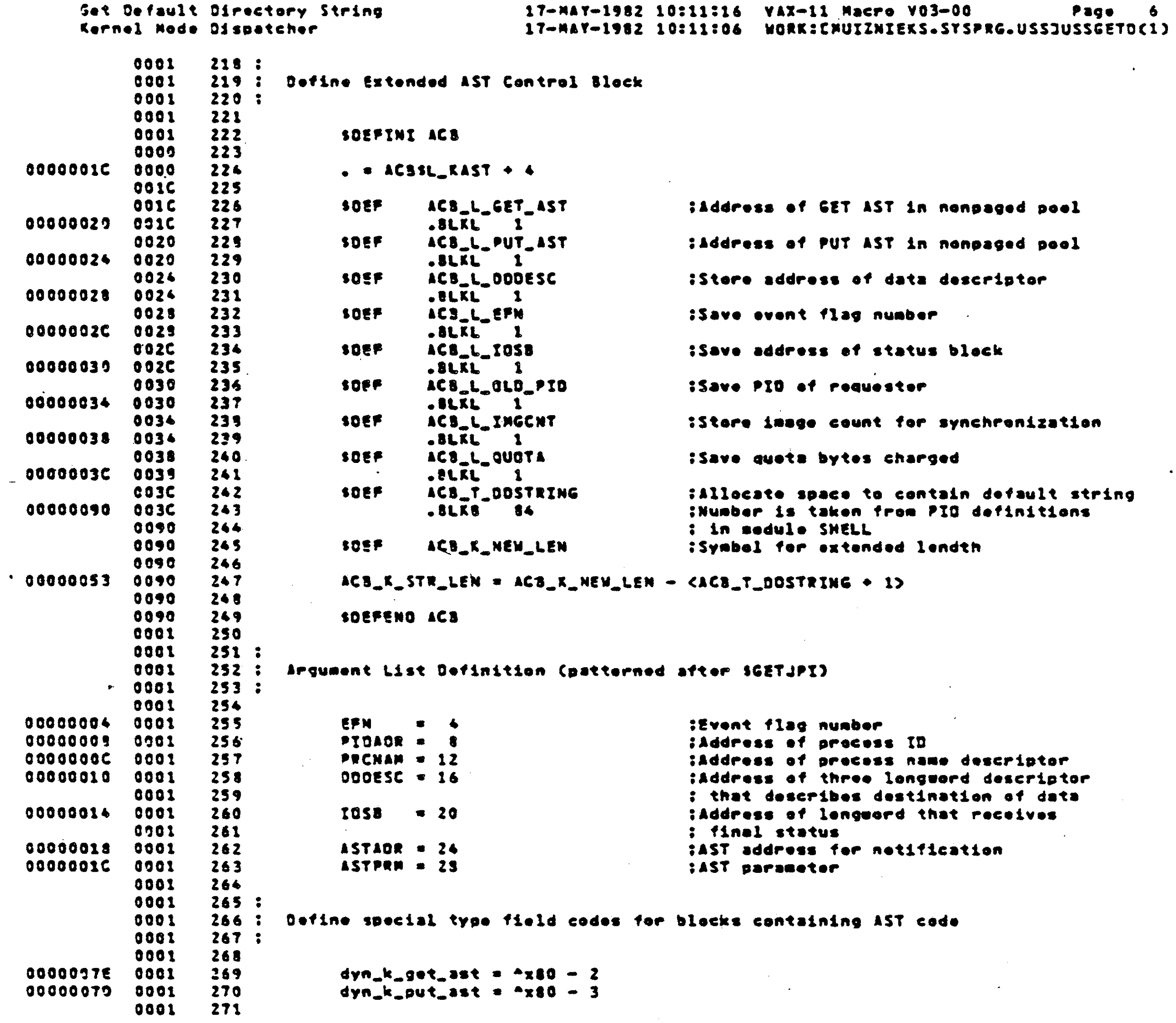
```
                      Get Default Directory String            17-MAY-1982 10:11:16  VAX-11 Macro V03-00         Page   4
                      Kernel Mode Dispatcher                  17-MAY-1982 10:11:06  WORK:[MUIZNIEKS.SYSPRG.USS]USSGETDD(1)

                                 0020    114          .SBTTL  Kernel Mode Dispatcher
                                 0020    115 ;++
                                 0020    116 ; Input Parameters:
                                 0020    117 ;
                                 0020    118 ;        (SP) - Return address if bad change mode value
                                 0020    119 ;
                                 0020    120 ;         R0  - Change mode argument value.
                                 0020    121 ;
                                 0020    122 ;         R4  - Current PCB Address. (Therefore R4 must be specified in all
                                 0020    123 ;                 register save masks for kernel routines.)
                                 0020    124 ;
                                 0020    125 ;         AP  - Argument pointer existing when the change
                                 0020    126 ;               mode instruction was executed.
                                 0020    127 ;
                                 0020    128 ;         FP  - Address of minimal call frame to exit
                                 0020    129 ;               the change mode dispatcher and return to
                                 0020    130 ;               the original mode.
                                 0020    131 ;--
                             00000000    132          .PSECT  USER_KERNEL_DISP0,BYTE,NOWRT,EXE,PIC
                                 0000    133 KACCVIO:                               ; Kernel access violation
       50    0000'8F      3C     0000    134          MOVZWL  #SS$_ACCVIO,R0          ; Set access violation status code
                          04     0005    135          RET                             ;  and return
                                 0006    136 KINSFARG:                              ; Kernel insufficient arguments.
       50    0000'8F      3C     0006    137          MOVZWL  #SS$_INSFARG,R0         ; Set status code and
                          04     000B    138          RET                             ;  return
                          05     000C    139 KNOTME: RSB                            ; RSB to forward request
                                 000D    140
                                 000D    141 KERNEL_DISPATCH::                      ; Entry to dispatcher
       51    0258 C0      9E     000D    142          MOVAB   W^-KCODE_BASE(R0),R1    ; Normalize dispatch code value
                   F8     19     0012    143          BLSS    KNOTME                  ; Branch if code value too low
             01    51     B1     0014    144          CMPW    R1,#KERNEL_COUNTER      ; Check high limit
                   F3     1E     0017    145          BGEQU   KNOTME                  ; Branch if out of range
                                 0019    146 ;
                                 0019    147 ; The dispatch code has now been verified as being handled by this dispatcher,
                                 0019    148 ; now the argument list will be probed and the required number of arguments
                                 0019    149 ; verified.
                                 0019    150 ;
    51   0000'CF41        9A     0019    151          MOVZBL  W^KERNEL_NARG[R1],R1    ; Get required argument count
51   00000004 9F41        DE     001F    152          MOVAL   @#4[R1],R1              ; Compute byte count including arg count
                                 0027    153          IFNORD  R1,(AP),KACCVIO         ; Branch if arglist not readable
  0258'CF40        6C     91     002D    154          CMPB    (AP),W^<KERNEL_NARG-KCODE_BASE>[R0] ; Check for required number
                   D1     1F     0033    155          BLSSU   KINSFARG                ;  of arguments
                   50     AF     0035    156          CASEW   R0,-                    ; Case on change mode
                                 0037    157                  -                       ; argument value
                                 0037    158                  #KCODE_BASE,-           ; Base value
       00    FDA8 8F             0037    159                  #<KERNEL_COUNTER-1>     ; Limit value (number of entries)
                                 0039    160 KCASE_BASE:                            ; Case table base address for DEFINE_SERVICE
                                 0039    161 ;
                                 0039    162 ;        Case table entries are made in the PSECT USER_KERNEL_DISP1 by
                                 0039    163 ;        invocations of the DEFINE_SERVICE macro.  The three PSECTS,
                                 0039    164 ;        USER_KERNEL_DISP0,1,2 will be abutted in lexical order at link-time.
                                 0039    165 ;
                             00000000    166          .PSECT  USER_KERNEL_DISP2,BYTE,NOWRT,EXE,PIC
                          05     0000    167          RSB                             ; Return to reject out of
                                 0001    168                                          ; range value
```

## Listing 2-2: USSGETDD Procedure (Page 5 of 12)



```
JSS..DD                         Get Default Directory String          17-MAY-1982 10:11:16  VAX-11 Macro V03-00
V02                             Kernel Mode Dispatcher                17-MAY-1982 10:11:06  WORK:[NUIZNIEKS.SYSPRG.USS3

                        0001          .title  USSGETDD        Get Default Directory String
                        0001          .ident  "V02"
                        0001
                        0001
                        0001    ;++
                        0001    ;
                        0001    ;  Facility:
                        0001    ;
                        0001    ;       This is an example of a user written system service that obtains
                        0001    ;       the default directory string from any process.
                        0001    ;
                        0001    ;  Environment:
                        0001    ;
                        0001    ;       The procedure executes in kernel mode to queue the special AST
                        0001    ;       to the specified process. A special AST is also used to return the
                        0001    ;       information to the requesting process.
                        0001    ;
                        0001    ;  Author:
                        0001    ;
                        0001    ;       Larry Kenah
                        0001    ;
                        0001    ;  Creation Date:
                        0001    ;
                        0001    ;       15 July 1980
                        0001    ;
                        0001    ;  Revisions:
                        0001    ;
                        0001    ;       Vik Muiznieks   26-MAR-1982
                        0001    ;
                        0001    ;       Fixed various synchronization bugs
                        0001    ;
                        0001    ;       Added charging process buffered I/O quota for buffers
                        0001    ;
                        0001    ;--
                        0001
                        0001    ;
                        0001    ;  Include Files:
                        0001    ;
                        0001
                        0001            $ACBDEF                         ;AST Control Block definitions
                        0001            $DYNDEF                         ;Data structure type codes
                        0001            $IPLDEF                         ;Synchronization IPL values
                        0001            $JIBDEF                         ;Job Information Block (quotas)
                        0001            $PCBDEF                         ;Software PCB fields
                        0001            $PHDDEF                         ;Process Header fields
                        0001            $PRIDEF                         ;Priority boost classes
                        0001            $PSLDEF                         ;Fields in PSL
```

## Listing 2-2: USSGETDD Procedure (Page 6 of 12)

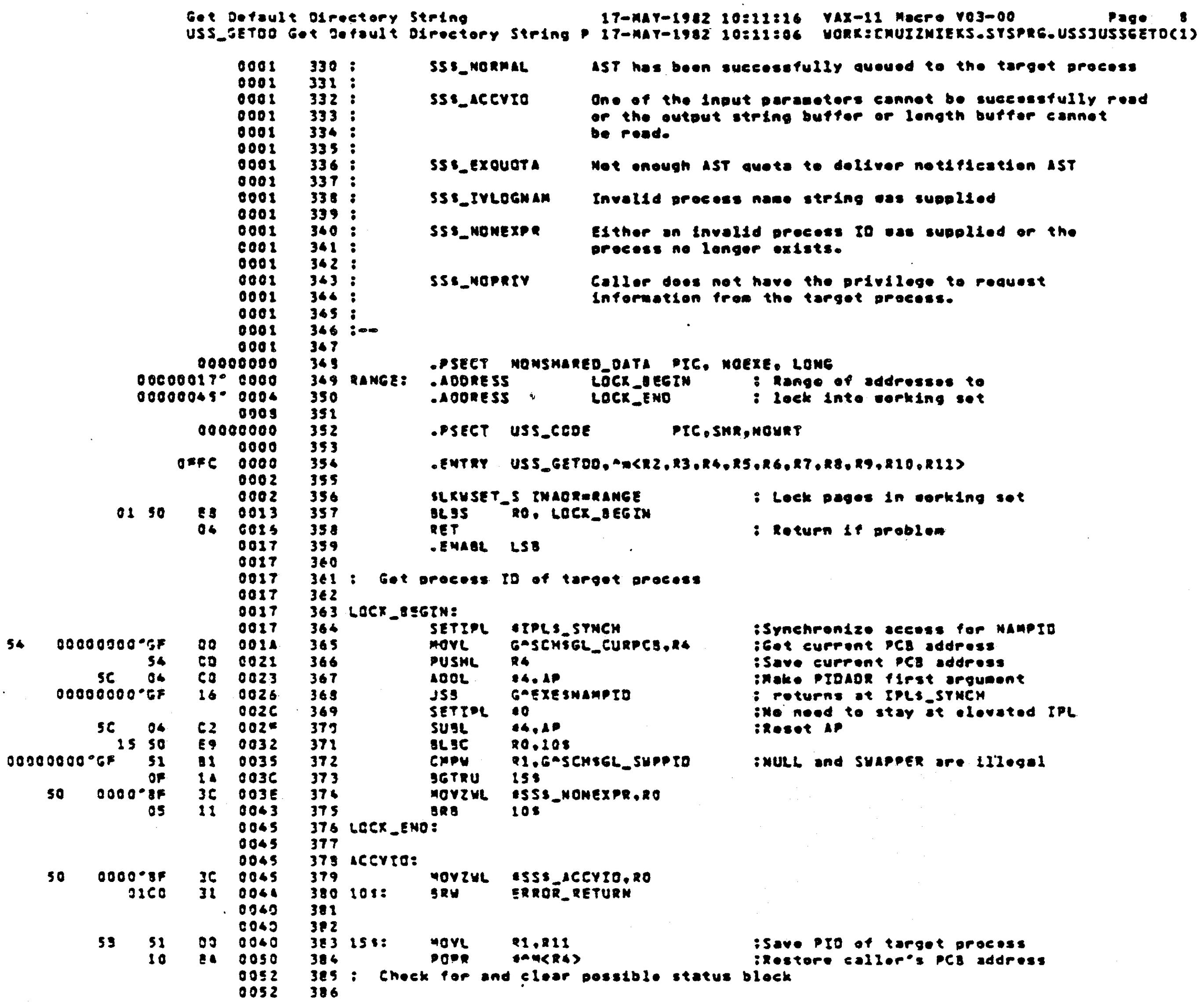


```
Get Default Directory String             17-MAY-1982 10:11:16  VAX-11 Macro V03-00          Page   6
Kernel Mode Dispatcher                   17-MAY-1982 10:11:06  WORK:[MUIZNIEKS.SYSPRG.USS]USSGETD(1)

          0001   218 ;
          0001   219 ;  Define Extended AST Control Block
          0001   220 ;
          0001   221
          0001   222         $DEFINI ACB
          0000   223
0000001C  0000   224         . = ACB$L_KAST + 4
          001C   225
          001C   226         $DEF    ACB_L_GET_AST          ;Address of GET AST in nonpaged pool
00000020  001C   227                 .BLKL   1
          0020   228         $DEF    ACB_L_PUT_AST          ;Address of PUT AST in nonpaged pool
00000024  0020   229                 .BLKL   1
          0024   230         $DEF    ACB_L_DDDESC           ;Store address of data descriptor
00000028  0024   231                 .BLKL   1
          0028   232         $DEF    ACB_L_EFN              ;Save event flag number
0000002C  0028   233                 .BLKL   1
          002C   234         $DEF    ACB_L_IOSB             ;Save address of status block
00000030  002C   235                 .BLKL   1
          0030   236         $DEF    ACB_L_OLD_PID          ;Save PID of requester
00000034  0030   237                 .BLKL   1
          0034   238         $DEF    ACB_L_IMGCNT           ;Store image count for synchronization
00000038  0034   239                 .BLKL   1
          0038   240         $DEF    ACB_L_QUOTA            ;Save quota bytes charged
0000003C  0038   241                 .BLKL   1
          003C   242         $DEF    ACB_T_DDSTRING         ;Allocate space to contain default string
00000090  003C   243                 .BLKB   84             ;Number is taken from PID definitions
          0090   244                                        ; in module SHELL
          0090   245         $DEF    ACB_K_NEW_LEN          ;Symbol for extended lendth
          0090   246
00000053  0090   247         ACB_K_STR_LEN = ACB_K_NEW_LEN - <ACB_T_DDSTRING + 1>
          0090   248
          0090   249         $DEFEND ACB
          0001   250
          0001   251 ;
          0001   252 ;  Argument List Definition (patterned after $GETJPI)
          0001   253 ;
          0001   254
00000004  0001   255         EFN    =  4                    ;Event flag number
00000008  0001   256         PIDADR =  8                    ;Address of process ID
0000000C  0001   257         PRCNAM = 12                    ;Address of process name descriptor
00000010  0001   258         DDDESC = 16                    ;Address of three longword descriptor
          0001   259                                        ; that describes destination of data
00000014  0001   260         IOSB   = 20                    ;Address of longword that receives
          0001   261                                        ; final status
00000018  0001   262         ASTADR = 24                    ;AST address for notification
0000001C  0001   263         ASTPRM = 28                    ;AST parameter
          0001   264
          0001   265 ;
          0001   266 ;  Define special type field codes for blocks containing AST code
          0001   267 ;
          0001   268
0000007E  0001   269         dyn_k_get_ast = ^x80 - 2
0000007D  0001   270         dyn_k_put_ast = ^x80 - 3
          0001   271
```

## Listing 2-2: USSGETDD Procedure (Page 7 of 12)

```
JSS_ .00                         Get Default Directory String          17-MAY-1982 10:11:16  VAX-11 Macro V03-00
/02                              USS_GETDD Get Default Directory String P 17-MAY-1982 10:11:06  WORK:[MUIZNIEKS.STSPRG.USS]

0001               .subtitle        USS_GETDD       Get Default Directory String Proce
0001
0001       :++
0001       :
0001       :  Functional Description:
0001       :
0001       :       This procedure obtains the default directory string for any proces
0001       :       in the system. The method used parallels the $GETJPI system servic
0001       :       A special kernel AST is delivered to the target process, where the
0001       :       default directory string is copied from its P1 space location to
0001       :       the extended AST control block. That block is then used to deliver
0001       :       another AST back to the requesting process.
0001       :
0001       :  Input Parameters:
0001       :
0001       :       EFN(AP)          Number of event flag to set when the requested
0001       :                        information is available.
0001       :
0001       :       PIDADR(AP)       Address of longword containing the process ID of t
0001       :                        process for which the information is being request
0001       :
0001       :       PRCNAM(AP)       Address of the string descriptor for the process n
0001       :                        of the process for which the information is being
0001       :                        requested.
0001       :
0001       :       DDDESC(AP)       Address of three longword descriptor that describe
0001       :                        where information will be stored.
0001       :
0001       :                        +---------------+---------------+
0001       :                        |     spare     | Buffer Length |
0001       :                        +---------------+---------------+
0001       :                        |        Buffer Address         |
0001       :                        +-------------------------------+
0001       :                        |   Address to Return Length    |
0001       :                        +-------------------------------+
0001       :
0001       :       IOSB(AP)         Used by the kernel AST to report errors back to
0001       :                        the original caller that cannot be detected in
0001       :                        the initial procedure. One such error might be
0001       :                        a protection change in the user's buffer.
0001       :
0001       :       ASTADR(AP)       Address of an AST that will be called when all of
0001       :                        the requested data has been supplied.
0001       :
0001       :       ASTPRM(AP)       Parameter that will be passed to that AST
0001       :
0001       :  Implicit Input:
0001       :
0001       :       R4      Address of PCB of caller (current process)
0001       :
0001       :  Output Parameters:
0001       :
0001       :       The default string (and optionally its length) are passed
0001       :       back to the caller.
0001       :
0001       :  Return Status:
0001       :
```

## Listing 2-2: USSGETDD Procedure (Page 8 of 12)

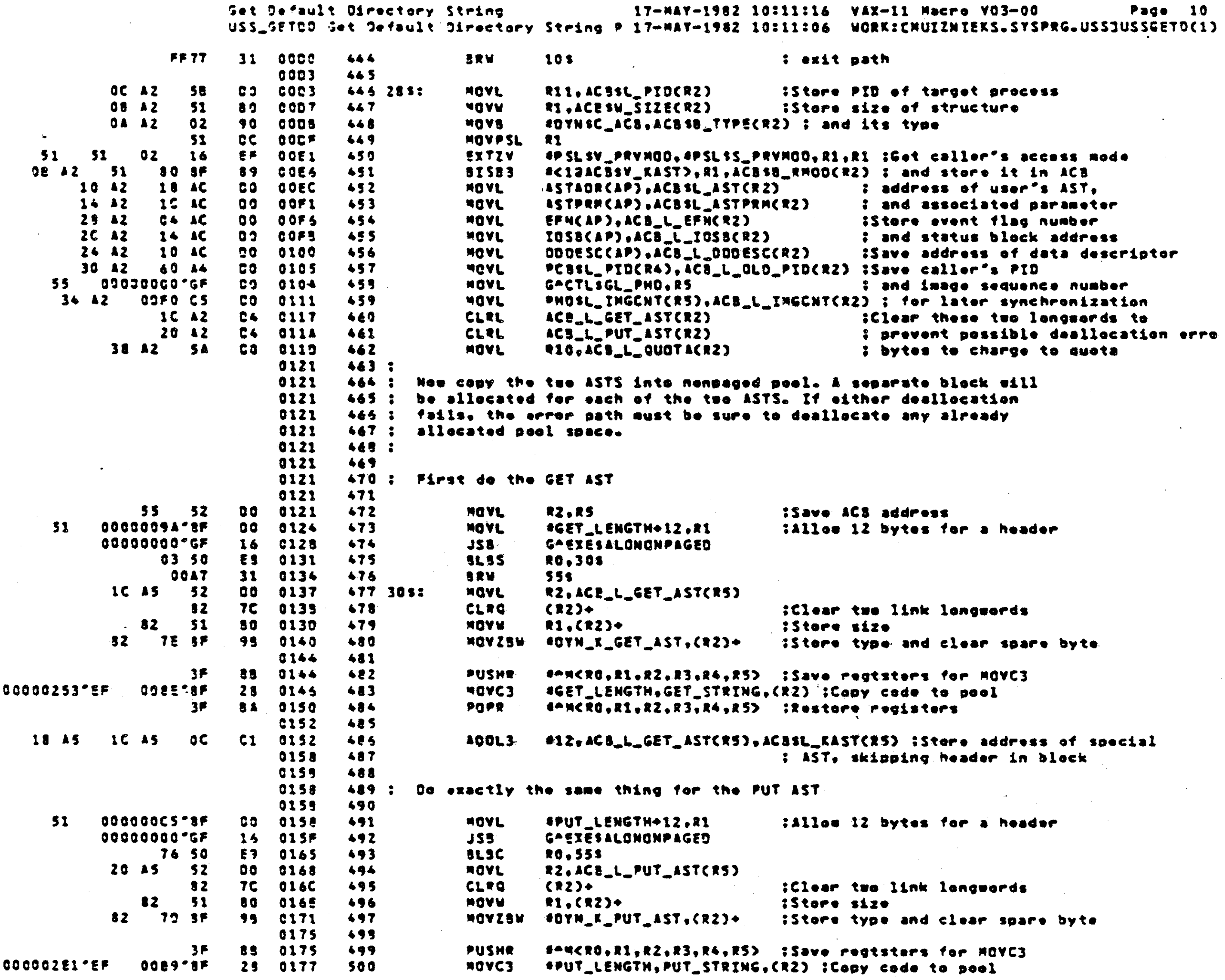


```
                    Get Default Directory String            17-MAY-1982 10:11:16  VAX-11 Macro V03-00          Page   8
                    USS_GETDD Get Default Directory String P 17-MAY-1982 10:11:06  WORK:CMUIZNIEKS.SYSPRG.USS]USSGETD(1)

                              0001    330 :          SS$_NORMAL        AST has been successfully queued to the target process
                              0001    331 :
                              0001    332 :          SS$_ACCVIO        One of the input parameters cannot be successfully read
                              0001    333 :                            or the output string buffer or length buffer cannot
                              0001    334 :                            be read.
                              0001    335 :
                              0001    336 :          SS$_EXQUOTA       Not enough AST quota to deliver notification AST
                              0001    337 :
                              0001    338 :          SS$_IVLOGNAM      Invalid process name string was supplied
                              0001    339 :
                              0001    340 :          SS$_NONEXPR       Either an invalid process ID was supplied or the
                              0001    341 :                            process no longer exists.
                              0001    342 :
                              0001    343 :          SS$_NOPRIV        Caller does not have the privilege to request
                              0001    344 :                            information from the target process.
                              0001    345 :
                              0001    346 :--
                              0001    347
                          00000000    348          .PSECT  NONSHARED_DATA  PIC, NOEXE, LONG
                00000017' 0000    349 RANGE:   .ADDRESS        LOCK_BEGIN       : Range of addresses to
                00000045' 0004    350          .ADDRESS        LOCK_END         : lock into working set
                              0008    351
                          00000000    352          .PSECT  USS_CODE        PIC,SHR,NOWRT
                              0000    353
                        0FFC  0000    354          .ENTRY  USS_GETDD,^m<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                              0002    355
                              0002    356          $LKWSET_S INADR=RANGE            : Lock pages in working set
               01 50     E8   0013    357          BLBS    R0, LOCK_BEGIN
                         04   0016    358          RET                              : Return if problem
                              0017    359          .ENABL  LSB
                              0017    360
                              0017    361 :  Get process ID of target process
                              0017    362
                              0017    363 LOCK_BEGIN:
                              0017    364          SETIPL  #IPL$_SYNCH              :Synchronize access for NAMPID
      54     00000000'GF   DO   001A    365          MOVL    G^SCH$GL_CURPCB,R4       :Get current PCB address
                      54   DD   0021    366          PUSHL   R4                       :Save current PCB address
               5C     04   C0   0023    367          ADDL    #4,AP                    :Make PIDADR first argument
             00000000'GF   16   0026    368          JSB     G^EXE$NAMPID             : returns at IPL$_SYNCH
                              002C    369          SETIPL  #0                       :No need to stay at elevated IPL
               5C     04   C2   002F    370          SUBL    #4,AP                    :Reset AP
                   15 50   E9   0032    371          BLBC    R0,10$
      00000000'GF     51   B1   0035    372          CMPW    R1,G^SCH$GL_SWPPID       :NULL and SWAPPER are illegal
                      0F   1A   003C    373          BGTRU   15$
          50    0000'8F    3C   003E    374          MOVZWL  #SS$_NONEXPR,R0
                      05   11   0043    375          BRB     10$
                              0045    376 LOCK_END:
                              0045    377
                              0045    378 ACCVIO:
          50    0000'8F    3C   0045    379          MOVZWL  #SS$_ACCVIO,R0
                    01C0   31   004A    380 10$:     BRW     ERROR_RETURN
                              004D    381
                              004D    382
               5B     51   D0   004D    383 15$:     MOVL    R1,R11                   :Save PID of target process
                      10   BA   0050    384          POPR    #^M<R4>                  :Restore caller's PCB address
                              0052    385 :  Check for and clear possible status block
                              0052    386
```

## Listing 2-2: USSGETDD Procedure (Page 9 of 12)

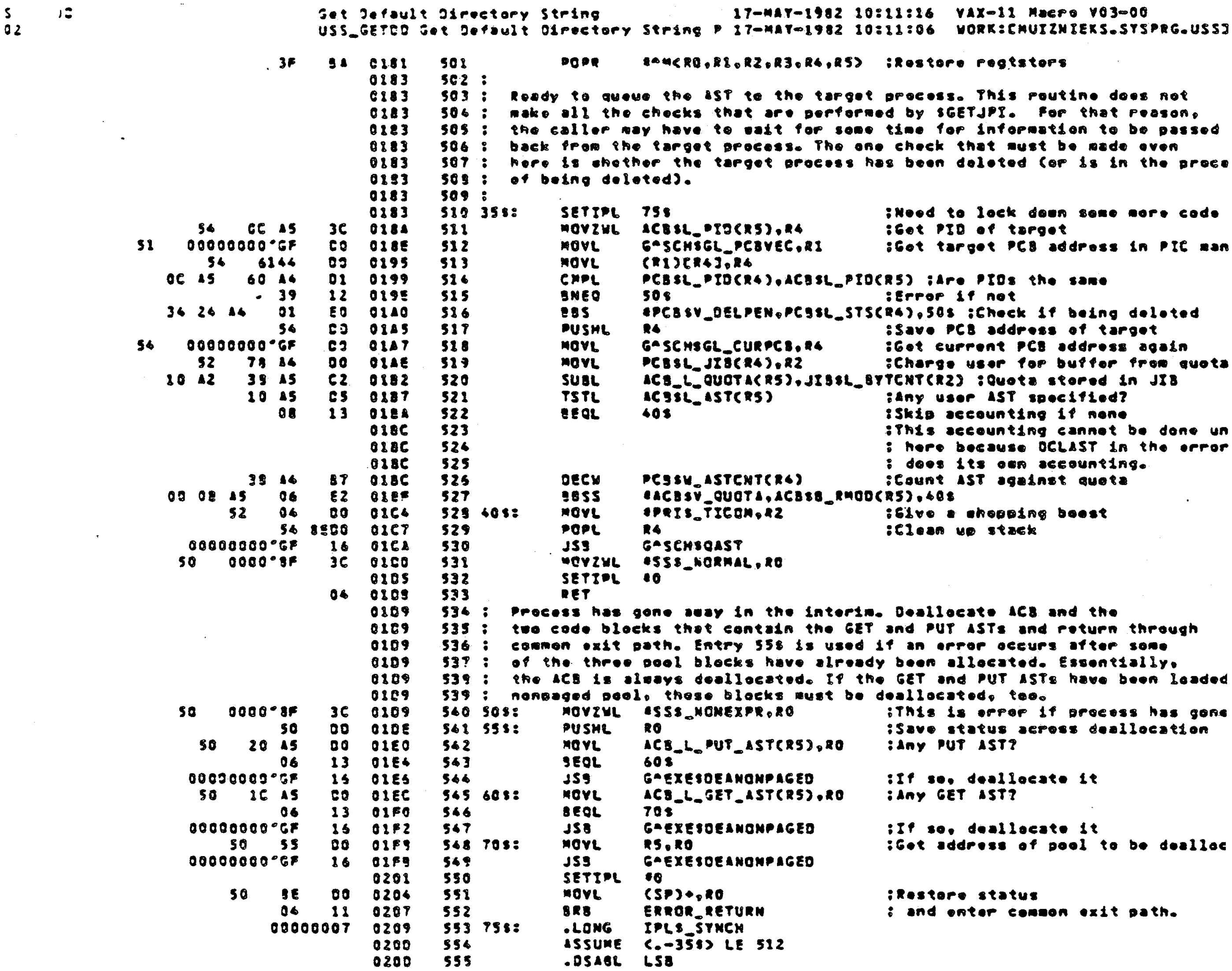
```
JS     JD                              Get Default Directory String          17-MAY-1982 10:11:16  VAX-11 Macro V03-00
V02                                    USS_GETDD Get Default Directory String P 17-MAY-1982 10:11:06  WORK:CMUIZNIEKS.SYSPRG.U

              51    14 AC     C0  0052            MOVL     IOSB(AP),R1             ;Get IOSB address
                       08     13  0056            BEQL     20$                     ;Skip if none
                                  0058            IFNOWRT  #4,(R1),ACCVIO          ;Check accessibility
                       61     D4  005E            CLRL     (R1)                    ;Clear it initially
                                  0060
                                  0060      ;  Clear event flag
                                  0060
              53    04 AC     9A  0060    20$:    MOVZBL   EFN(AP),R3              ;Get event flag number
            00000000'GF       16  0064            JSB      G^SCH$CLREF             ;Clear that flag
                    D0 50     E9  006A            BLBC     R0,10$                  ;Exit if errors
                                  006D
                                  006D      ;  Check for enough AST quota if ASTADR argument present
                                  006D
                    18 AC     D5  006D            TSTL     ASTADR(AP)              ;Argument specified
                       0A     13  0070            BEQL     25$                     ;Skip check if not
           50     0000'8F     3C  0072            MOVZWL   #SS$_EXQUOTA,R0         ;Assume not enough quota
                    38 A4     B5  0077            TSTW     PCB$W_ASTCNT(R4)        ;Any quota left
                       CE     1B  007A            BLEQU    10$                     ;Error if none
                                  007C
                                  007C      ;  Check accessibility of data descriptor
                                  007C
              55    10 AC     D0  007C    25$:    MOVL     DDDESC(AP),R5           ;Get address of descriptor
                                  0080            IFNORD   #12,(R5),ACCVIO         ;Is descriptor readable?
                 56    65     3C  0086            MOVZWL   (R5),R6                 ;Buffer size to R6
              57    04 A5     D0  0089            MOVL     4(R5),R7                ; and address to R7
                                  008D            IFNOWRT  R6,(R7),ACCVIO          ;Is text buffer writeable?
              58    08 A5     D0  0093            MOVL     8(R5),R8                ;Get address of length buffer
                                  0097            IFNOWRT  #4,(R8),ACCVIO          ;Is it writeable?
                                  009D
                                  009D      ;  Check for sufficient quota by summing sizes of all needed blocks
                                  009D
          51     00000090 8F  D0  009D            MOVL     #ACB_K_NEW_LEN,R1       ; AST block length plus    P-2
     51   51     000003C5'8F  C1  00A4            ADDL3    #<PUT_LENGTH+12>,R1,R1  ; put string length plus   P 2.
     51   51     0000009A'8F  C1  00AC            ADDL3    #<GET_LENGTH+12>,R1,R1  ; get string length
                 5A    51     D0  00B4            MOVL     R1,R10                  ; save quota bytes to be charge
            00000000'GF       16  00B7            JSB      G^EXE$BUFFRQUOTA        ; Check quota
                    8A 50     E9  00BD            BLBC     R0, 10$
                                  00C0
                                  00C0      ;  At this point, all checks have been made. The access checks must sti
                                  00C0      ;  be made again when it is time to move data to the user's buffer.
                                  00C0      ;  The asynchronous nature of this service allows the calling process
                                  00C0      ;  to continue execution while the default directory string is being
                                  00C0      ;  obtained. Protection could be changed on the buffer, causing a
                                  00C0      ;  possible access violation from kernel mode.
                                  00C0      ;
                                  00C0      ;  One optimization that is possible here is to check whether the
                                  00C0      ;  target process is the same as the caller. The default directory
                                  00C0      ;  can be obtained in a much more straightforward manner than is being
                                  00C0      ;  done here. In fact, an RMS call already exists to accomplish this.
                                  00C0      ;
                                  00C0      ;  Now allocate an extended AST control block and store the
                                  00C0      ;  relevant parameters.
                                  00C0
          51     00000090 8F  D0  00C0            MOVL     #ACB_K_NEW_LEN,R1       ;Set size of extended ACB
            00000000'GF       16  00C7            JSB      G^EXE$ALLOCBUF          ;Allocate nonpaged pool space
                                  00CD                                             ;We are at IPL 2 now
                    C3 50     E9  00CD            BLBS     R0,28$                  ;Return error status through co
```

## Listing 2-2: USSGETDD Procedure (Page 1Ø of 12)



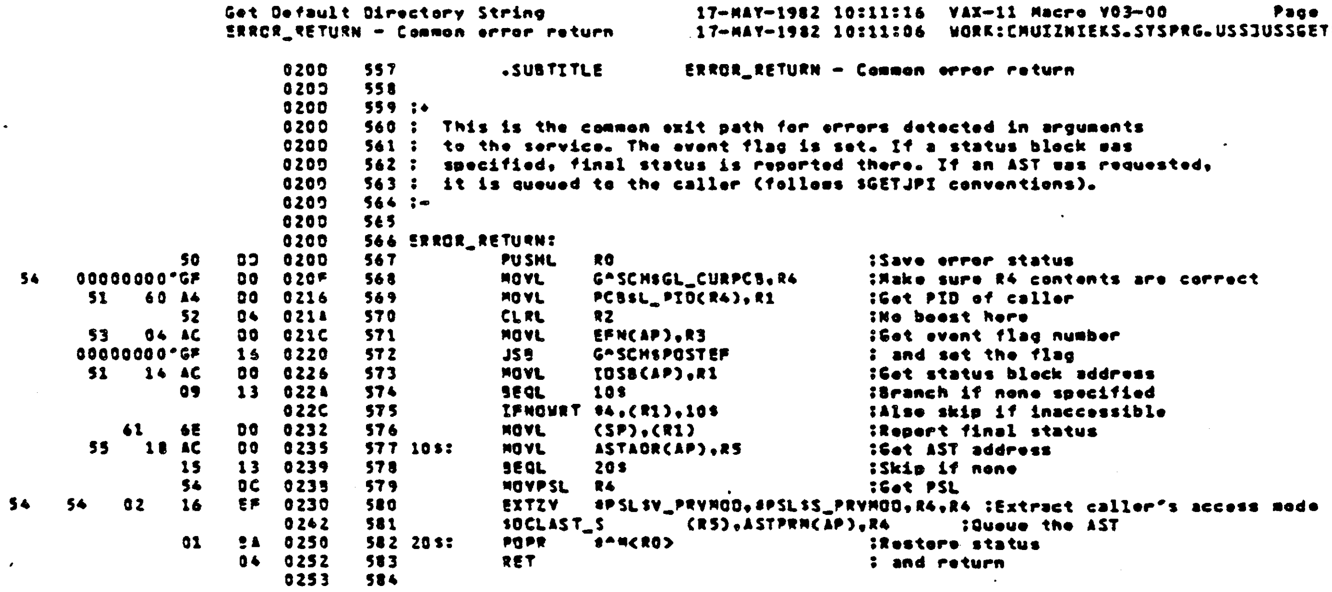
```
                              Get Default Directory String            17-MAY-1982 10:11:16  VAX-11 Macro V03-00         Page  10
                              USS_GETDD Get Default Directory String P 17-MAY-1982 10:11:06  WORK:[NUIZNIEKS.SYSPRG.USS]USSGETD(1)

               FF77   31  0000    444         BRW     10$                     ; exit path
                          0003    445
        0C A2    5B   D0  0003    446 28$:    MOVL    R11,ACB$L_PID(R2)       ;Store PID of target process
        08 A2    51   B0  0007    447         MOVW    R1,ACB$W_SIZE(R2)       ;Store size of structure
        0A A2    02   90  000B    448         MOVB    #DYN$C_ACB,ACB$B_TYPE(R2) ; and its type
                 51   DC  000F    449         MOVPSL  R1
 51    51    02   16   EF  00E1    450         EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R1,R1 ;Get caller's access mode
08 A2   51    80 8F   89  00E6    451         BISB3   #<1@ACB$V_KAST>,R1,ACB$B_RMOD(R2) ; and store it in ACB
    10 A2    18 AC   D0  00EC    452         MOVL    ASTADR(AP),ACB$L_AST(R2)       ; address of user's AST,
    14 A2    1C AC   D0  00F1    453         MOVL    ASTPRM(AP),ACB$L_ASTPRM(R2)    ; and associated parameter
    28 A2    04 AC   D0  00F6    454         MOVL    EFN(AP),ACB_L_EFN(R2)          ;Store event flag number
    2C A2    14 AC   D0  00FB    455         MOVL    IOSB(AP),ACB_L_IOSB(R2)        ; and status block address
    24 A2    10 AC   D0  0100    456         MOVL    DDDESC(AP),ACB_L_DDDESC(R2)    ;Save address of data descriptor
    30 A2    60 A4   D0  0105    457         MOVL    PCB$L_PID(R4),ACB_L_OLD_PID(R2) ;Save caller's PID
  55   00000000'GF   D0  010A    458         MOVL    G^CTL$GL_PHD,R5                ; and image sequence number
 34 A2    00F0 C5    D0  0111    459         MOVL    PHD$L_IMGCNT(R5),ACB_L_IMGCNT(R2) ; for later synchronization
          1C A2      D4  0117    460         CLRL    ACB_L_GET_AST(R2)              ;Clear these two longwords to
          20 A2      D4  011A    461         CLRL    ACB_L_PUT_AST(R2)              ; prevent possible deallocation erro
      38 A2    5A    D0  011D    462         MOVL    R10,ACB_L_QUOTA(R2)            ; bytes to charge to quota
                          0121    463 ;
                          0121    464 ;  Now copy the two ASTS into nonpaged pool. A separate block will
                          0121    465 ;  be allocated for each of the two ASTS. If either deallocation
                          0121    466 ;  fails, the error path must be sure to deallocate any already
                          0121    467 ;  allocated pool space.
                          0121    468 ;
                          0121    469
                          0121    470 ;  First do the GET AST
                          0121    471
          55    52   D0  0121    472         MOVL    R2,R5                   ;Save ACB address
  51   0000009A'8F   D0  0124    473         MOVL    #GET_LENGTH+12,R1       ;Allow 12 bytes for a header
       00000000'GF   16  0128    474         JSB     G^EXE$ALONONPAGED
             03 50   E9  0131    475         BLBS    R0,30$
              00A7   31  0134    476         BRW     55$
       1C A5    52   D0  0137    477 30$:    MOVL    R2,ACB_L_GET_AST(R5)
                82   7C  013B    478         CLRQ    (R2)+                   ;Clear two link longwords
          82    51   B0  013D    479         MOVW    R1,(R2)+                ;Store size
     82    7E 8F     9B  0140    480         MOVZBW  #DYN_K_GET_AST,(R2)+    ;Store type and clear spare byte
                          0144    481
                3F   BB  0144    482         PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ;Save registers for MOVC3
00000253'EF   008E'8F    28  0146    483         MOVC3   #GET_LENGTH,GET_STRING,(R2) ;Copy code to pool
                3F   BA  0150    484         POPR    #^M<R0,R1,R2,R3,R4,R5>  ;Restore registers
                          0152    485
  18 A5   1C A5    0C   C1  0152    486         ADDL3   #12,ACB_L_GET_AST(R5),ACB$L_KAST(R5) ;Store address of special
                          0158    487                                         ; AST, skipping header in block
                          0158    488
                          0158    489 ;  Do exactly the same thing for the PUT AST
                          0158    490
  51   000000C5'8F   D0  0158    491         MOVL    #PUT_LENGTH+12,R1       ;Allow 12 bytes for a header
       00000000'GF   16  015F    492         JSB     G^EXE$ALONONPAGED
             76 50   E9  0165    493         BLBC    R0,55$
       20 A5    52   D0  0168    494         MOVL    R2,ACB_L_PUT_AST(R5)
                82   7C  016C    495         CLRQ    (R2)+                   ;Clear two link longwords
          82    51   B0  016E    496         MOVW    R1,(R2)+                ;Store size
     82    7D 8F     9B  0171    497         MOVZBW  #DYN_K_PUT_AST,(R2)+    ;Store type and clear spare byte
                          0175    498
                3F   BB  0175    499         PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ;Save registers for MOVC3
000002E1'EF   0089'8F    28  0177    500         MOVC3   #PUT_LENGTH,PUT_STRING,(R2) ;Copy code to pool
```

## Listing 2-2: USSGETDD Procedure (Page 11 of 12)



```
S     )C                      Get Default Directory String            17-MAY-1982 10:11:16  VAX-11 Macro V03-00
02                            USS_GETDD Get Default Directory String P 17-MAY-1982 10:11:06  WORK:[MUIZNIEKS.SYSPRG.USS]

                  3F    BA  0181   501          POPR    #^M<R0,R1,R2,R3,R4,R5>  ;Restore regtstors
                            0183   502 :
                            0183   503 :  Ready to queue the AST to the target process. This routine does not
                            0183   504 :  make all the checks that are performed by $GETJPI.  For that reason,
                            0183   505 :  the caller may have to wait for some time for information to be passed
                            0183   506 :  back from the target process. The one check that must be made even
                            0183   507 :  here is whether the target process has been deleted (or is in the proce
                            0183   508 :  of being deleted).
                            0183   509 :
                            0183   510 35$:     SETIPL  75$                         ;Need to lock down some more code
            54    0C A5  3C 018A   511          MOVZWL  ACB$L_PID(R5),R4            ;Get PID of target
     51    00000000'GF   C0 018E   512          MOVL    G^SCH$GL_PCBVEC,R1          ;Get target PCB address in PIC man
            54    6144   D0 0195   513          MOVL    (R1)[R4],R4
      0C A5    60 A4     D1 0199   514          CMPL    PCB$L_PID(R4),ACB$L_PID(R5) ;Are PIDs the same
                   39    12 019E   515          BNEQ    50$                         ;Error if not
    34 24 A4    01       E0 01A0   516          BBS     #PCB$V_DELPEN,PCB$L_STS(R4),50$ ;Check if being deleted
                   54    DD 01A5   517          PUSHL   R4                          ;Save PCB address of target
     54    00000000'GF   D0 01A7   518          MOVL    G^SCH$GL_CURPCB,R4          ;Get current PCB address again
            52    78 A4  D0 01AE   519          MOVL    PCB$L_JIB(R4),R2            ;Charge user for buffer from quota
      10 A2    38 A5     C2 01B2   520          SUBL    ACB_L_QUOTA(R5),JIB$L_BYTCNT(R2) ;Quota stored in JIB
                10 A5    D5 01B7   521          TSTL    ACB$L_AST(R5)               ;Any user AST specified?
                   08    13 01BA   522          BEQL    40$                         ;Skip accounting if none
                            01BC   523                                              ;This accounting cannot be done un
                            01BC   524                                              ; here because DCLAST in the error
                            01BC   525                                              ; does its own accounting.
                38 A4    B7 01BC   526          DECW    PCB$W_ASTCNT(R4)            ;Count AST against quota
    03 08 A5    06       E2 01BF   527          BBSS    #ACB$V_QUOTA,ACB$B_RMOD(R5),40$
            52    04     D0 01C4   528 40$:     MOVL    #PRI$_TICOM,R2              ;Give a whopping boost
                   54 8ED0  01C7   529          POPL    R4                          ;Clean up stack
         00000000'GF     16 01CA   530          JSB     G^SCH$QAST
     50    0000'8F       3C 01CD   531          MOVZWL  #SS$_NORMAL,R0
                            01D5   532          SETIPL  #0
                         04 01D8   533          RET
                            01D9   534 :  Process has gone away in the interim. Deallocate ACB and the
                            01D9   535 :  two code blocks that contain the GET and PUT ASTs and return through
                            01D9   536 :  common exit path. Entry 55$ is used if an error occurs after some
                            01D9   537 :  of the three pool blocks have already been allocated. Essentially,
                            01D9   538 :  the ACB is always deallocated. If the GET and PUT ASTs have been loaded
                            01D9   539 :  nonpaged pool, these blocks must be deallocated, too.
     50    0000'8F       3C 01D9   540 50$:     MOVZWL  #SS$_NONEXPR,R0             ;This is error if process has gone
                   50    DD 01DE   541 55$:     PUSHL   R0                          ;Save status across deallocation
      50    20 A5        D0 01E0   542          MOVL    ACB_L_PUT_AST(R5),R0        ;Any PUT AST?
                   06    13 01E4   543          BEQL    60$
         00000000'GF     16 01E6   544          JSB     G^EXE$DEANONPAGED           ;If so, deallocate it
      50    1C A5        D0 01EC   545 60$:     MOVL    ACB_L_GET_AST(R5),R0        ;Any GET AST?
                   06    13 01F0   546          BEQL    70$
         00000000'GF     16 01F2   547          JSB     G^EXE$DEANONPAGED           ;If so, deallocate it
         50    55        D0 01F8   548 70$:     MOVL    R5,R0                       ;Get address of pool to be dealloc
         00000000'GF     16 01FB   549          JSB     G^EXE$DEANONPAGED
                            0201   550          SETIPL  #0
         50    8E        D0 0204   551          MOVL    (SP)+,R0                    ;Restore status
                   04    11 0207   552          BRB     ERROR_RETURN                ; and enter common exit path.
               00000007     0209   553 75$:     .LONG   IPL$_SYNCH
                            020D   554          ASSUME  <.-35$> LE 512
                            020D   555          .DSABL  LSB
```

## Listing 2-2: USSGETDD Procedure (Page 12 of 12)

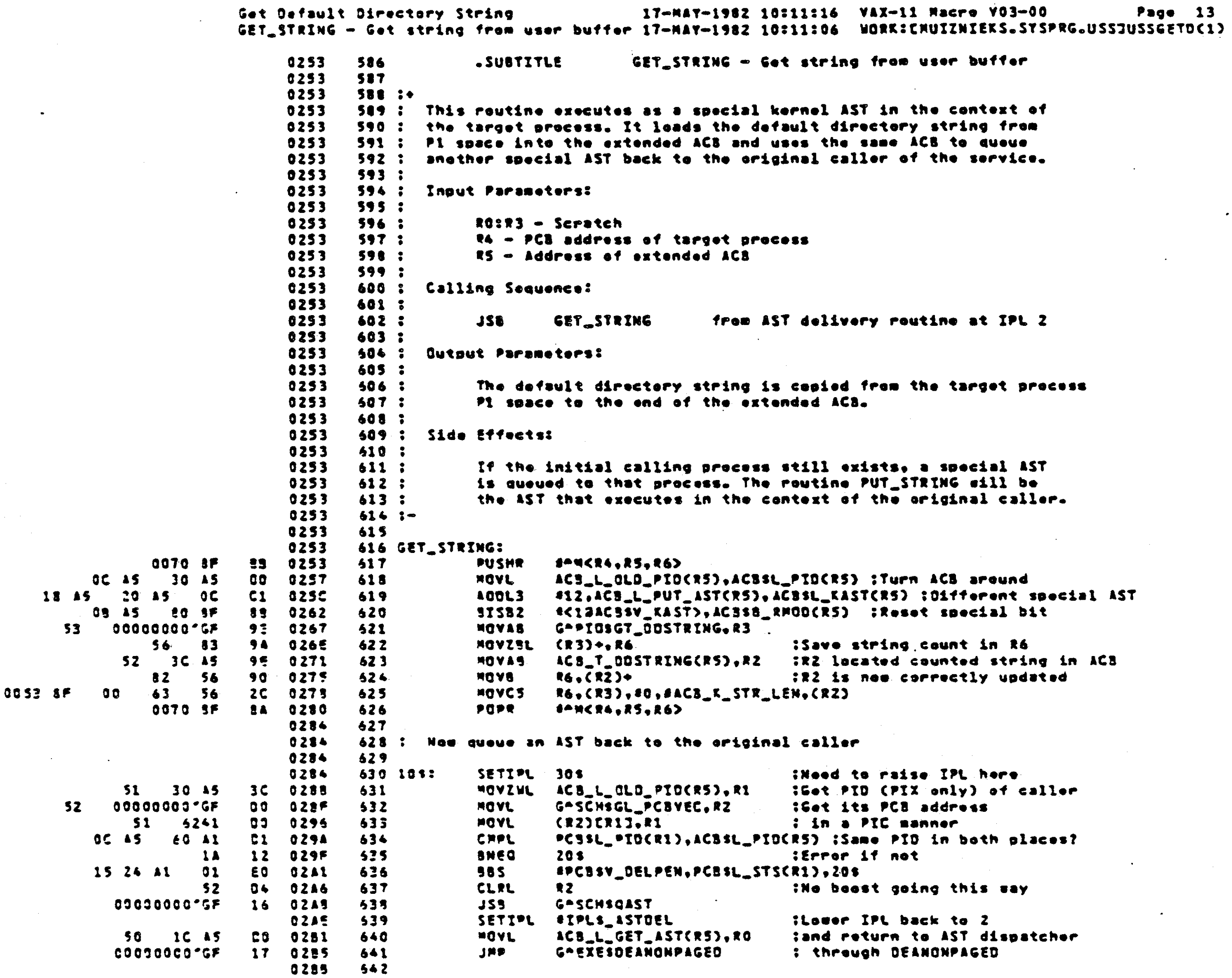


```
                                Get Default Directory String          17-MAY-1982 10:11:16  VAX-11 Macro V03-00           Page  12
                                ERROR_RETURN - Common error return     17-MAY-1982 10:11:06  WORK:[MUIZNIEKS.SYSPRG.USS]USSGETD(1)

                                     020D   557          .SUBTITLE       ERROR_RETURN - Common error return
                                     020D   558
                                     020D   559 ;+
                                     020D   560 ;  This is the common exit path for errors detected in arguments
                                     020D   561 ;  to the service. The event flag is set. If a status block was
                                     020D   562 ;  specified, final status is reported there. If an AST was requested,
                                     020D   563 ;  it is queued to the caller (follows $GETJPI conventions).
                                     020D   564 ;-
                                     020D   565
                                     020D   566 ERROR_RETURN:
                        50    DD     020D   567          PUSHL    R0                          ;Save error status
  54    00000000'GF       D0     020F   568          MOVL     G^SCH$GL_CURPCB,R4          ;Make sure R4 contents are correct
        51    60 A4       D0     0216   569          MOVL     PCB$L_PID(R4),R1            ;Get PID of caller
                        52    D4     021A   570          CLRL     R2                          ;No boost here
        53    04 AC       D0     021C   571          MOVL     EFN(AP),R3                  ;Get event flag number
          00000000'GF     16     0220   572          JSB      G^SCH$POSTEF                ; and set the flag
        51    14 AC       D0     0226   573          MOVL     IOSB(AP),R1                 ;Get status block address
                        09    13     022A   574          BEQL     10$                         ;Branch if none specified
                                     022C   575          IFNOWRT  #4,(R1),10$                 ;Also skip if inaccessible
              61    6E    D0     0232   576          MOVL     (SP),(R1)                   ;Report final status
        55    18 AC       D0     0235   577 10$:     MOVL     ASTADR(AP),R5               ;Get AST address
                        15    13     0239   578          BEQL     20$                         ;Skip if none
                        54    DC     023B   579          MOVPSL   R4                          ;Get PSL
  54    54    02    16    EF     023D   580          EXTZV    #PSL$V_PRVMOD,#PSL$S_PRVMOD,R4,R4 ;Extract caller's access mode
                                     0242   581          $DCLAST_S      (R5),ASTPRM(AP),R4        ;Queue the AST
                        01    BA     0250   582 20$:     POPR     #^M<R0>                     ;Restore status
                              04     0252   583          RET                                  ; and return
                                     0253   584
```

## The GET String AST Procedure

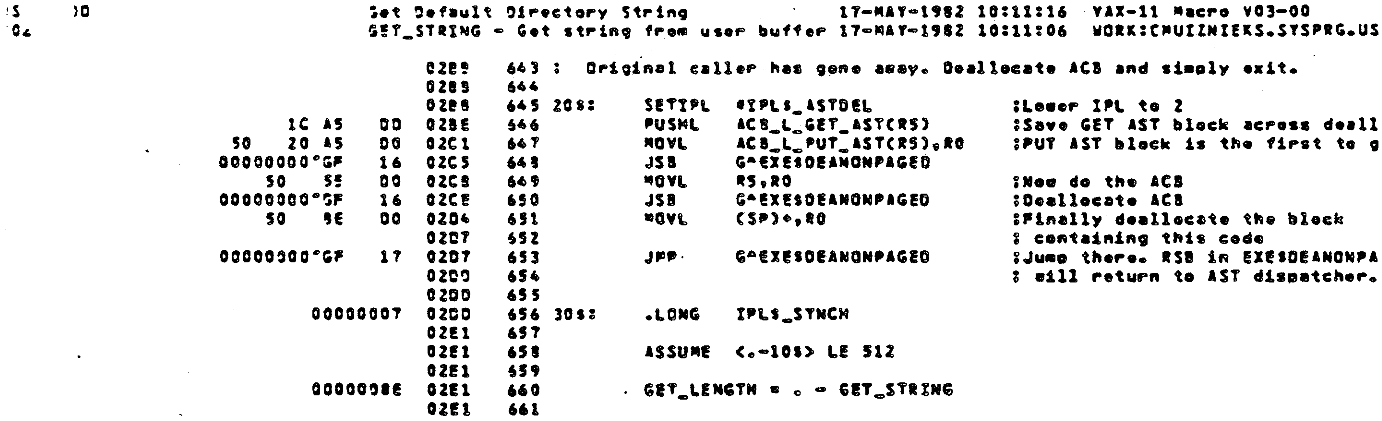
- Executes in context of target process
  - At IPL 2, since special kernel AST
  - Entered via JSB, not CALL, from AST delivery interrupt service routine
- Reformats initial ACB fields
  - So can use same ACB to queue PUT string AST back to caller
  - Specifies AST should be another special kernel AST
- Loads default directory string into ACB
  - No protection checks are necessary to guard against access violations in kernel mode since:
    - ACB in nonpaged pool
    - Default directory string in protected area of P1 space
- Checks that caller process still exists
  - Raises IPL to SYNCH
- Queues AST back to caller
  - So PUT string procedure can execute
- Deallocates GET string code block
  - By JMP to deallocate nonpaged pool routine
  - That routine exits with RSB, which returns control to AST delivery interrupt service routine

## Listing 2-3: GET String AST Procedure (Page 1 of 2)



```
                              Get Default Directory String          17-MAY-1982 10:11:16  VAX-11 Macro V03-00         Page  13
                              GET_STRING - Get string from user buffer 17-MAY-1982 10:11:06  WORK:[MUIZNIEKS.SYSPRG.USS]USSGETD(1)

                                     0253   586         .SUBTITLE       GET_STRING - Get string from user buffer
                                     0253   587
                                     0253   588 ;+
                                     0253   589 ;  This routine executes as a special kernel AST in the context of
                                     0253   590 ;  the target process. It loads the default directory string from
                                     0253   591 ;  P1 space into the extended ACB and uses the same ACB to queue
                                     0253   592 ;  another special AST back to the original caller of the service.
                                     0253   593 ;
                                     0253   594 ;  Input Parameters:
                                     0253   595 ;
                                     0253   596 ;       R0:R3 - Scratch
                                     0253   597 ;       R4 - PCB address of target process
                                     0253   598 ;       R5 - Address of extended ACB
                                     0253   599 ;
                                     0253   600 ;  Calling Sequence:
                                     0253   601 ;
                                     0253   602 ;       JSB     GET_STRING      from AST delivery routine at IPL 2
                                     0253   603 ;
                                     0253   604 ;  Output Parameters:
                                     0253   605 ;
                                     0253   606 ;       The default directory string is copied from the target process
                                     0253   607 ;       P1 space to the end of the extended ACB.
                                     0253   608 ;
                                     0253   609 ;  Side Effects:
                                     0253   610 ;
                                     0253   611 ;       If the initial calling process still exists, a special AST
                                     0253   612 ;       is queued to that process. The routine PUT_STRING will be
                                     0253   613 ;       the AST that executes in the context of the original caller.
                                     0253   614 ;-
                                     0253   615
                                     0253   616 GET_STRING:
                0070 8F         BB   0253   617         PUSHR   #^M<R4,R5,R6>
          0C A5    30 A5        D0   0257   618         MOVL    ACB_L_OLD_PID(R5),ACB$L_PID(R5) ;Turn ACB around
    18 A5    20 A5    0C        C1   025C   619         ADDL3   #12,ACB_L_PUT_AST(R5),ACB$L_KAST(R5) ;Different special AST
        08 A5    80 8F          88   0262   620         BISB2   #<1@ACB$V_KAST>,ACB$B_RMOD(R5)  ;Reset special bit
   53    00000000'GF            9E   0267   621         MOVAB   G^PIO$GT_DDSTRING,R3
                56    83        9A   026E   622         MOVZBL  (R3)+,R6                  ;Save string count in R6
          52    3C A5           9E   0271   623         MOVAB   ACB_T_DDSTRING(R5),R2     ;R2 located counted string in ACB
                82    56        90   0275   624         MOVB    R6,(R2)+                  ;R2 is now correctly updated
0053 8F    00    63    56       2C   0278   625         MOVC5   R6,(R3),#0,#ACB_K_STR_LEN,(R2)
                0070 8F         BA   0280   626         POPR    #^M<R4,R5,R6>
                                     0284   627
                                     0284   628 ;  Now queue an AST back to the original caller
                                     0284   629
                                     0284   630 10$:    SETIPL  30$                       ;Need to raise IPL here
          51    30 A5           3C   0288   631         MOVZWL  ACB_L_OLD_PID(R5),R1      ;Get PID (PIX only) of caller
   52    00000000'GF            D0   028F   632         MOVL    G^SCH$GL_PCBVEC,R2        ;Get its PCB address
          51    6241            D0   0296   633         MOVL    (R2)[R1],R1               ; in a PIC manner
       0C A5    60 A1           D1   029A   634         CMPL    PCB$L_PID(R1),ACB$L_PID(R5) ;Same PID in both places?
                      1A        12   029E   635         BNEQ    20$                       ;Error if not
       15 24 A1    01           E0   02A1   636         BBS     #PCB$V_DELPEN,PCB$L_STS(R1),20$
                      52        D4   02A6   637         CLRL    R2                        ;No boost going this way
        00000000'GF             16   02A8   638         JSB     G^SCH$QAST
                                     02AE   639         SETIPL  #IPL$_ASTDEL              ;Lower IPL back to 2
          50    1C A5           D0   02B1   640         MOVL    ACB_L_GET_AST(R5),R0      ;and return to AST dispatcher
        00000000'GF             17   02B5   641         JMP     G^EXE$DEANONPAGED         ; through DEANONPAGED
                                     02BB   642
```

## Listing 2-3: GET String ASt Procedure (Page 2 of 2)



```
:S     )D                  Get Default Directory String              17-MAY-1982 10:11:16  VAX-11 Macro V03-00
 04                        GET_STRING - Get string from user buffer 17-MAY-1982 10:11:06  WORK:CMUIZNIEKS.SYSPRG.US

                               02B8   643 ;  Original caller has gone away. Deallocate ACB and simply exit.
                               02B8   644
                               02B8   645 20$:    SETIPL   #IPL$_ASTDEL           ;Lower IPL to 2
             1C A5     DD      02BE   646         PUSHL    ACB_L_GET_AST(R5)      ;Save GET AST block across deall
     50      20 A5     D0      02C1   647         MOVL     ACB_L_PUT_AST(R5),R0   ;PUT AST block is the first to g
  00000000'GF          16      02C5   648         JSB      G^EXE$DEANONPAGED
     50      55        D0      02C8   649         MOVL     R5,R0                  ;Now do the ACB
  00000000'GF          16      02CE   650         JSB      G^EXE$DEANONPAGED      ;Deallocate ACB
     50      8E        D0      02D4   651         MOVL     (SP)+,R0               ;Finally deallocate the block
                               02D7   652                                         ; containing this code
  00000000'GF          17      02D7   653         JMP      G^EXE$DEANONPAGED      ;Jump there. RSB in EXE$DEANONPA
                               02DD   654                                         ; will return to AST dispatcher.
                               02DD   655
             00000007          02DD   656 30$:    .LONG    IPL$_SYNCH
                               02E1   657
                               02E1   658         ASSUME   <.-10$> LE 512
                               02E1   659
             0000098E          02E1   660       . GET_LENGTH = . - GET_STRING
                               02E1   661
```

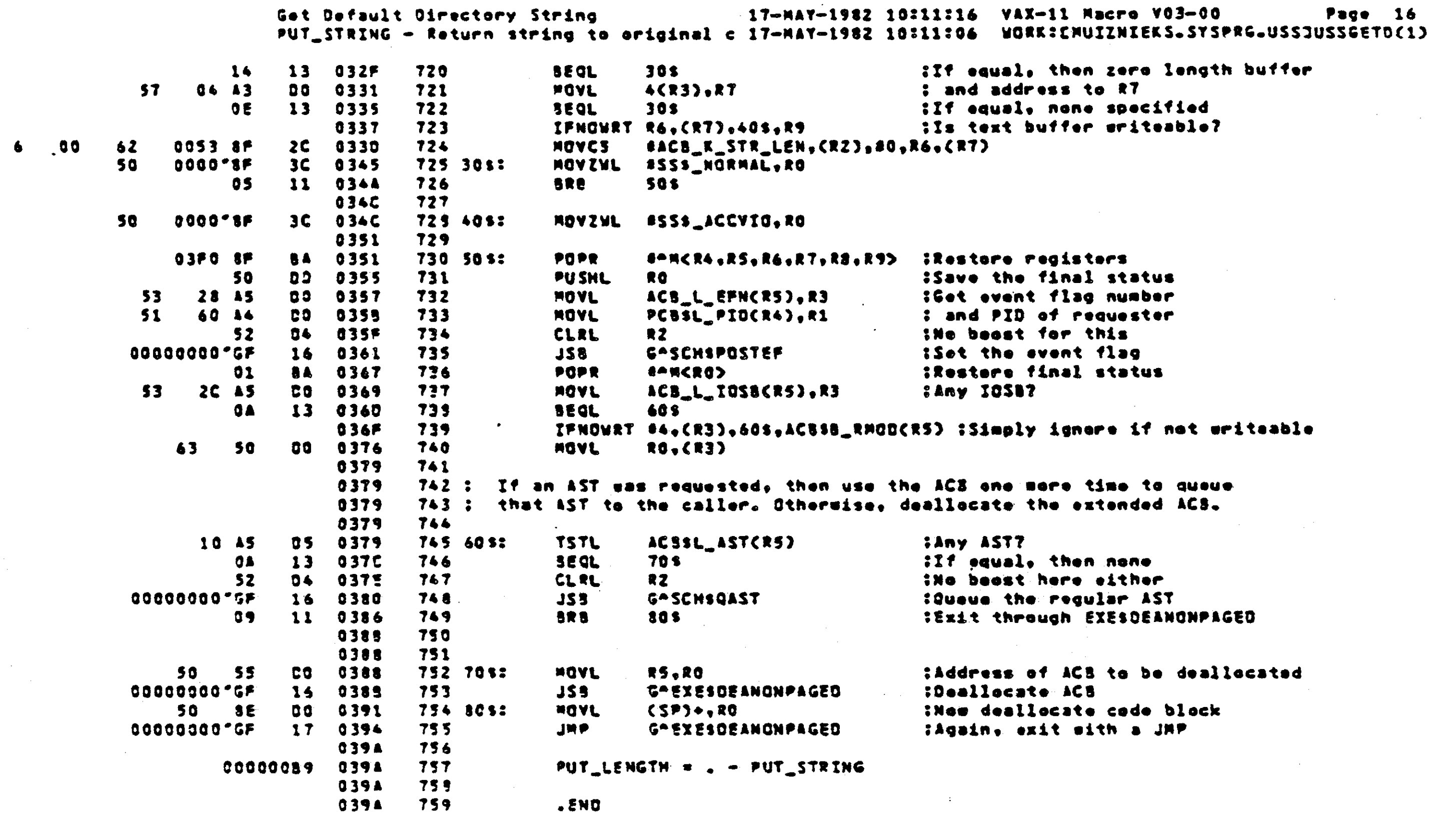
## The PUT String AST Procedure

- Executes in context of caller process
  - At IPL 2, since special kernel AST
  - Entered via JSB, not CALL, from AST delivery interrupt service routine
- Checks to make sure image that called system service still executing
  - To insure IOSB and AST address are addresses in calling image, not some arbitrary image
  - An image counter in process header, PHD$L_IMGCNT, is incremented as part of image rundown
    - Access PHD through P1 space window
- Rechecks protection of IOSB and location to receive directory string
  - Needed because of asynchronous nature of service
  - Possible for image to change protection of pages while ASTs executing ($SETPRT system service)
  - Prevents possible kernel mode access violations
- Copies default directory string to specified buffer
- Sets specified event flag
- Loads IOSB, if requested
- Tests for user-requested AST
  - If requested, reuses same ACB for user AST
  - If not requested, deallocates ACB
- Exits by jumping to deallocate nonpaged pool routine
  - To deallocate block with PUT string procedure code
  - To return control to AST delivery interrupt service routine

## Listing 2-4: PUT String AST Procedure (Page 1 of 2)

```
U      TOO                         Get Default Directory String           17-MAY-1982 10:11:16  VAX-11 Macro V03-00
V\                                 PUT_STRING - Return string to original c 17-MAY-1982 10:11:06  WORK:[MUIZNIEKS.SYSPRG.U

                                       02E1             .SUBTITLE       PUT_STRING - Return string to original caller
                                       02E1
                                       02E1     ;+
                                       02E1     ;  This routine executes as a special kernel AST in the context of
                                       02E1     ;  the original caller. It moves the default directory string
                                       02E1     ;  of the target process from the extended ACB into the user specified |
                                       02E1     ;
                                       02E1     ;  Input Parameters:
                                       02E1     ;
                                       02E1     ;       R0:R3 - Scratch
                                       02E1     ;       R4 - PCB address of original caller
                                       02E1     ;       R5 - Address of extended ACB
                                       02E1     ;
                                       02E1     ;  Calling Sequence:
                                       02E1     ;
                                       02E1     ;       JSB     PUT_STRING      from AST delivery routine at IPL 2
                                       02E1     ;
                                       02E1     ;  Output Parameters:
                                       02E1     ;
                                       02E1     ;       The default directory string is copied from the extended ACB
                                       02E1     ;       to the user specified buffer.
                                       02E1     ;
                                       02E1     ;  Side Effects:
                                       02E1     ;
                                       02E1     ;       If all access checks are OK, data is moved to user buffer. The
                                       02E1     ?       designated event flag is set. An AST may be delivered if the
                                       02E1     ;       original call requested one.  Buffered quota bytes are returned
                                       02E1     ;-
                                       02E1
                                       02E1     PUT_STRING:
                          20 A5    DD  02E1             PUSHL   ACB_L_PUT_AST(R5)       ;Save address of block containi
                                       02E4                                             ; code for JMP exit through rou
                                       02E4                                             ; EXE$DEANONPAGED
                     50   78 A4    D0  02E4             MOVL    PCB$L_JIB(R4),R0        ;Restore quota bytes charged
                  10 A0   38 A5    C0  02E8             ADDL    ACB_L_QUOTA(R5),JIB$L_BYTCNT(R0)
                                       02ED
                                       02ED     ;  Make sure that the same image is running.
                                       02ED
              53   00000000'GF     D0  02ED             MOVL    G^CTL$GL_PHD,R3
               36 A5    00F0 C3    D1  02F4             CMPL    PHD$L_IMGCNT(R3),ACB_L_IMGCNT(R5)
                             08    13  02FA             BEQL    10$
                  03 08 A5   06    E1  02FC             BBC     #ACB$V_QUOTA,ACB$B_RMOD(R5),5$ ;Was caller's AST quota
                          38 A4    B6  0301             INCW    PCB$W_ASTCNT(R4)        ;Give it back because ASTDEL,
                                       0304                                             ; which usually gives back AST
                                       0304                                             ; cannot, because it never gets
                           0081    31  0304     5$:     BRW     70$
                                       0307
                        03F0 8F    BB  0307     10$:    PUSHR   #^M<R4,R5,R6,R7,R8,R9>  ;Save some registers
                     52   3C A5    9E  030B             MOVAB   ACB_T_DDSTRING(R5),R2   ;Address of counted string in A
                     53   24 A5    D0  030F             MOVL    ACB_L_DDDESC(R5),R3     ;Get buffer descriptor
                     59   0B A5    9A  0313             MOVZBL  ACB$B_RMOD(R5),R9       ;Get caller's original access m
                                       0317             IFNORD  #12,(R3),40$,R9         ;Can it still be read?
                     58   08 A3    D0  031D             MOVL    8(R3),R8                ;Get address of length buffer
                             09    13  0321             BEQL    20$
                                       0323             IFNOWRT #2,(R8),40$,R9          ;Is it writeable?
                        68   82    9B  0329             MOVZBW  (R2)+,(R8)
                        56   63    B0  032C     20$:    MOVW    (R3),R6                 ;Buffer size to R6
```

Listing 2-4: PUT String AST Procedure (Page 2 of 2)



```
                             Get Default Directory String            17-MAY-1982 10:11:16  VAX-11 Macro V03-00         Page  16
                             PUT_STRING - Return string to original c 17-MAY-1982 10:11:06  WORK:[MUIZNIEKS.SYSPRG.USS]USSGETD(1)

              14     13  032F    720            BEQL     30$                       ;If equal, then zero length buffer
       57   04 A3    D0  0331    721            MOVL     4(R3),R7                  ; and address to R7
              0E     13  0335    722            BEQL     30$                       ;If equal, none specified
                         0337    723            IFNOWRT  R6,(R7),40$,R9            ;Is text buffer writeable?
6   .00    62  0053 8F   2C  033D    724            MOVC5    #ACB_K_STR_LEN,(R2),#0,R6,(R7)
       50  0000'8F   3C  0345    725 30$:       MOVZWL   #SS$_NORMAL,R0
              05     11  034A    726            BRB      50$
                         034C    727
       50  0000'8F   3C  034C    728 40$:       MOVZWL   #SS$_ACCVIO,R0
                         0351    729
         03F0 8F     BA  0351    730 50$:       POPR     #^M<R4,R5,R6,R7,R8,R9>  ;Restore registers
              50     DD  0355    731            PUSHL    R0                        ;Save the final status
       53   28 A5    D0  0357    732            MOVL     ACB_L_EFN(R5),R3          ;Get event flag number
       51   60 A4    D0  035B    733            MOVL     PCB$L_PID(R4),R1          ; and PID of requester
              52     D4  035F    734            CLRL     R2                        ;No boost for this
      00000000'GF    16  0361    735            JSB      G^SCH$POSTEF              ;Set the event flag
              01     BA  0367    736            POPR     #^M<R0>                   ;Restore final status
       53   2C A5    D0  0369    737            MOVL     ACB_L_IOSB(R5),R3         ;Any IOSB?
              0A     13  036D    738            BEQL     60$
                         036F    739            IFNOWRT  #4,(R3),60$,ACB$B_RMOD(R5) ;Simply ignore if not writeable
          63   50    D0  0376    740            MOVL     R0,(R3)
                         0379    741
                         0379    742 ;  If an AST was requested, then use the ACB one more time to queue
                         0379    743 ;  that AST to the caller. Otherwise, deallocate the extended ACB.
                         0379    744
            10 A5    D5  0379    745 60$:       TSTL     ACB$L_AST(R5)             ;Any AST?
              0A     13  037C    746            BEQL     70$                       ;If equal, then none
              52     D4  037E    747            CLRL     R2                        ;No boost here either
      00000000'GF    16  0380    748            JSB      G^SCH$QAST                ;Queue the regular AST
              09     11  0386    749            BRB      80$                       ;Exit through EXE$DEANONPAGED
                         0388    750
                         0388    751
          50   55    D0  0388    752 70$:       MOVL     R5,R0                     ;Address of ACB to be deallocated
      00000000'GF    16  038B    753            JSB      G^EXE$DEANONPAGED         ;Deallocate ACB
          50   8E    D0  0391    754 80$:       MOVL     (SP)+,R0                  ;Now deallocate code block
      00000000'GF    17  0394    755            JMP      G^EXE$DEANONPAGED         ;Again, exit with a JMP
                         039A    756
            00000089     039A    757            PUT_LENGTH = . - PUT_STRING
                         039A    758
                         039A    759            .END
```

## Listing 2-5: Test Program (Page 1 of 2)

```
                                                                     17-MAY-1982 10:11:39  VAX-11 Macro V03-00
                                                                     27-MAR-1982 15:27:16  WORK:CMUIZNIEKS.SYSPRG.US
                                     0000     1 ;                                                      TEST.MAR
                                     0000     2
                                     0000     3 ;       This program tests the default directory system
                                     0000     4 ;       service.  It prompts the user for a process name
                                     0000     5 ;       and returns that user's default directory string.
                                     0000     6 ;
                                     0000     7 ;       Note that the process being examined must be in the
                                     0000     8 ;       same group as the process running this program; if
                                     0000     9 ;       not, the "no such process" error message will be
                                     0000    10 ;       generated (also, must use enter exact upper/lower
                                     0000    11 ;       case letters in process name).
                                     0000    12
                                 00000000    13         .psect nonshared_data   pic, noexe, long
                                     0000    14
                                     0000    15 arglist:                               ; argument list for call
                           00000007  0000    16         .long   7                      ; seven parameters
                           00000003  0004    17         .long   3                      ; use EFN # 3
                           00000028' 0008    18         .address pid                   ; in case use PID
                           0000002C' 000C    19         .address prcnaml               ; using process name
                           0000006A' 0010    20         .address dddescl               ; descriptor block
                           00000020' 0014    21         .address iosbl                 ; returned status
                           000000A9' 0018    22         .address astrout               ; AST routine address
                           00000000  001C    23         .long   0                      ; AST parameter
                           00000000  0020    24 iosbl:  .long   0                      ; also used in QIO calls
                           00000000  0024    25         .long   0
                           00000000  0028    26 pid:    .long   0                      ; not used in this test
                           00000000  002C    27 prcnaml: .long  0                      ; process name desc.
                           00000034' 0030    28         .address prc
                           0000005C  0034    29 prc:    .blkb   40                     ; will be supplied by user
65 6D 61 6E 20 73 73 65 63 6F 72 50  005C    30 pmpt:   .ascii  /Process name: /
                                20 3A  0068
                           0000000E  006A    31 pm = . - pmpt
                           00000200  006A    32 dddescl: .long  512                    ; leave lots of space
                           0000007A' 006E    33         .address buffer                ; default dir string returned
                           00000076' 0072    34         .address buflenl               ; length of default dir string
                           00000000  0076    35 buflenl: .long  0
                           0000027A  007A    36 buffer: .blkb   512
                                     027A    37
                               0000  027A    38 ttchan: .word   0                      ; used to communicate with tty
4F 43 24 53 59 53 00000284'010E000B' 027C    39 ttname: .ascid  /SYS$COMMAND/
                       44 4E 41 4D 4D  0284
                                     028F    40
6D 6F 72 66 20 65 67 61 73 73 65 4D  028F    41 ames:   .ascii  /Message from AST routine/
65 6E 69 74 75 6F 72 20 54 53 41 20  029B
                           00000018  02A7    42 alen = . - ames
                                     02A7    43
                                 00000000    44         .psect  code    pic, shr, nowrt
                           0000      0000    45 start:  .word   0                      ; save no registers
                                     0002    46
                                     0002    47 ;       Establish channel to terminal
                                     0002    48         $assign_s chan=ttchan, devnam=ttname
                      03 50     E8   0017    49         blbs    r0, 10$
                       0083     31   001A    50         brw     err
                                     001D    51
                                     001D    52 ;       Get process name
                                     001D    53 10$:    $qiow_s chan=ttchan,func=#io$_readprompt,iosb=iosbl, -
                                     001D    54                 p1=prc,p2=#40,p5=#pmpt,p6=#pm
```

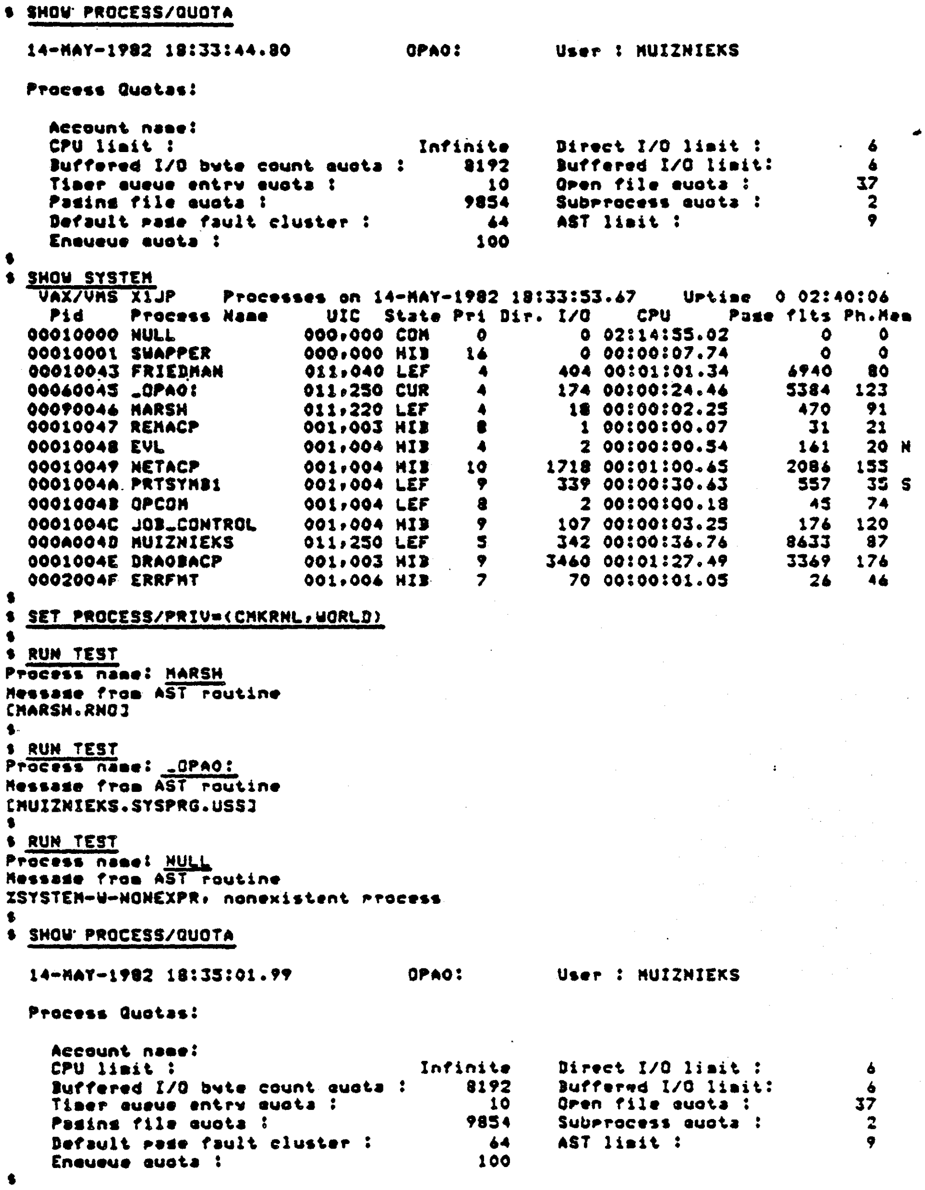
## Listing 2-5: Test Program (Page 2 of 2)

```
                                                        17-MAY-1982 10:11:39  VAX-11 Macro V03-00           Page   2
                                                        27-MAR-1982 15:27:16  WORK:[MUIZNIEKS.SYSPRG.USS]TEST.MA(1)

             51 50    E9  004C     55           blbc    r0, err
002C'EF   00000022'EF  3C  004F     56           movzwl  iosb1+2, prcnaml                 ; set process name len.
                           005A     57
                           005A     58 ;         Call user-written system service to get default dir. string
0000'GF   00000000'EF  FA  005A     59           callg   arglist,g^uss_getdd
             38 50    E9  0065     60           blbc    r0,err
                           0068     61
                           0068     62 ;         Wait for request to complete using EFN # 3
                           0068     63           $waitfr_s efn=#3
             2C 50    E9  0071     64           blbc    r0,err
                           0074     65
                           0074     66 ;         Print default directory
                           0074     67           $qiow_s chan=ttchan,func=#io$_writevblk,p1=buffer,p2=buflen1,p4=#32
                      04  009F     68           ret
                           00A0     69
                           00A0     70 ;         Error exit path
                           00A0     71 err:      $exit_s r0
                           00A9     72
                           00A9     73 ;         AST routine entered before event flag wait satisfied
                           00A9     74
                    0000  00A9     75 astrout: .word  0                             ; save no registers
                           00AB     76
                           00AB     77 ;         Display message indicating AST delivered
                           00AB     78           $qiow_s chan=ttchan,func=#io$_writevblk,p1=ames,p2=#alen,p4=#32
                      04  00C2     79           ret
                           00C3     80
                           00C3     81           .end    start
```

## Listing 2-6: MAKETEST.COM Command Procedure

```
$!                                              MAKETEST.COM
$!
$!      This command procedure builds all components for the
$!      default directory system service, dispatcher, and
$!      sample test program.
$!
$  SET VERIFY
$!
$!      First, assemble and link the system service and dispatcher
$!
$ MACRO/LIST USSGETDD.MAR + SYS$LIBRARY:LIB/LIB
$ LINK/PROTECT/NOSYSSHR/SHARE=USSGETDD/MAP=USSGETDD/FULL SYS$INPUT/OPTIONS
!
!       Options file for the link of User System Service example.
!
        SYS$SYSTEM:SYS.STB/SELECTIVE
!
!       Create a separate cluster for the transfer vector.
!
CLUSTER=TRANSTER_VECTOR,,,SYS$DISK:[]USSGETDD
!
GSMATCH=LEQUAL,1,1
$!
$!      Next, assemble and link test program
$!
$ MACRO/LIST TEST
$ LINK TEST/MAP/FULL,SYS$INPUT/OPTIONS
!
!       Options file for USSTEST
        USSGETDD.EXE/SHARE
$!
$!      Prepare to test program
$!
$ SET PROCESS/PRIV=(CMKRNL,SYSPRV,WORLD)
$ COPY USSGETDD.EXE SYS$SHARE:*.*
$ PURGE *.MAP, *.LIS, *.OBJ, *.EXE, SYS$SHARE:USSGETDD.EXE
$ RUN SYS$SYSTEM:INSTALL
SYS$SHARE:USSGETDD.EXE/SHARE/PROTECT
$ SET NOVERIFY
```

## Listing 2-7: Sample Run



```
$ SHOW PROCESS/QUOTA

  14-MAY-1982 18:33:44.80          OPA0:        User : MUIZNIEKS

  Process Quotas:

    Account name:
    CPU limit :                         Infinite    Direct I/O limit :           6
    Buffered I/O byte count quota :         8192    Buffered I/O limit:          6
    Timer queue entry quota :                 10    Open file quota :           37
    Paging file quota :                     9854    Subprocess quota :           2
    Default page fault cluster :              64    AST limit :                  9
    Enqueue quota :                          100
$
$ SHOW SYSTEM
   VAX/VMS X1JP    Processes on 14-MAY-1982 18:33:53.67    Uptime  0 02:40:06
    Pid      Process Name      UIC   State Pri Dir. I/O     CPU       Page flts Ph.Mem
  00010000 NULL            000,000 COM    0          0 02:14:55.02          0     0
  00010001 SWAPPER         000,000 HIB   16          0 00:00:07.74          0     0
  00010043 FRIEDMAN        011,040 LEF    4        404 00:01:01.34       6940    80
  00060045 _OPA0:          011,250 CUR    4        174 00:00:24.46       5384   123
  00090046 MARSH           011,220 LEF    4         18 00:00:02.25        470    91
  00010047 REMACP          001,003 HIB    8          1 00:00:00.07         31    21
  00010048 EVL             001,004 HIB    4          2 00:00:00.54        161    20 N
  00010049 NETACP          001,004 HIB   10       1718 00:01:00.65       2086   153
  0001004A PRTSYMB1        001,004 LEF    9        339 00:00:30.63        557    35 S
  0001004B OPCOM           001,004 LEF    8          2 00:00:00.18         45    74
  0001004C JOB_CONTROL     001,004 HIB    9        107 00:00:03.25        176   120
  000A004D MUIZNIEKS       011,250 LEF    5        342 00:00:36.76       8633    87
  0001004E DRAOBACP        001,003 HIB    9       3460 00:01:27.49       3369   176
  0002004F ERRFMT          001,006 HIB    7         70 00:00:01.05         26    46
$
$ SET PROCESS/PRIV=(CMKRNL,WORLD)
$
$ RUN TEST
Process name: MARSH
Message from AST routine
[MARSH.RNO]
$
$ RUN TEST
Process name: _OPA0:
Message from AST routine
[MUIZNIEKS.SYSPRG.USS]
$
$ RUN TEST
Process name: NULL
Message from AST routine
%SYSTEM-W-NONEXPR, nonexistent process
$
$ SHOW PROCESS/QUOTA

  14-MAY-1982 18:35:01.99          OPA0:        User : MUIZNIEKS

  Process Quotas:

    Account name:
    CPU limit :                         Infinite    Direct I/O limit :           6
    Buffered I/O byte count quota :         8192    Buffered I/O limit:          6
    Timer queue entry quota :                 10    Open file quota :           37
    Paging file quota :                     9854    Subprocess quota :           2
    Default page fault cluster :              64    AST limit :                  9
    Enqueue quota :                          100
$
```

# WRITING COMMAND LANGUAGE INTERPRETERS

## INTRODUCTION

Every interactive computer system requires some kind of command language that allows a user to specify the operations that are to be performed on the system. A Command Language Interpreter (CLI) is procedure-based code that executes in supervisor mode, in the context of a process, to receive, check the syntax of, parse, and perform commands entered by a user. VMS supports two CLIs - DCL and MCR. The DIGITAL Command Language (DCL) is the primary command language on VMS, while the Monitor Console Routine (MCR) is provided primarily for compatibility with PDP-11 systems.

There are several ways in which the command language interface can be altered. In previous courses you learned how to use symbols to tailor the DCL command environment. You also learned how to create foreign commands that could obtain information from the command line that invoked them. In addition to the command language, VMS provides a command language editor that allows you to add new commands to DCL (on a per-process or system-wide basis). It also provides a set of run-time routines for obtaining information from the command line.

If the type of command interface you need to build cannot be represented in a DCL-like manner (with commands that have parameters and qualifiers), or if you need faster command interpretation, or if you require features and capabilities that cannot be achieved using DCL, you may have to write a totally independent command language interpreter.

This module discusses how to use the DCL command language editor to extend the capabilities of DCL. In addition, the basic structure of a CLI is analyzed to aid you in writing your own CLI. It is highly recommended that you try to solve your problem by building on DCL, using the command language editor rather than writing your own CLI, since the system interfaces are much stabler, more formalized, and better documented.

# WRITING COMMAND LANGUAGE INTERPRETERS

## OBJECTIVES

- Use the command language editor to add a new command to DCL.
- Write an alternative command language interpreter (CLI) to DCL and MCR.

## RESOURCES

LOGINOUT source listings

DCL source listings

MCR source listings

VAX-11 Utilities Reference Manual

VAX/VMS Internals/Data Structures

# WRITING COMMAND LANGUAGE INTERPRETERS

## TOPICS

- Ways to modify the Command Language Interface
- Command Language Interpreter
- Command Language Editor
- VMS CLI
- Sample CLI

## Ways To Modify The Command Language Interface

| | METHOD | COMMENT |
|---|---|---|
| Modifying | DCL | |
| | Symbols | Per-process |
| | Foreign Commands | Per-process |
| | Command Language Editor | Per-process and System wide |
| Replacing | DCL | |
| | Writing your own CLI | Only if above methods fail or not possible |

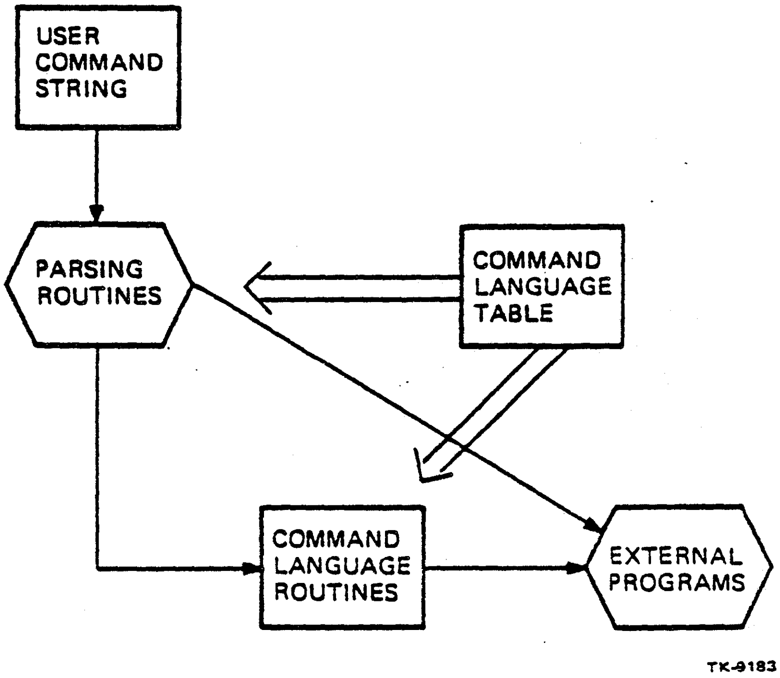


Figure 3-1 DCL (CLI) Overview

- Command String parsed according to information in command language table
- Internal routine or program invoked to perform requested operation

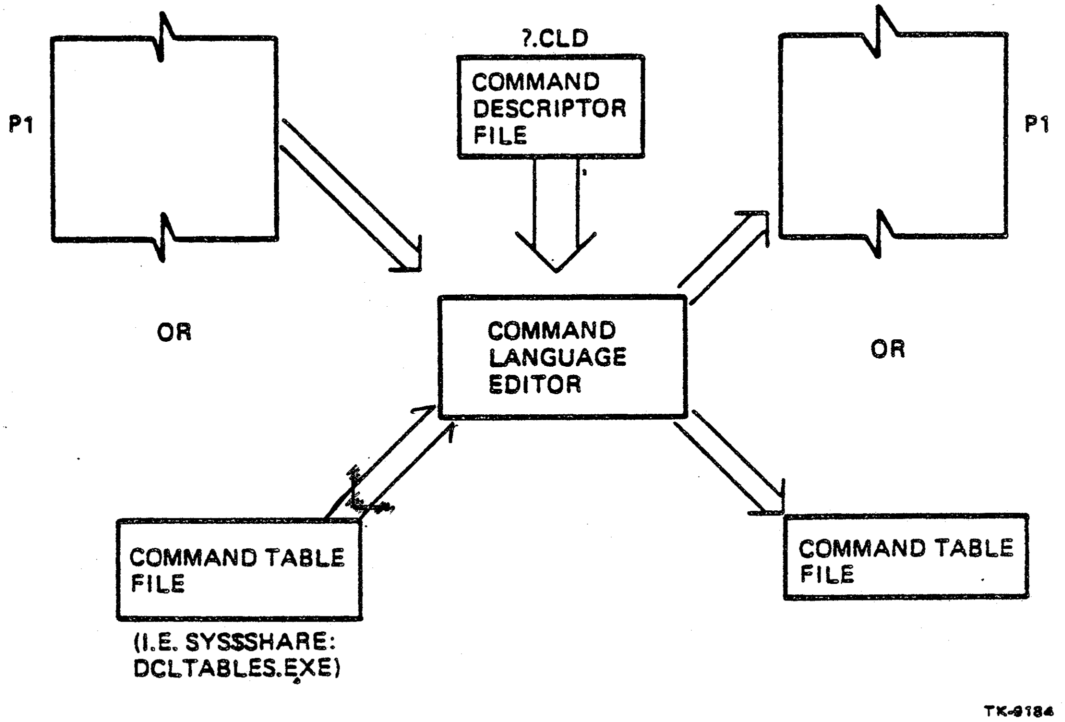


Figure 3-2 The Command Language Editor

- Command Language Editor invoked with the "SET COMMAND" DC command.

  Command format is:

  $SET COMMAND/qualifier(s) File_spec

- Command qualifiers and the file_spec govern what will occur The file_spec is called the "Command Language Descriptor file.

- The DCL table input can be from:
  1. Your Pl space
  2. A DCL Command Table File

- Output can go to:
  1. Your Pl space
  2. A DCL Command Table File

## The SET COMMAND Qualifiers

| FUNCTION | QUALIFIER |
|---|---|
| Specify the Command Table file to be edited | /TABLE= file_spec |
| Specify the output location of the edited Command Language Table file | /OUTPUT= file_spec |
| Delete one or more DCL verbs | /DELETE=(Verb,...) |
| Create Listing | /LISTING |
| Create Object file parsing table (See CLI$DCL_PARSE) | /OBJECT |

## Command Language Descriptor (CLD) File

The CLD file is the parameter file for the SET COMMAND. It contains information on DCL verbs that are to be added/modified.

### Basic Command Language Descriptor Format

```
DEFINE  VERB     winkin

        IMAGE    blinkin

        PARAMETER  P1, options

        QUALIFIER  qualifier_name, options
```

1. DEFINE VERB

   Specifies the verb's name is "winkin"

2. IMAGE

   The image to be invoked is called "blinkin".
   The directory default is SYS$SYSTEM.
   The file type default is .EXE.

3. PARAMETER

   Defines a parameter for the command.
   The symbol P1 identifies the first parameter.

4. QUALIFIER

   Defines a qualifier for the verb.

Examples

1. Add a verb to the P1 table

```
        $ SET COMMAND  GOGETEM.CLD

             GOGETEM.CLD
          -----------------------
          | DEFINE VERB DOIT     |
          |        IMAGE  DONE   |
          |                      |
          |----------------------|
```

2. Delete a verb in P1 space

```
         $ SET COMMAND/DELETE=(COPY)
```

3. Replace present dcltables in P1 space with a prepared set of tables.

```
         $ SET COMMAND/TABLE=SYS$SHARE:ALMOSTDCL.EXE
```

Using the /OUTPUT qualifier will alter each of the above examples to create DCL table files instead of changing P1 space.

## CLD Keywords

| | FUNCTION | KEYWORDS |
|---|---|---|
| MUTUALLY EXCLUSIVE | Image name to be run | IMAGE (DEF = VERB NAME) |
| | Internal routine name | ROUTINE |
| | Internal routine CLD | MODULE (discussed later) |
| | Parameter and position | PARAMETER [Pn] |
| | Parameter Characteristics | LABEL=name<br>PROMPT=string<br>VALUE= REQUIRED<br>DEFAULT=string<br>LIST<br>TYPE (discussed later) |
| | Qualifier and name | QUALIFIER name |
| | Qualifier characteristics | LABEL=name<br>PROMPT=string<br>DEFAULT<br>BATCH<br>[NON]NEGATABLE<br>PLACEMENT= GLOBAL<br>LOCAL<br>POSITIONAL<br>VALUE= DEFAULT<br>LIST<br>TYPE (discussed later)<br>SYNTAX (discussed later) |

Partial Listing of EDIT.CLD

```
define verb edit
  image edt
   prefix cli$k_edit_
  parameter p1,prompt="File",value(required,type=$infile)
          qualifier decide
  qualifier exact
  qualifier expert
  qualifier increment,        value
  qualifier isave,    value
  qualifier line
  qualifier listing,  value(type=$outfile)
  qualifier lower
  qualifier output,   value(type=$outfile)
  qualifier pline,    value
  qualifier read_only
  qualifier save,             value
  qualifier start,    value
  qualifier step,             value
  qualifier truncate, value
  qualifier tab
  qualifier checksum, value
  qualifier report
  qualifier header
  qualifier update,   value(list),placement=local
  qualifier edt
  qualifier command,  default,value(type=$infile)
  qualifier recover
  qualifier journal,  default,value(type=$infile)
  qualifier bak,              default
  qualifier num,              default
  outputs (output,listing)
```

Listing 3-1  Part of EDIT.CLD

## Using SYNTAX and TYPE to Modify Command Definitions

```
DEFINE SYNTAX name1

      [IMAGE]
      [PARAMETER]
      [QUAILFIER]

DEFINE TYPE name2

      KEYWORD

DEFINE VERB abbott
      IMAGE costello
      PARAMETER P1, VALUE(TYPE=name2)
      QUALIFIER HELP, SYNTAX=name1
```

- When qualifier HELP is used, the alternate SYNTAX name1 is used.
- TYPE name2 defines tha value used in the parameter.
- Name1 and name2 must be DEFINEd before being referenced in a parameter and/or qualifier. Note order in sample above.

## Some of the TYPE Designation Definitions

| TYPE Designations | Meaning |
|---|---|
| $DATETIME | Date/Time Specification |
| $DEVICE | Device Name |
| $DIRECTORY | Directory Specification |
| $INFILE | Input File Specification |
| $INLOG | Input Logical Name |
| $INSYM | Input Symbol Name |
| $NUMBER | Numeric Quantity |
| $OUTFILE | Output File Specification |
| $OUTLOG | Output Logical Name |
| $PROCESS | Process Name |
| $UIC | UIC Specification |
| $NODE | NO FORMAT CHECK AT THIS TIME |
| $OUTSYM | |
| $PRIVILEGE | |
| $PROTECTION | |

Partial Listing of EDIT.CLD

```
define syntax edit_using_sos         ❶
      image backtrans

define syntax edit_using_slp         ❷
      image backtrans

define syntax sumslp                 ❸
      image sumslp

define type audit_options            ❹
      keyword position,value
      keyword size,value

define verb edit
  image edt
   prefix cli$k_edit_
  parameter p1,prompt="File",value(required,type=$infile)
  qualifier audit_trail,      value(list,type=audit_options)
          qualifier decide                                ❹
  qualifier exact
  qualifier expert
  qualifier increment,        value
  qualifier isave,    value
  qualifier line
  qualifier listing,  value(type=$outfile)
  qualifier lower
  qualifier output,   value(type=$outfile)
  qualifier pline,    value
  qualifier read_only
  qualifier save,             value
  qualifier slp,              syntax=edit_using_slp    ❷
  qualifier sos,              syntax=edit_using_sos    ❶
  qualifier start,    value
  qualifier step,             value
  qualifier truncate, value
  qualifier tab
  qualifier checksum, value
  qualifier report
  qualifier header
  qualifier sum,              syntax=sumslp               ❸
  qualifier update,   value(list),placement=local
  qualifier edt
  qualifier command,  default,value(type=$infile)
  qualifier recover
  qualifier journal,  default,value(type=$infile)
  qualifier bak,              default
  qualifier num,              default
  outputs (output,listing)
```

Listing 3-2 Part of EDIT.CLD

## CLI Callback Routines

| FUNCTION | ROUTINE |
|---|---|
| Get value of parameter or qualifier | CLI$GET_VALUE(label,retbuf) |
| Check for presence of paramater or qualifier | CLI$PRESENT(label) |
| Issue error if some unprocessed parts of command in buffer | CLI$END_PARSE() |
| Parse a command string based on predefined parsing table | CLI$DCL_PARSE(cmd_string,table_names) |
| Invoke internal routine for verb specified | CLI$DISPATCH() |

```
;                                                 NAME.MAR
;
;  This program will obtain the information from
;  a command line and process it.
;
   .TITLE  NAME
   $DSCDEF
   $CLIDEF
;
   .PSECT  NONSHARED_DATA  PIC, NOEXE, LONG
FIRST_NAME:
   .BLKW   1
   .BYTE   DSC$K_DTYPE_T, DSC$K_CLASS_D
   .BLKL   1
LAST_NAME:
   .BLKW   1
   .BYTE   DSC$K_DTYPE_T, DSC$K_CLASS_D
   .BLKL   1
MIDDLE_NAME:
   .BLKW   1
   .BYTE   DSC$K_DTYPE_T, DSC$K_CLASS_D
   .BLKL   1
FIRST:     .ASCID  /FIRST/
LAST:      .ASCID  /LAST/
MIDDLE: .ASCID  /MIDDLE/
   .PSECT  CODE    PIC, SHR, NOWRT, LONG
   .ENTRY  BEGIN, ^M<>
;
;  Get values for parameters and qualifiers
   PUSHAQ  FIRST_NAME
   PUSHAQ  FIRST
   CALLS  #2,G^CLI$GET_VALUE
   PUSHAQ  LAST_NAME
   PUSHAQ  LAST
   CALLS  #2,G^CLI$GET_VALUE
   PUSHAQ  MIDDLE_NAME
   PUSHAQ  MIDDLE
   CALLS  #2,G^CLI$GET_VALUE
;
;  Process paramters and qualifiers
   PUSHAQ  FIRST_NAME
   CALLS  #1,G^LIB$PUT_OUTPUT
   PUSHAQ  MIDDLE_NAME
   CALLS  #1,G^LIB$PUT_OUTPUT
   PUSHAQ  LAST_NAME
   CALLS  #1,G^LIB$PUT_OUTPUT
   RET
   .END    BEGIN
```

Listing 3-3 NAME.MAR Program

## Command Language Descriptor File for NAME.MAR

```
DEFINE VERB name

   IMAGE my_directory:NAME

   PARAMETER p1, PROMPT="First Name",
                     LABEL=first,
                     VALUE(REQUIRED,LIST)

   PARAMETER p2, PROMPT="Last Name",
                  LABEL=last,
                  VALUE(DEFAULT="student")

   QUALIFIER middle, VALUE(REQUIRED)
```

## Sample Run of NAME.MAR

```
$NAME
%DCL-W-IVVERB unrecognized command
  \NAME\
$
$SET COMMAND  my_directory:NAME.CLD
$
$NAME
$_First Name: Laura
$_Last Name:
LAURA

student
$NAME/middle=elizabeth
$_First Name: Laura
$_Last Name: M
LAURA
ELIZABETH
M
$
```

```
;                                           COMMAND.MAR
;
;  This program contains a command language, and
;  uses the command language interface routines
;  to parse and process the commands.
;
;       The macro to check the status after a routine
;
   .MACRO  CHECK_STATUS    code=rØ, ?go

   blbs    code, go
   pushl   code
   calls   #1, g^lib$stop
go:
   .endm   check_status
;
   .TITLE  COMMAND
;  .LIBRARY        /MAR$LIB:MACROS/
   $CLIDEF
   $DSCDEF
;
   .PSECT  NONSHARED_DATA  PIC, NOEXE, LONG
COMMAND:
   .BLKW   1
   .BYTE   DSC$K_DTYPE_T, DSC$K_CLASS_D
   .BLKL   1
PROMPT_STRING:
   .ASCID  /TEST> /
   .PSECT  CODE    PIC, SHR, NOWRT, LONG
   .ENTRY  BEGIN, ^M<>
;
;  Get the input line
GET_COMMAND:
   PUSHAQ  PROMPT_STRING
   PUSHAQ  COMMAND
   CALLS   #2,G^LIB$GET_INPUT
   CHECK_STATUS
;
;  Check for valid syntax
   PUSHAQ  TEST_TABLES
   PUSHAQ  COMMAND
   CALLS   #2,G^CLI$DCL_PARSE
   CHECK_STATUS
;
;  Dispatch to the appropriate routine
   CALLS   #Ø,G^CLI$DISPATCH
   CHECK_STATUS
   BRW     GET_COMMAND
```

Listing 3-4 COMMAND.MAR program (page 1 of 3)

```
   .ENTRY  REPORT_COMMAND,^M<>
   .SAVE
   .PSECT  NONSHARED_DATA  PIC, NOEXE, LONG
ERROR:     .ASCID  /Error in file name/
SUB_NAME:
   .ASCID  /MY_SUB/
FILE_NAME:
   .BLKW   1
   .BYTE   DSC$K_DTYPE_T, DSC$K_CLASS_D
   .BLKL   1
MY_COMMAND:
   .ASCID  /PRINT /
DST:       .BLKW   1
   .BYTE   DSC$K_DTYPE_T, DSC$K_CLASS_D
   .BLKA   1
FILESPEC:
   .ASCID  /FILESPEC/
EDIT:      .ASCID  /EDIT/
   .RESTORE
   .PSECT  CODE    PIC, SHR, NOWRT, LONG
;
;  Retrieve the file specification
   PUSHAQ  FILESPEC
   CALLS   #1,G^CLI$PRESENT
   BLBS    R0,SUCCESS
   PUSHAQ  ERROR
   CALLS   #1,G^LIB$PUT_OUTPUT
   RET
SUCCESS:
   PUSHAQ  FILE_NAME
   PUSHAQ  FILESPEC
   CALLS   #2,G^CLI$GET_VALUE
   BLBS    R0,WORKED
   RET
WORKED:
   PUSHAQ  EDIT
   CALLS   #1,G^CLI$PRESENT
   BLBC    R0,PRINT
   CALLS   #0,EDIT_QUALIFIER
PRINT:
;
;  Create command
   CLRW    DST+DSC$W_LENGTH
   PUSHAQ  MY_COMMAND
   PUSHAQ  DST
   CALLS   #2,G^STR$APPEND
   PUSHAQ  FILE_NAME
   PUSHAW  DST
   CALLS   #2,G^STR$APPEND
```

Listing 3-4 COMMAND.MAR Program (page 2 of 3)

```
;
;    Print the file

     CLRQ     -(SP)
     CLRQ     -(SP)
     CLRQ     -(SP)
     PUSHAQ   SUB_NAME
     CLRQ     -(SP)
     CLRL     -(SP)
     PUSHAQ   DST
     CALLS    #10,G^LIB$SPAWN
     CHECK_STATUS
     RET


     .ENTRY   EXIT_COMMAND,^M<>
     $EXIT_S
     RET


     .ENTRY   EDIT_QUALIFIER, ^M<>
     .SAVE
     .PSECT   NONSHARED_DATA   PIC, NOEXE, LONG
EDT_COMMAND:
     .ASCID   /EDIT /
     .RESTORE
     .PSECT   CODE     PIC, SHR, NOWRT, LONG
;
;    Create command
     CLRW     DST+DSC$W_LENGTH
     PUSHAQ   EDT_COMMAND
     PUSHAQ   DST
     CALLS    #2,G^STR$APPEND
     PUSHAQ   FILE_NAME
     PUSHAW   DST
     CALLS    #2,G^STR$APPEND
;
;    Invoke EDT
     CLRQ     -(SP)
     CLRQ     -(SP)
     CLRQ     -(SP)
     PUSHAQ   SUB_NAME
     CLRQ     -(SP)
     CLRL     -(SP)
     PUSHAQ   DST
     CALLS    #10,G^LIB$SPAWN
     CHECK_STATUS
     RET
     .END     BEGIN
```

Listing 3-4 COMMAND.MAR (page 3 of 3)

Command Language Descriptor file for COMMAND.MAR

```
MODULE TEST_TABLES
DEFINE VERB REPORT
        ROUTINE REPORT_COMMAND
        PARAMETER P1, LABEL = FILESPEC
        QUALIFIER EDIT
DEFINE VERB EXIT
        ROUTINE EXIT_COMMAND
```

Sample run for COMMAND.MAR

```
$ SET COMMAND/OBJECT=TEST          TEST.CLD

$ MACRO/LIST                    COMMAND.MAR

$ LINK                         COMMAND,TEST

$ RUN COMMAND

 TEST>
```

## Legal Commands for COMMAND.MAR

1. REPORT        file_spec

   Print the file

2. REPORT/EDIT  file_spec

   Allow editing before printing the file

3. EXIT

   Exit the program

## CLI Overview

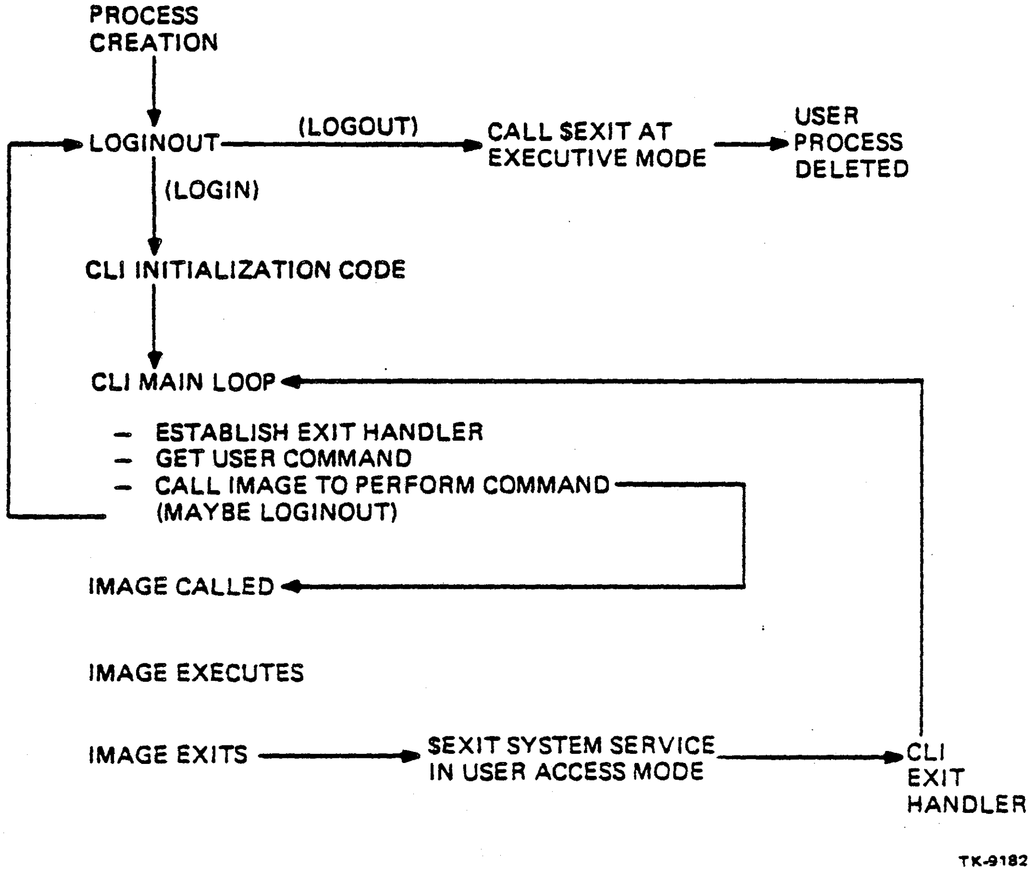


Figure 3-3 CLI Overview

# LOGINOUT

- Found in SYS$SYSTEM:LOGINOUT.EXE
- Validates username and password
- Maps requested CLI (from SYS$SYSTEM:cli.EXE)
    - CLI specified by /CLI=cliname after USERNAME
    - If missing, try value in UAF
    - If missing, uses DCL
    - Starting address of CLI in CTL$AG_CLIMAGE
    - CLI file must be INSTALLed (usually /SHARE)
- Establishes process permanent files (SYS$...)
- Initializes CLI-independent data area (PPD)
- Exits with REI (from executive mode)
    - To transfer control to CLI base address
    - CLI entry point should have NO entry mask
    - To switch to supervisor access mode

## LOGINOUT-CLI Communication

- Data Structures Involved (in P1 Space)

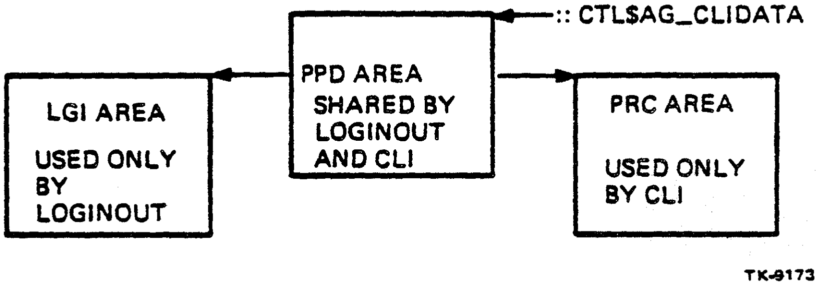


Figure 3-4 LOGINOUT-CLI Communication

- LOGINOUT to CLI Communication
  - Logical Names
    - PROC1-8 ; procedures to execute intially
    - P1-8 ; parameters for batch jobs
    - SYS$xxx ; process permanent files
  - PPD Area
    - Descriptor of CLI private data area
    - Descriptor for symbol table storage
    - Flags (Disable Control-Y, Type of Process)
    - Channel to SYS$INPUT
    - Descriptive information about SYS$INPUT and SYS$OUTPUT
- CLI to LOGINOUT Communication
  - Final status code in PPD area

## CLI Initialization Code

- Entered following REI from LOGINOUT (no entry mask)
- Establishes initial call frame
  - So can establish condition handler for CLI
  - Uses CALL instruction
- Establishes supervisor stack pointer (CTL$AL_STACK+8)
- Runs down LOGINOUT image ($RUNDWN)
- Deletes unneeded logical names created by LOGINOUT
  - Leaves process permanent files (in executive mode)
- Does any CLI-specific initialization
  - Initialize process RMS structures to allow terminal I/O
  - Establish CHMS handler ($DCLCMH)
    - Entered with change mode code on stack
    - Must remove code from stack
    - Exits with REI
  - Establish CLI callback routine (CTL$AL_CLICALBK)
  - Anything else the CLI needs to do once
- Enter CLI main loop

## CLI Main Loop

- Establish condition handler for CLI errors
- Establish exit handler ($DCLEXH)
  - Will return control back to main loop following image exit
  - If missing, process will be deleted
  - Want exit handler declared at supervisor access mode
- Prompt user for CLI-specific command
  - Do command specific processing within CLI, or
  - Run external image
- Loop for next command
  - Store final exit status in CLI-independent data area (PPD)
  - Run SYS$SYSTEM:LOGINOUT.EXE to log user off

## Invoking Images From CLI

- Save supervisor mode SP in CLI-specific fashion
  - Will be needed by exit handler to return control to main loop
- Use $IMGACT to activate image
  - Does not transfer control to image
  - Sets up page tables for image
- Change to user access mode (via REI)
- Create top level call frame for image (via CALL)
  - Establish EXE$CATCH_ALL as condition handler
- Establish EXE$CATCH_ALL as last chance handler ($SETEXV)
- Perform address relocations ($IMGFIX)
- Build CLI argument list to pass to image (6 arguments)

  1. Address of Transfer Vector Array
  2. Address of CLI Utility Dispatcher (Callback Routine)
  3. Address of Image Header
  4. Address of Image File Descriptor
  5. Link Flags From Image Header
  6. CLI Flags

- CALL image at first transfer vector
- If image returns, JMP to G^EXE$EXIT_IMAGE (for $EXIT)
  - Image may not return if calls $EXIT directly
- Control will be returned to main loop by exit handler

## Arguments to $IMGACT

NAME String descriptor of filename to activate

DFLTNAME String descriptor for default file name

HDRBUF Address of 512 byte buffer in which image header, image file descriptor, and address of most recently used FAB are returned. The first 3 longwords in the buffer are the addresses (in the buffer) of:

1. The image header ($IHDDEF)
2. The image file descriptor ($IFDDEF)
3. Address of FAB for most recent open (FAB is in image activator scratch pages, or Ø if no FAB available)

IMGCTL Image activation control parameters

Bit Ø = IAC$V_NOACNT (set if not activating image) Used by INSTALL to complete enhancement of known file entries

Bit 1 = IAC$V_WRITABLE (set if image is writeable)

Bit 2 = IAC$V_SHAREABLE (used in recursive call) Set if image is shareable image being activated as part of executable image

Bit 3 = IAC$V_PRIVILEGE (set if executable image has amplified privileges) Requires the shareable image to be installed as a known file (bit 2 also must be set)

Bit 4 = IAC$V_MERGE (merging one executable image into address space of another) Causes the stack, I/O segment, and privilege amplification to be ignored. Must be set to allow call from user access mode.

Bit 5 = IAC$V_EXPREG (set if INADR2 indicates which address region, PØ or P1, to use, instead of a specific address range) Bit 4 must also be set.

INADR2 Address of quadword containing input address range in which to place image.

RETADR2 Address of quadword to contain the return address range into which image actually mapped.

IDENT2 Address of quadword containing the version number and matching criteria for a shareable image.

ACMODE Access mode of owner of pages (not currently used).

## CLI Exit Handler

- CALLed after image exit (from $EXIT)
    - In supervisor mode
    - To rundown image ($RUNDWN)

- Should exit with RSB back to main loop

- First restore supervisor mode SP in CLI-specific way

- If exit with RET, process will be deleted
    - Control transferred back to $EXIT
    - $EXIT looks for more exit handlers
    - If none found, deletes process

## Sample CLI - MYCLI

- Based on DCL

- Prompts for image file name to run, can be:
  - Native mode image
  - Compatibility mode image

- For RSX utilities like PIP, can pass command lines:
  - SYS$SYSTEM:PIP   *.EXE/FU  (for full directory)

- Cannot pass command lines to DCL utilities
  - DCL callback facilities not implemented

- References various storage areas:
  - PPD$...  locations for LOGINOUT-CLI communication
  - PRC_...  as process work area (CLI-specific)
  - PRD_...  for RMS data structures (CLI-specific)
  - WRK_...  temporary stack storage (CLI-specific)

- Establishes control-Y AST routine:
  - Prints message when entered
  - Forces currently executing image to exit

## General Layout of Sample CLI

SUP$START:: Initialization Code

- Establishes supervisor mode SP
- Runs down LOGINOUT image
- Initializes private storage (PRC area)
- Establishes CLI callback routine (SUP$UTILSERV)
- Establishes control-Y AST routine (SUP$CTRLY)
- Establishes CHMS handler (HAND)
- Initializes RMS area (PRD) for terminal
- Displays CLI running message to user
- Enters main loop (SUP$RESTART)

SUP$RESTART:: Main Loop of CLI

- Establishes condition handler (SUP$EXCEPT) for CLI errors
- Gets user command
- Activates image (SUP$IMGACT)
  - Establishes exit handler (SUP$EXIT)
  - Calls $IMGACT
  - Saves supervisor mode SP
  - Changes to user access mode (REI)
  - Calls image

## General Layout of Sample CLI (Continued)

SUP$EXCEPT:: Condition Handler for CLI Errors

- No special recovery attempted
- Resets new exit handler that exits with success code (so process deleted), rather than returning control to CLI main loop

SUP$UTILSERV:: CLI Callback Routine

- Can be called from image, or by LOGINOUT when process being deleted
- Only handles "get command line" requests (for PIP-like utilities)
  - o Returns command line length and address

SUP$EXIT:: Exit Handler

- Runs down any open files
- Runs down user image
- Restores supervisor mode SP
- Returns control to main loop

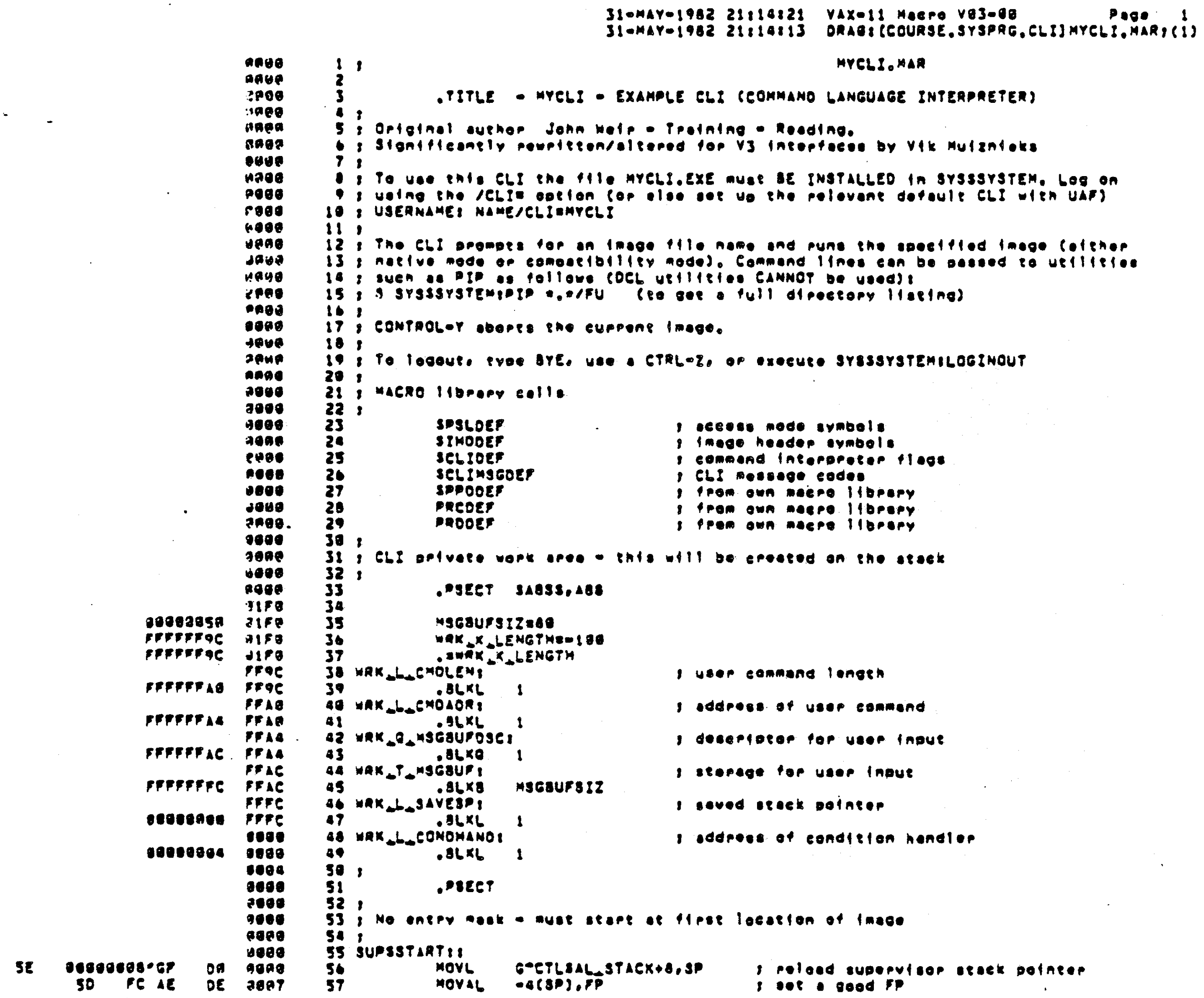
SUP$CTRLY:: Control-Y AST Routine

- Re-establishes itself for next control-Y
- Outputs message that routine entered
- Forces current image to exit (if any)

HAND:: Change Mode to Supervisor (CHMS) Handler

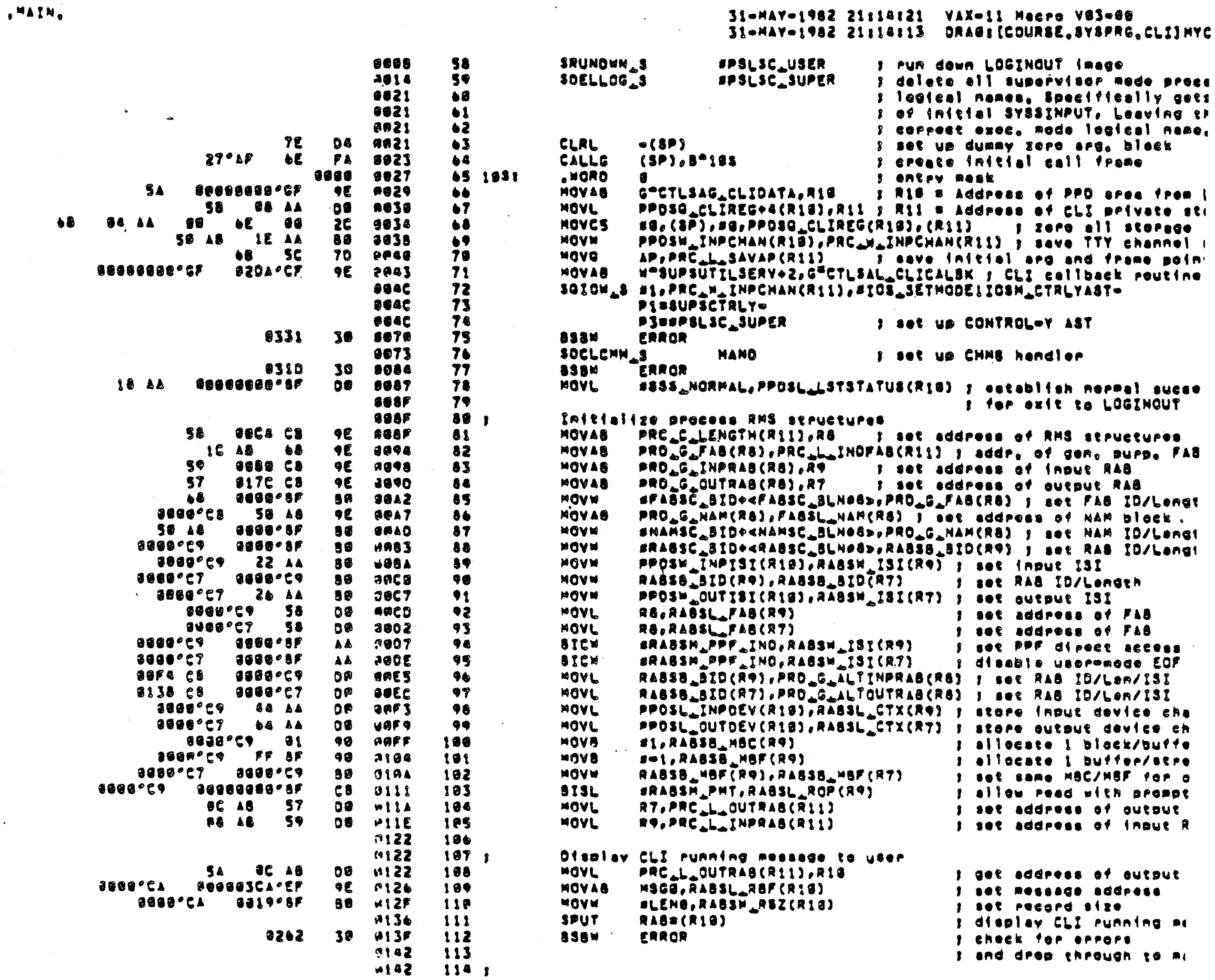
- No special action taken
- Removes change mode code from stack
- Dismisses CHMS exception with REI

Listing 3-5: Sample CLI (Page 1 of 6)



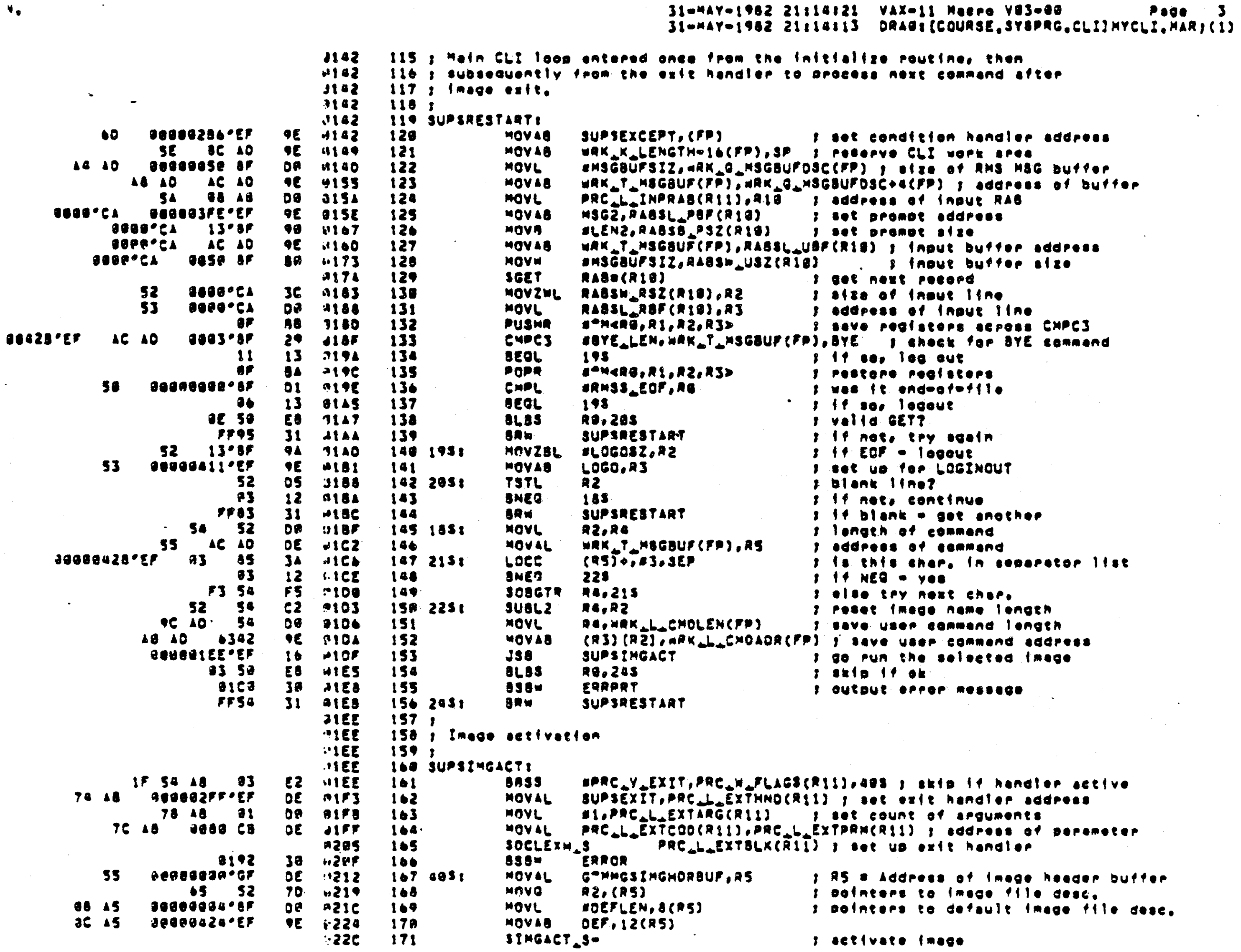
```
                                                     31-MAY-1982 21:14:21  VAX-11 Macro V03-00         Page   1
                                                     31-MAY-1982 21:14:13  DRA0:[COURSE.SYSPRG.CLI]MYCLI.MAR;(1)

                       0000     1 ;                                                      MYCLI.MAR
                       0000     2
                       0000     3         .TITLE  - MYCLI - EXAMPLE CLI (COMMAND LANGUAGE INTERPRETER)
                       0000     4 ;
                       0000     5 ; Original author  John Weir - Training - Reading.
                       0000     6 ; Significantly rewritten/altered for V3 interfaces by Vik Muiznieks
                       0000     7 ;
                       0000     8 ; To use this CLI the file MYCLI.EXE must BE INSTALLED in SYS$SYSTEM. Log on
                       0000     9 ; using the /CLI= option (or else set up the relevant default CLI with UAF)
                       0000    10 ; USERNAME: NAME/CLI=MYCLI
                       0000    11 ;
                       0000    12 ; The CLI prompts for an image file name and runs the specified image (either
                       0000    13 ; native mode or compatibility mode). Command lines can be passed to utilities
                       0000    14 ; such as PIP as follows (DCL utilities CANNOT be used):
                       0000    15 ; $ SYS$SYSTEM:PIP *.*/FU   (to get a full directory listing)
                       0000    16 ;
                       0000    17 ; CONTROL-Y aborts the current image.
                       0000    18 ;
                       0000    19 ; To logout, type BYE, use a CTRL-Z, or execute SYS$SYSTEM:LOGINOUT
                       0000    20 ;
                       0000    21 ; MACRO library calls
                       0000    22 ;
                       0000    23         $PSLDEF                         ; access mode symbols
                       0000    24         $IHDDEF                         ; image header symbols
                       0000    25         $CLIDEF                         ; command interpreter flags
                       0000    26         $CLIMSGDEF                      ; CLI message codes
                       0000    27         $PPODEF                         ; from own macro library
                       0000    28         PRCDEF                          ; from own macro library
                       0000    29         PRDDEF                          ; from own macro library
                       0000    30 ;
                       0000    31 ; CLI private work area - this will be created on the stack
                       0000    32 ;
                       0000    33         .PSECT  $ABS$,ABS
                       0000    34
             00000050  0000    35         MSGBUFSIZ=80
             FFFFFF9C  0000    36         WRK_K_LENGTH=-100
             FFFFFF9C  0000    37         .=WRK_K_LENGTH
                       FF9C    38 WRK_L_CMDLEN:                           ; user command length
             FFFFFFA0  FF9C    39         .BLKL   1
                       FFA0    40 WRK_L_CMDADR:                           ; address of user command
             FFFFFFA4  FFA0    41         .BLKL   1
                       FFA4    42 WRK_Q_MSGBUFDSC:                        ; descriptor for user input
             FFFFFFAC  FFA4    43         .BLKQ   1
                       FFAC    44 WRK_T_MSGBUF:                           ; storage for user input
             FFFFFFFC  FFAC    45         .BLKB   MSGBUFSIZ
                       FFFC    46 WRK_L_SAVESP:                           ; saved stack pointer
             00000000  FFFC    47         .BLKL   1
                       0000    48 WRK_L_CONDHAND:                         ; address of condition handler
             00000004  0000    49         .BLKL   1
                       0004    50 ;
                       0000    51         .PSECT
                       0000    52 ;
                       0000    53 ; No entry mask - must start at first location of image
                       0000    54 ;
                       0000    55 SUP$START::
5E   00000008'GF   D0  0000    56         MOVL    G^CTL$AL_STACK+8,SP     ; reload supervisor stack pointer
  5D   FC AE       DE  0007    57         MOVAL   -4(SP),FP               ; set a good FP
```

Listing 3-5: Sample CLI (Page 2 of 6)



```
.MAIN.                                                          31-MAY-1982 21:14:21  VAX-11 Macro V03-00
                                                                31-MAY-1982 21:14:13  DRA0:[COURSE.SYSPRG.CLI]MYC

                                    0000   58         $RUNDOWN_S      #PSL$C_USER     ; run down LOGINOUT image
                                    0014   59         $DELLOG_S       #PSL$C_SUPER    ; delete all supervisor mode proce
                                    0021   60                                         ; logical names. Specifically gets
                                    0021   61                                         ; of initial SYS$INPUT, Leaving th
                                    0021   62                                         ; correct exec. mode logical name,
                         7E   D4    0021   63         CLRL    -(SP)                   ; set up dummy zero arg. block
                 27'AF   6E   FA    0023   64         CALLG   (SP),B^10$              ; create initial call frame
                            0000    0027   65 10$:    .WORD   0                       ; entry mask
         5A   00000000'GF     9E    0029   66         MOVAB   G^CTL$AG_CLIDATA,R10    ; R10 = Address of PPD area from l
                  5B   08 AA  D0    0030   67         MOVL    PPD$Q_CLIREG+4(R10),R11 ; R11 = Address of CLI private st
   6B   04 AA   00   6E   00  2C    0034   68         MOVC5   #0,(SP),#0,PPD$Q_CLIREG(R10),(R11)     ; zero all storage
            58 AB   1E AA     B0    0038   69         MOVW    PPD$W_INPCHAN(R10),PRC_W_INPCHAN(R11) ; save TTY channel
                    6B   5C   7D    0040   70         MOVQ    AP,PRC_L_SAVAP(R11)     ; save initial arg and frame poin
    00000000'GF   020A'CF     9E    0043   71         MOVAB   W^SUP$UTILSERV+2,G^CTL$AL_CLICALBK ; CLI callback routine
                                    004C   72         $QIOW_S #1,PRC_W_INPCHAN(R11),#IO$_SETMODE!IO$M_CTRLYAST-
                                    004C   73                 P1=SUP$CTRLY-
                                    004C   74                 P3=#PSL$C_SUPER         ; set up CONTROL-Y AST
                      0331    30    0070   75         BSBW    ERROR
                                    0073   76         $DCLCMH_S       HAND            ; set up CHMS handler
                      031D    30    0084   77         BSBW    ERROR
        18 AA   00000000'8F   D0    0087   78         MOVL    #SS$_NORMAL,PPD$L_LSTSTATUS(R10) ; establish normal succe
                                    008F   79                                                  ; for exit to LOGINOUT
                                    008F   80 ;       Initialize process RMS structures
              58   00C4 CB    9E    008F   81         MOVAB   PRC_C_LENGTH(R11),R8    ; set address of RMS structures
               1C AB   68     9E    0094   82         MOVAB   PRD_G_FAB(R8),PRC_L_INDFAB(R11) ; addr. of gen. purp. FAB
              59   00B0 C8    9E    0098   83         MOVAB   PRD_G_INPRAB(R8),R9     ; set address of input RAB
              57   017C C8    9E    009D   84         MOVAB   PRD_G_OUTRAB(R8),R7     ; set address of output RAB
              68   0000'8F    B0    00A2   85         MOVW    #FAB$C_BID+<FAB$C_BLN@8>,PRD_G_FAB(R8) ; set FAB ID/Lengt
         0000'C8    50 A8     9E    00A7   86         MOVAB   PRD_G_NAM(R8),FAB$L_NAM(R8) ; set address of NAM block .
          50 A8   0000'8F     B0    00AD   87         MOVW    #NAM$C_BID+<NAM$C_BLN@8>,PRD_G_NAM(R8) ; set NAM ID/Lengt
        0000'C9   0000'8F     B0    00B3   88         MOVW    #RAB$C_BID+<RAB$C_BLN@8>,RAB$B_BID(R9) ; set RAB ID/Lengt
         0000'C9    22 AA     B0    00BA   89         MOVW    PPD$W_INPISI(R10),RAB$W_ISI(R9) ; set input ISI
        0000'C7   0000'C9     B0    00C0   90         MOVW    RAB$B_BID(R9),RAB$B_BID(R7)     ; set RAB ID/Length
         0000'C7    26 AA     B0    00C7   91         MOVW    PPD$W_OUTISI(R10),RAB$W_ISI(R7) ; set output ISI
            0000'C9    58     D0    00CD   92         MOVL    R8,RAB$L_FAB(R9)                ; set address of FAB
            0000'C7    58     D0    00D2   93         MOVL    R8,RAB$L_FAB(R7)                ; set address of FAB
        0000'C9   0000'8F     AA    00D7   94         BICW    #RAB$M_PPF_IND,RAB$W_ISI(R9)    ; set PPF direct access
        0000'C7   0000'8F     AA    00DE   95         BICW    #RAB$M_PPF_IND,RAB$W_ISI(R7)    ; disable user-mode EOF
        00F4 C8   0000'C9     D0    00E5   96         MOVL    RAB$B_BID(R9),PRD_G_ALTINPRAB(R8) ; set RAB ID/Len/ISI
        0138 C8   0000'C7     D0    00EC   97         MOVL    RAB$B_BID(R7),PRD_G_ALTOUTRAB(R8) ; set RAB ID/Len/ISI
         0000'C9    44 AA     D0    00F3   98         MOVL    PPD$L_INPDEV(R10),RAB$L_CTX(R9) ; store input device cha
         0000'C7    64 AA     D0    00F9   99         MOVL    PPD$L_OUTDEV(R10),RAB$L_CTX(R7) ; store output device ch
            0000'C9    01     90    00FF  100         MOVB    #1,RAB$B_MBC(R9)                ; allocate 1 block/buffe
         0000'C9    FF 8F     90    0104  101         MOVB    #-1,RAB$B_MBF(R9)               ; allocate 1 buffer/stre
        0000'C7   0000'C9     B0    010A  102         MOVW    RAB$B_MBF(R9),RAB$B_MBF(R7)     ; set same MBC/MBF for o
    0000'C9   00000000'8F     C8    0111  103         BISL    #RAB$M_PMT,RAB$L_ROP(R9)        ; allow read with prompt
               0C AB    57    D0    011A  104         MOVL    R7,PRC_L_OUTRAB(R11)            ; set address of output
               08 AB    59    D0    011E  105         MOVL    R9,PRC_L_INPRAB(R11)            ; set address of input R
                                    0122  106
                                    0122  107 ;       Display CLI running message to user
               5A    0C AB    D0    0122  108         MOVL    PRC_L_OUTRAB(R11),R10           ; get address of output
    0000'CA   000003CA'EF     9E    0126  109         MOVAB   MSG0,RAB$L_RBF(R10)             ; set message address
        0000'CA   0019'8F     B0    012F  110         MOVW    #LEN0,RAB$W_RSZ(R10)            ; set record size
                                    0136  111         $PUT    RAB=(R10)                       ; display CLI running m
                      0262    30    013F  112         BSBW    ERROR                           ; check for errors
                                    0142  113                                                 ; and drop through to m
                                    0142  114 ;
```

Listing 3-5: Sample CLI (Page 3 of 6)

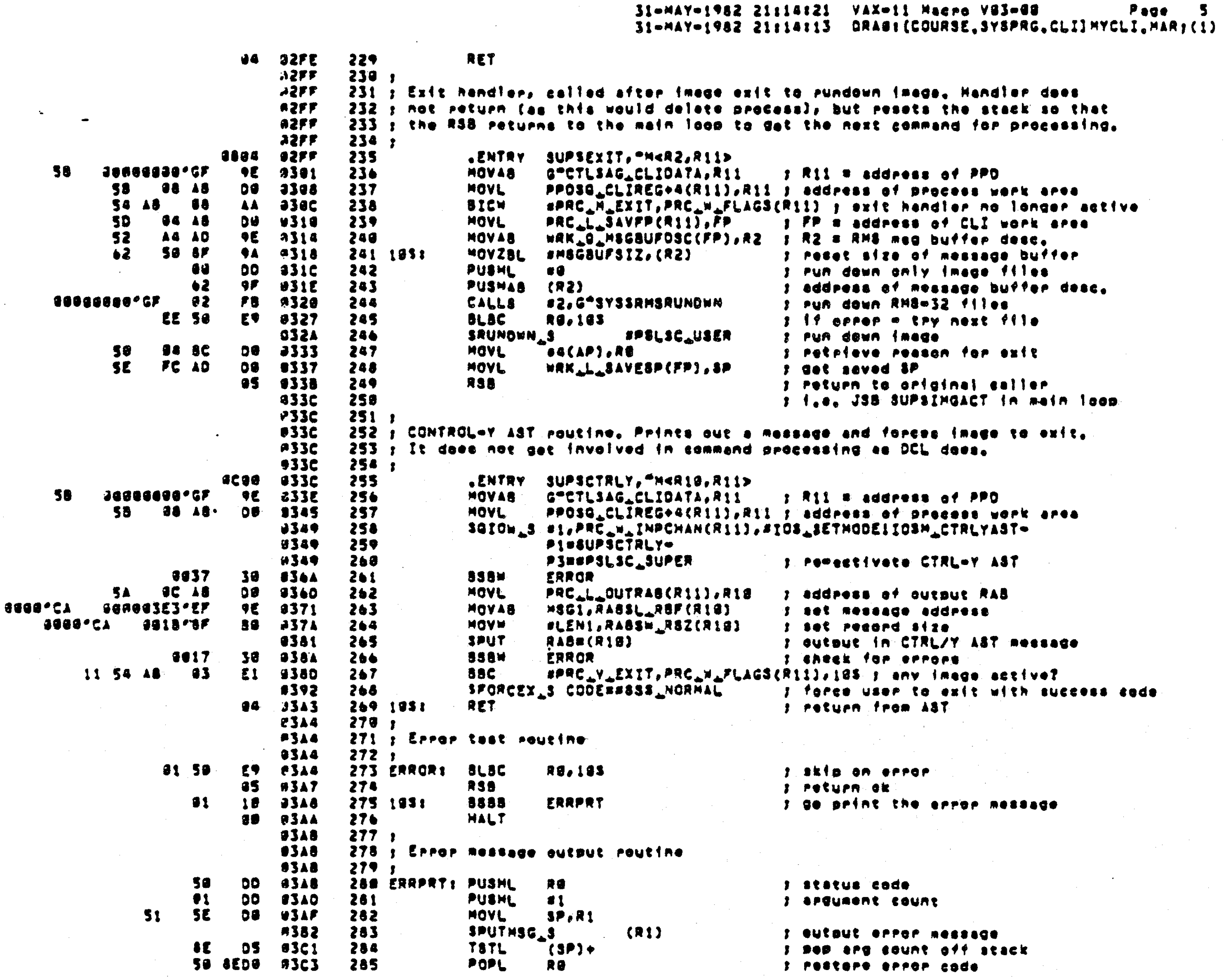


```
                                                   31-MAY-1982 21:14:21  VAX-11 Macro V03-00          Page   3
                                                   31-MAY-1982 21:14:13  DRA0:[COURSE.SYSPRG.CLI]MYCLI.MAR;(1)

                                0142      ; Main CLI loop entered once from the initialize routine, then
                                0142      ; subsequently from the exit handler to process next command after
                                0142      ; image exit.
                                0142      ;
                                0142      SUP$RESTART:
       60  00000286'EF    9E    0142              MOVAB   SUP$EXCEPT,(FP)          ; set condition handler address
            5E    8C AD   9E    0149              MOVAB   WRK_K_LENGTH-16(FP),SP  ; reserve CLI work area
    A4 AD  00000050 8F    D0    014D              MOVL    #MSGBUFSIZ,WRK_Q_MSGBUFDSC(FP) ; size of RMS MSG buffer
        A8 AD     AC AD   9E    0155              MOVAB   WRK_T_MSGBUF(FP),WRK_Q_MSGBUFDSC+4(FP) ; address of buffer
            5A    08 AB   D0    015A              MOVL    PRC_L_INPRAB(R11),R10    ; address of input RAB
  0000'CA  000003FE'EF    9E    015E              MOVAB   MSG2,RAB$L_PBF(R10)      ; set prompt address
      0000'CA     13'8F   90    0167              MOVB    #LEN2,RAB$B_PSZ(R10)     ; set prompt size
      0000'CA     AC AD   9E    016D              MOVAB   WRK_T_MSGBUF(FP),RAB$L_UBF(R10) ; input buffer address
    0000'CA     0050 8F   B0    0173              MOVW    #MSGBUFSIZ,RAB$W_USZ(R10)        ; input buffer size
                                017A              $GET    RAB=(R10)                ; get next record
       52     0000'CA     3C    0183              MOVZWL  RAB$W_RSZ(R10),R2        ; size of input line
       53     0000'CA     D0    0188              MOVL    RAB$L_RBF(R10),R3        ; address of input line
                    0F    BB    018D              PUSHR   #^M<R0,R1,R2,R3>         ; save registers across CMPC3
00428'EF   AC AD   0003'8F    29    018F          CMPC3   #BYE_LEN,WRK_T_MSGBUF(FP),BYE  ; check for BYE command
                    11    13    019A              BEQL    19$                      ; if so, log out
                    0F    BA    019C              POPR    #^M<R0,R1,R2,R3>         ; restore registers
      50   00000000'8F    D1    019E              CMPL    #RMS$_EOF,R0             ; was it end-of-file
                    06    13    01A5              BEQL    19$                      ; if so, logout
                 0E 50    E8    01A7              BLBS    R0,20$                   ; valid GET?
                  FF95    31    01AA              BRW     SUP$RESTART              ; if not, try again
         52     13'8F     9A    01AD 19$:         MOVZBL  #LOGOSZ,R2               ; if EOF - logout
      53   00000411'EF    9E    01B1              MOVAB   LOGO,R3                  ; set up for LOGINOUT
                    52    D5    01B8 20$:         TSTL    R2                       ; blank line?
                    03    12    01BA              BNEQ    18$                      ; if not, continue
                  FF83    31    01BC              BRW     SUP$RESTART              ; if blank - get another
               54   52    D0    01BF 18$:         MOVL    R2,R4                    ; length of command
          55     AC AD    DE    01C2              MOVAL   WRK_T_MSGBUF(FP),R5      ; address of command
 00000428'EF     03 85    3A    01C6 21$:         LOCC    (R5)+,#3,SEP             ; is this char. in separator list
                    03    12    01CE              BNEQ    22$                      ; if NEQ - yes
                 F3 54    F5    01D0              SOBGTR  R4,21$                   ; else try next char.
              52    54    C2    01D3 22$:         SUBL2   R4,R2                    ; reset image name length
           9C AD    54    D0    01D6              MOVL    R4,WRK_L_CMDLEN(FP)      ; save user command length
         A0 AD    6342    9E    01DA              MOVAB   (R3)[R2],WRK_L_CMDADR(FP) ; save user command address
           000001EE'EF    16    01DF              JSB     SUP$IMGACT               ; go run the selected image
                 03 50    E8    01E5              BLBS    R0,24$                   ; skip if ok
                  01C3    30    01E8              BSBW    ERRPRT                   ; output error message
                  FF54    31    01EB 24$:         BRW     SUP$RESTART
                                01EE      ;
                                01EE      ; Image activation
                                01EE      ;
                                01EE      SUP$IMGACT:
          1F 54 AB   03   E2    01EE              BBSS    #PRC_V_EXIT,PRC_W_FLAGS(R11),40$ ; skip if handler active
   74 AB   000002FF'EF    DE    01F3              MOVAL   SUP$EXIT,PRC_L_EXTHND(R11) ; set exit handler address
             78 AB   01   D0    01FB              MOVL    #1,PRC_L_EXTARG(R11)     ; set count of arguments
       7C AB   0080 CB    DE    01FF              MOVAL   PRC_L_EXTCOD(R11),PRC_L_EXTPRM(R11) ; address of parameter
                                0205              $DCLEXH_S        PRC_L_EXTBLK(R11) ; set up exit handler
                  0192    30    020F              BSBW    ERROR
      55   00000000'GF    DE    0212 40$:         MOVAL   G^MMG$IMGHDRBUF,R5       ; R5 = Address of image header buffer
              65    52    7D    0219              MOVQ    R2,(R5)                  ; pointers to image file desc.
   08 A5   00000004'8F    D0    021C              MOVL    #DEFLEN,8(R5)            ; pointers to default image file desc.
   0C A5   00000424'EF    9E    0224              MOVAB   DEF,12(R5)
                                022C              $IMGACT_S-                       ; activate image
```

## Listing 3-5: Sample CLI (Page 4 of 6)

```
.MAIN.                                                      31-MAY-1982 21:14:21  VAX-11 Macro V03-00
                                                            31-MAY-1982 21:14:13  DRA0:[COURSE,SYSPRG,CLI]MYCl

                        022C              NAME=(R5)-                     ; image file name
                        022C              DFLNAM=8(R5)-                  ; default file name
                        022C              HDRBUF=(R5)                    ; image header buffer
            0F 50   E8  0244        BLBS    R0,40$                       ; if set - activation ok
               50   DD  0247        PUSHL   R0                           ; save error code
                        0249        $RUNDWN_S       #PSL$C_USER          ; run down bad image
            50 8ED0     0252        POPL    R0                           ; restore error code
                    05  0255        RSB                                  ; return to main loop
     FC AD    5E    D0  0256 40$:   MOVL    SP,WRK_L_SAVESP(FP)          ; save current SP
  7E    0F    16    78  025A        ASHL    #PSL$V_PRVMOD,#PSL$C_USER@2+PSL$C_USER,-(SP)
                        025E                                             ; set up user PSL on stack
          01'AF     9F  025E        PUSHAB  B^50$                        ; set up user PC on stack
                    02  0261        REI                                  ; switch to user mode
               5C   7C  0262 50$:   CLRQ    AP                           ; clear AP,FP
     00'AF     00   FB  0264        CALLS   #0,B^60$                     ; set top level call frame
                  0000  0268 60$:   .WORD   0
 6D     00000000'GF 9E  026A        MOVAB   G^EXE$CATCH_ALL,(FP)         ; set exception to catch all
                        0271        $SETEXV_S #2,G^EXE$CATCH_ALL         ; last chance vector
                        0284        $IMGFIX_S                            ; perform address relocation
            22 50   E9  0288        BLBC    R0,65$                       ; exit if error
 54     00000000'GF 7D  028E        MOVQ    G^MMG$IMGHDRBUF,R4           ; addresses of image header & fil
                        0295 ; CLI argument list
               7E   D4  0295        CLRL    -(SP)                        ; CLI flags
            20 A4   DD  0297        PUSHL   IHD$L_LNKFLAGS(R4)           ; link flags from image header
         7E    54   7D  029A        MOVQ    R4,-(SP)                     ; image file name & image header
 7E     00000208'EF 9E  029D        MOVAB   SUP$UTILSERV,-(SP)           ; CLI callback address
     50    02 A4    3C  02A4        MOVZWL  IHD$W_ACTIVOFF(R4),R0        ; offset to transfer vectors
        50     54   C0  02A8        ADDL    R4,R0                        ; address of transfer vector arra
               60   DF  02AB        PUSHAL  (R0)                         ; save address of transfer vector
        90     06   FB  02AD        CALLS   #6,@(R0)+                    ; call image entry
    00000000'GF     17  02B0 65$:   JMP     G^EXE$EXIT_IMAGE             ; go do $EXIT_S
                        02B6 ;
                        02B6 ; Condition handler for CLI errors. Not called for user errors as these
                        02B6 ; are caught by EXE$CATCH_ALL which prints error dump and does an exit.
                        02B6 ; CLI errors are special - reset exit handler and jump to EXE$CATCH_ALL.
                        02B6 ;
                  0800  02B6              .ENTRY  SUP$EXCEPT,^M<R11>
 5B     00000000'GF 9E  02B8        MOVAB   G^CTL$AG_CLIDATA,R11         ; get address of PPD
     5B    08 AB    D0  02BF        MOVL    PPD$Q_CLIREG+4(R11),R11      ; get address of process work are
  78 AB    CE'AF    9E  02C3        MOVAB   B^10$,PRC_L_EXTHND(R11)      ; reset exit handler address
    00000002'GF     17  02C8        JMP     G^EXE$CATCH_ALL+2            ; take special error exit path
                  0000  02CE 10$:   .WORD   0                            ; entry mask for special error h
 50     00000000'8F D0  02D0        MOVL    #SS$_NORMAL,R0               ; set success
                    04  02D7        RET
                        02D8 ;
                        02D8 ; CLI service routine to pass remainder of command line to utilities
                        02D8 ; as for RSX type GMCRS.  Note that will NOT handle requests from DCL
                        02D8 ; utilities like DIRECTORY.  Assumes callback request of proper type.
                        02D8 ;
                  0C00  02D8              .ENTRY  SUP$UTILSERV,^M<R10,R11>
 5B     00000000'GF 9E  02DA        MOVAB   G^CTL$AG_CLIDATA,R11         ; address of PPD
     5B    08 AB    D0  02E1        MOVL    PPD$Q_CLIREG+4(R11),R11      ; address of process work area
     5B    04 AB    D0  02E5        MOVL    PRC_L_SAVFP(R11),R11         ; address of saved FP
     5A    04 AC    D0  02E9        MOVL    4(AP),R10                    ; address of CLI request block
  08 AA    0C AB    D0  02ED        MOVL    WRK_L_CMDLEN(R11),CLI$W_RQSIZE(R10) ; return line length
  0C AA    10 AB    D0  02F2        MOVL    WRK_L_CMDADR(R11),CLI$A_RQADDR(R10) ; return line addres
 50     00030001 8F D0  02F7        MOVL    #CLI$_NORMAL,R0              ; set success return
```

Listing 3-5: Sample CLI (Page 5 of 6)



```
                                          31-MAY-1982 21:14:21  VAX-11 Macro V03-00         Page   5
                                          31-MAY-1982 21:14:13  DRA0:[COURSE.SYSPRG.CLI]MYCLI.MAR;(1)

                   04 02FE          RET
                      02FF    ;
                      02FF    ; Exit handler, called after image exit to rundown image. Handler does
                      02FF    ; not return (as this would delete process), but resets the stack so that
                      02FF    ; the RSB returns to the main loop to get the next command for processing.
                      02FF    ;
                 0804 02FF          .ENTRY  SUP$EXIT,^M<R2,R11>
5B   00000000'GF   9E 0301          MOVAB   G^CTL$AG_CLIDATA,R11     ; R11 = address of PPD
  5B     08 AB     D0 0308          MOVL    PPD$Q_CLIREG+4(R11),R11 ; address of process work area
  54 AB     08     AA 030C          BICW    #PRC_M_EXIT,PRC_W_FLAGS(R11) ; exit handler no longer active
  5D     04 AB     D0 0310          MOVL    PRC_L_SAVFP(R11),FP      ; FP = address of CLI work area
  52     A4 AD     9E 0314          MOVAB   WRK_Q_MSGBUFDSC(FP),R2   ; R2 = RMS msg buffer desc.
  62     50 8F     9A 0318 10$:     MOVZBL  #MSGBUFSIZ,(R2)          ; reset size of message buffer
            00     DD 031C          PUSHL   #0                       ; run down only image files
            62     9F 031E          PUSHAB  (R2)                     ; address of message buffer desc.
00000000'GF 02     FB 0320          CALLS   #2,G^SYS$RMSRUNDOWN      ; run down RMS-32 files
         EE 50     E9 0327          BLBC    R0,10$                   ; if error - try next file
                      032A          $RUNDOWN_S      #PSL$C_USER      ; run down image
  50     04 BC     D0 0333          MOVL    @4(AP),R0                ; retrieve reason for exit
  5E     FC AD     D0 0337          MOVL    WRK_L_SAVESP(FP),SP      ; get saved SP
                   05 033B          RSB                              ; return to original caller
                      033C                                           ; i.e. JSB SUP$IMGACT in main loop
                      033C    ;
                      033C    ; CONTROL-Y AST routine. Prints out a message and forces image to exit.
                      033C    ; It does not get involved in command processing as DCL does.
                      033C    ;
                 0C00 033C          .ENTRY  SUP$CTRLY,^M<R10,R11>
5B   00000000'GF   9E 033E          MOVAB   G^CTL$AG_CLIDATA,R11     ; R11 = address of PPD
  5B     08 AB     D0 0345          MOVL    PPD$Q_CLIREG+4(R11),R11 ; address of process work area
                      0349          $QIOW_S #1,PRC_W_INPCHAN(R11),#IO$_SETMODE!IO$M_CTRLYAST-
                      0349                  P1=SUP$CTRLY-
                      0349                  P3=#PSL$C_SUPER          ; re-activate CTRL-Y AST
            0037   30 036A          BSBW    ERROR
     5A  0C AB     D0 036D          MOVL    PRC_L_OUTRAB(R11),R10    ; address of output RAB
0000'CA   000003E3'EF  9E 0371      MOVAB   MSG1,RAB$L_RBF(R10)      ; set message address
  0000'CA   0018'8F  B0 037A        MOVW    #LEN1,RAB$W_RSZ(R10)     ; set record size
                      0381          $PUT    RAB=(R10)                ; output in CTRL/Y AST message
            0017   30 038A          BSBW    ERROR                    ; check for errors
    11 54 AB   03  E1 038D          BBC     #PRC_V_EXIT,PRC_W_FLAGS(R11),10$ ; any image active?
                      0392          $FORCEX_S CODE=#SS$_NORMAL       ; force user to exit with success code
                   04 03A3 10$:     RET                              ; return from AST
                      03A4    ;
                      03A4    ; Error test routine
                      03A4    ;
         01 50     E9 03A4 ERROR:   BLBC    R0,10$                   ; skip on error
                   05 03A7          RSB                              ; return ok
            01     10 03A8 10$:     BSBB    ERRPRT                   ; go print the error message
                   00 03AA          HALT
                      03AB    ;
                      03AB    ; Error message output routine
                      03AB    ;
            50     DD 03AB ERRPRT:  PUSHL   R0                       ; status code
            01     DD 03AD          PUSHL   #1                       ; argument count
      51    5E     D0 03AF          MOVL    SP,R1
                      03B2          $PUTMSG_S       (R1)             ; output error message
            8E     D5 03C1          TSTL    (SP)+                    ; pop arg count off stack
         50 8ED0      03C3          POPL    R0                       ; restore error code
```

## Listing 3-5: Sample CLI (Page 6 of 6)

```
.MAIN.                                                                                      31-MAY-1982 21:14:21  VAX-11 Macro V03-00      P
                                                                                            31-MAY-1982 21:14:13  DRA0:[COURSE.SYSPRG.CLI]MYCL

                                          05  03C6   286            RSB
                                              03C7   287 ;
                                              03C7   288 ; CHMS handler - for this CLI is a no-op
                                              03C7   289 ;
                                    8E    D5  03C7   290 HAND::     TSTL    (SP)+                       ; remove change mode code from stac
                                          02  03C9   291            REI                                 ; return after CHMS call
                                              03CA   292 ;
                                              03CA   293 ;          Messages displayed at terminal
                                              03CA   294 ;
                                    00000020  03CA   295            SPACE = 32
                                    0000000D  03CA   296            CR = 13
                                    0000000A  03CA   297            LF = 10
                                    00000009  03CA   298            TAB = 9
 4D 41 58 45 20 2A 2A 2A 09 09 0A 0D  03CA   299 MSG0:      .ASCII  <CR><LF><TAB><TAB>/*** EXAMPLE CLI ***/<CR><LF>
 0D 2A 2A 2A 20 49 4C 43 20 45 4C 50  03D6
                                  0A  03E2
                            00000019  03E3   300            LEN0=.-MSG0
 54 4E 4F 43 20 2A 2A 2A 09 09 0A 0D  03E3   301 MSG1:      .ASCII  <CR><LF><TAB><TAB>/*** CONTROL-Y AST ***/<CR><LF>
 2A 2A 20 54 53 41 20 59 2D 4C 4F 52  03EF
                            0A 0D 2A  03FB
                            0000001B  03FE   302            LEN1=.-MSG1
 47 41 4D 49 20 52 45 54 4E 45 0A 0D  03FE   303 MSG2:      .ASCII  <CR><LF>/ENTER IMAGE NAME /
          20 45 4D 41 4E 20 45  040A
                            00000013  0411   304            LEN2=.-MSG2
                                      0411   305 ;
 4C 3A 4D 45 54 53 59 53 24 53 59 53  0411   306 LOGO:      .ASCII  /SYS$SYSTEM:LOGINOUT/    ; image name for LOGOUT
          54 55 4F 4E 49 47 4F  041D
                            00000013  0424   307            LOGOSZ=.-LOGO
                         45 58 45 2E  0424   308 DEF:       .ASCII  /.EXE/                      ; defaults for image file name
                            00000004  0428   309            DEFLEN=.-DEF
                                      0428   310 ;
                            2F 09 20  0428   311 SEP:       .ASCII  <SPACE><TAB>"/"             ; separators
                                      042B   312 ;
                            45 59 42  042B   313 BYE:       .ASCII  /BYE/                       ; command to leave system
                            00000003  042E   314            BYE_LEN=.-BYE                       ; length of command
                                      042E   315 ;
                                      042E   316            .END    SUP$START
```

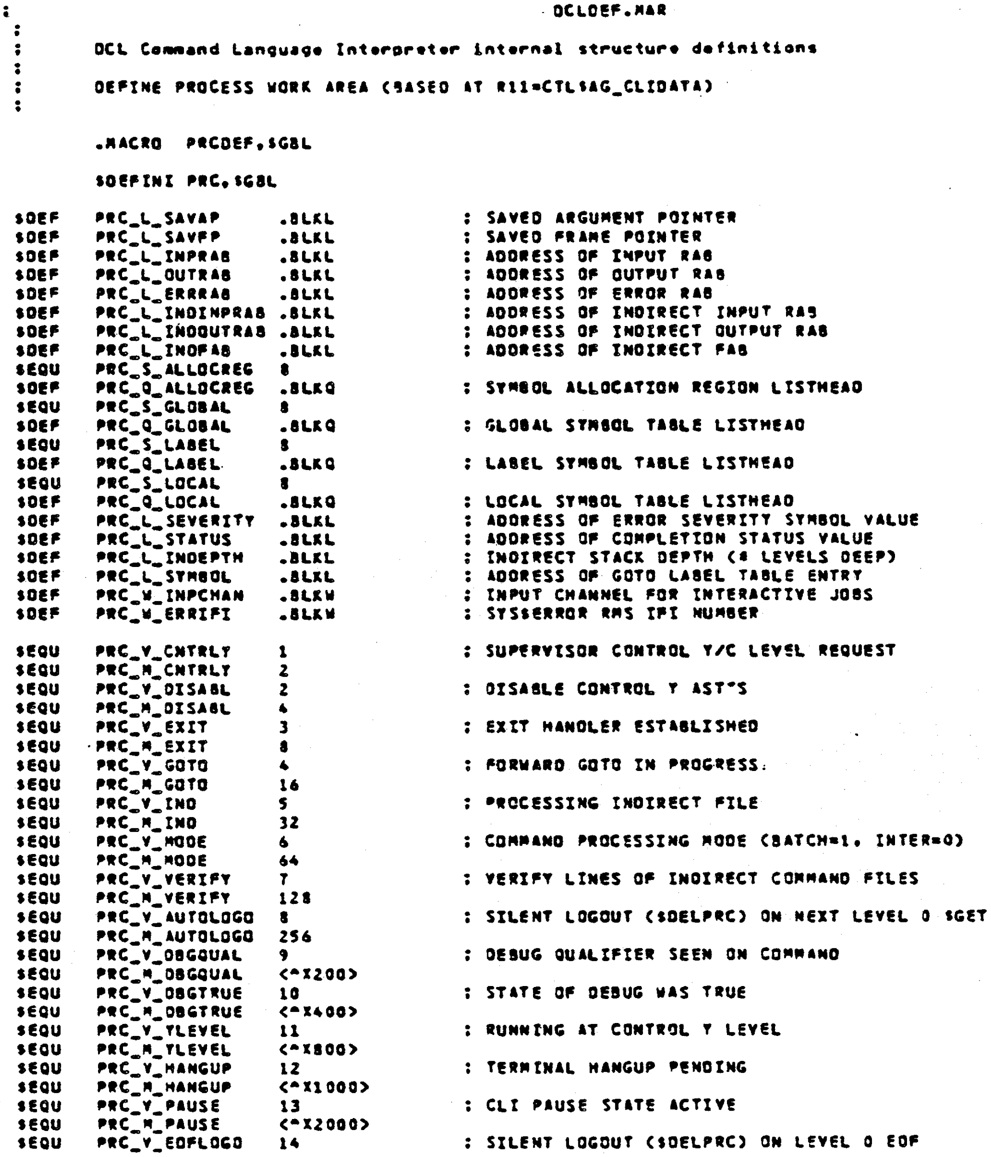
## Listing 3-6: PPDDEF.MAR File (Page 1 of 2)

```
;                                                              PPDDEF.MAR
;       LOGINOUT data structure definitions
;
; Define LOGIN <--> CLI <--> LOGOUT communication region
;
; This structure is based at CTL$AG_CLIDATA.  It contains all cells
; which are used by both LOGINOUT and the CLI.
;
; In addition to the following data items, the following logical names
; are also passed from LOGIN to the CLI initialization code:
;
;       PROC1-8                         ; Procedures to initially execute
;       P1-8                            ; Initial parameters for batch jobs
;       SYS$INPUT                       ; Primary input stream
;       SYS$OUTPUT                      ; Primary output stream
;       SYS$ERROR                       ; Primary error stream
;       SYS$COMMAND                     ; Command terminal
;

        .MACRO  $PPDDEF,$GBL

        $DEFINI PPD,$GBL


$DEF    PPD$W_SIZE      .BLKW           ; Actual size of structure
$EQU    PPD$V_NOCTLY    0               ; Initially disable CTRL/Y in CLI
$EQU    PPD$M_NOCTLY    1
$EQU    PPD$V_MODE      1               ; 1 if network, batch, or detached
$EQU    PPD$M_MODE      2               ; 0 if subprocess or interactive
$DEF    PPD$W_FLAGS     .BLKW           ; Flags
$EQU    PPD$S_CLIREG    8
$DEF    PPD$Q_CLIREG    .BLKB   4       ; Descriptor of CLI private data storage
                                        ; (approximately 2-3 pages or so)
$DEF    PPD$L_PRC       .BLKL           ; Address of CLI private data storage
$EQU    PPD$S_CLISYMTBL 8
$DEF    PPD$Q_CLISYMTBL .BLKQ           ; Descriptor of symbol table storage
                                        ; (size from SYSGEN param CLISYMTBL)
$DEF    PPD$L_LGI       .BLKL           ; Address of LOGINOUT private storage
$DEF    PPD$L_LSTSTATUS .BLKL           ; Final status code from CLI to LOGOUT
$DEF    PPD$B_NPROCS    .BLKB           ; Number of procedures to initially
                                        ; execute (names in lognames PROC1-N)
                        .BLKB   1
$DEF    PPD$W_INPCHAN   .BLKW           ; Channel to SYS$INPUT (used to $CANCEL
                                        ; outstanding I/O)
$DEF    PPD$W_INPIFI    .BLKW           ; SYS$INPUT IFI
$DEF    PPD$W_INPISI    .BLKW           ; SYS$INPUT ISI
$DEF    PPD$W_OUTIFI    .BLKW           ; SYS$OUTPUT IFI
$EQU    PPD$C_DVIFID    28              ; Length of DVI/DID/FID block
$DEF    PPD$W_OUTISI    .BLKW           ; SYS$OUTPUT ISI
$EQU    PPD$S_INPDVI    16
$DEF    PPD$T_INPDVI    .BLKB   16      ; SYS$INPUT ASCIC device name
$EQU    PPD$S_INPDID    6
$DEF    PPD$W_INPDID    .BLKW   3       ; SYS$INPUT directory file id
$EQU    PPD$S_INPFID    6
$DEF    PPD$W_INPFID    .BLKW   3       ; SYS$INPUT file id
$DEF    PPD$L_INPDEV    .BLKL           ; SYS$INPUT device characteristics
$EQU    PPD$S_OUTDVI    16
$DEF    PPD$T_OUTDVI    .BLKB   16      ; SYS$OUTPUT ASCIC device name
$EQU    PPD$S_OUTDID    6
$DEF    PPD$W_OUTDID    .BLKW   3       ; SYS$OUTPUT directory file id
$EQU    PPD$S_OUTFID    6
```

## Listing 3-6: PPDDEF.MAR File (Page 2 of 2)

```
$DEF    PPD$W_OUTFID    .BLKW   3       ; SYS$OUTPUT file id
$DEF    PPD$L_OUTDEV    .BLKL           ; SYS$OUTPUT device characteristics
$DEF    PPD$C_LENGTH

$DEF    PPD$K_LENGTH                    ; Length of fixed portion
        $DEFEND PPD,$GBL,DEF

        .ENDM   $PPDDEF
```

## Listing 3-7: DCLDEF.MAR File (Page 1 of 3)

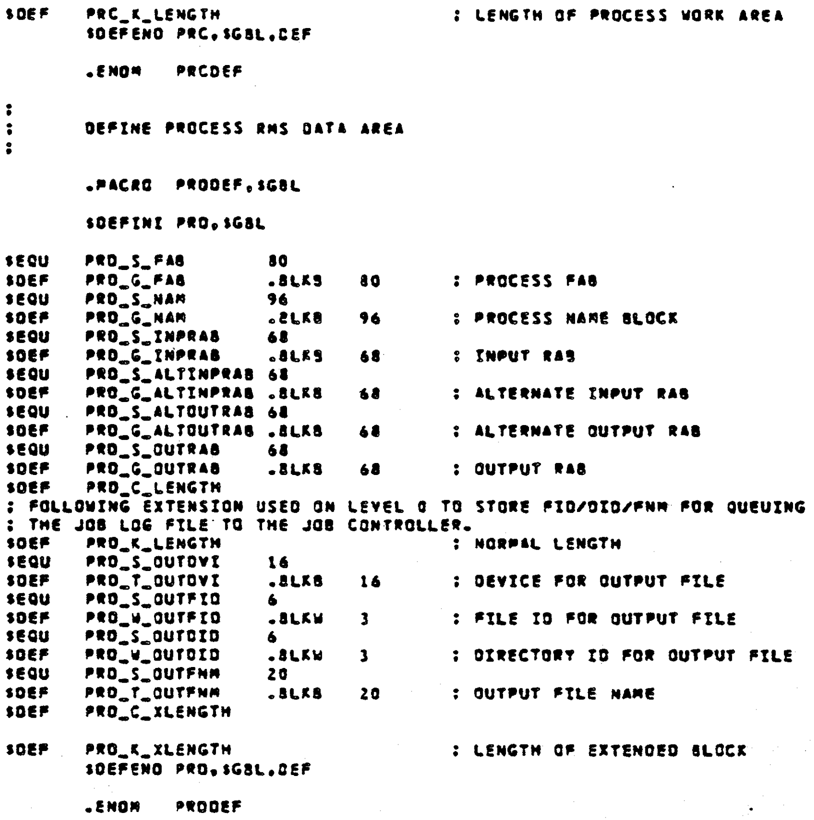


```
;                                          DCLDEF.MAR
;
;       DCL Command Language Interpreter internal structure definitions
;
;       DEFINE PROCESS WORK AREA (BASED AT R11=CTL$AG_CLIDATA)
;

        .MACRO  PRCDEF,$GBL

        $DEFINI PRC,$GBL

$DEF    PRC_L_SAVAP     .BLKL           ; SAVED ARGUMENT POINTER
$DEF    PRC_L_SAVFP     .BLKL           ; SAVED FRAME POINTER
$DEF    PRC_L_INPRAB    .BLKL           ; ADDRESS OF INPUT RAB
$DEF    PRC_L_OUTRAB    .BLKL           ; ADDRESS OF OUTPUT RAB
$DEF    PRC_L_ERRRAB    .BLKL           ; ADDRESS OF ERROR RAB
$DEF    PRC_L_INDINPRAB .BLKL           ; ADDRESS OF INDIRECT INPUT RAB
$DEF    PRC_L_INDOUTRAB .BLKL           ; ADDRESS OF INDIRECT OUTPUT RAB
$DEF    PRC_L_INDFAB    .BLKL           ; ADDRESS OF INDIRECT FAB
$EQU    PRC_S_ALLOCREG  8
$DEF    PRC_Q_ALLOCREG  .BLKQ           ; SYMBOL ALLOCATION REGION LISTHEAD
$EQU    PRC_S_GLOBAL    8
$DEF    PRC_Q_GLOBAL    .BLKQ           ; GLOBAL SYMBOL TABLE LISTHEAD
$EQU    PRC_S_LABEL     8
$DEF    PRC_Q_LABEL     .BLKQ           ; LABEL SYMBOL TABLE LISTHEAD
$EQU    PRC_S_LOCAL     8
$DEF    PRC_Q_LOCAL     .BLKQ           ; LOCAL SYMBOL TABLE LISTHEAD
$DEF    PRC_L_SEVERITY  .BLKL           ; ADDRESS OF ERROR SEVERITY SYMBOL VALUE
$DEF    PRC_L_STATUS    .BLKL           ; ADDRESS OF COMPLETION STATUS VALUE
$DEF    PRC_L_INDEPTH   .BLKL           ; INDIRECT STACK DEPTH (# LEVELS DEEP)
$DEF    PRC_L_SYMBOL    .BLKL           ; ADDRESS OF GOTO LABEL TABLE ENTRY
$DEF    PRC_W_INPCHAN   .BLKW           ; INPUT CHANNEL FOR INTERACTIVE JOBS
$DEF    PRC_W_ERRIFI    .BLKW           ; SYS$ERROR RMS IFI NUMBER

$EQU    PRC_V_CNTRLY    1               ; SUPERVISOR CONTROL Y/C LEVEL REQUEST
$EQU    PRC_M_CNTRLY    2
$EQU    PRC_V_DISABL    2               ; DISABLE CONTROL Y AST"S
$EQU    PRC_M_DISABL    4
$EQU    PRC_V_EXIT      3               ; EXIT HANDLER ESTABLISHED
$EQU    PRC_M_EXIT      8
$EQU    PRC_V_GOTO      4               ; FORWARD GOTO IN PROGRESS.
$EQU    PRC_M_GOTO      16
$EQU    PRC_V_IND       5               ; PROCESSING INDIRECT FILE
$EQU    PRC_M_IND       32
$EQU    PRC_V_MODE      6               ; COMMAND PROCESSING MODE (BATCH=1, INTER=0)
$EQU    PRC_M_MODE      64
$EQU    PRC_V_VERIFY    7               ; VERIFY LINES OF INDIRECT COMMAND FILES
$EQU    PRC_M_VERIFY    128
$EQU    PRC_V_AUTOLOGO  8               ; SILENT LOGOUT ($DELPRC) ON NEXT LEVEL 0 $GET
$EQU    PRC_M_AUTOLOGO  256
$EQU    PRC_V_DBGQUAL   9               ; DEBUG QUALIFIER SEEN ON COMMAND
$EQU    PRC_M_DBGQUAL   <^X200>
$EQU    PRC_V_DBGTRUE   10              ; STATE OF DEBUG WAS TRUE
$EQU    PRC_M_DBGTRUE   <^X400>
$EQU    PRC_V_YLEVEL    11              ; RUNNING AT CONTROL Y LEVEL
$EQU    PRC_M_YLEVEL    <^X800>
$EQU    PRC_V_HANGUP    12              ; TERMINAL HANGUP PENDING
$EQU    PRC_M_HANGUP    <^X1000>
$EQU    PRC_V_PAUSE     13              ; CLI PAUSE STATE ACTIVE
$EQU    PRC_M_PAUSE     <^X2000>
$EQU    PRC_V_EOFLOGO   14              ; SILENT LOGOUT ($DELPRC) ON LEVEL 0 EOF
```

## Listing 3-7: DCLDEF.MAR File (Page 2 of 3)

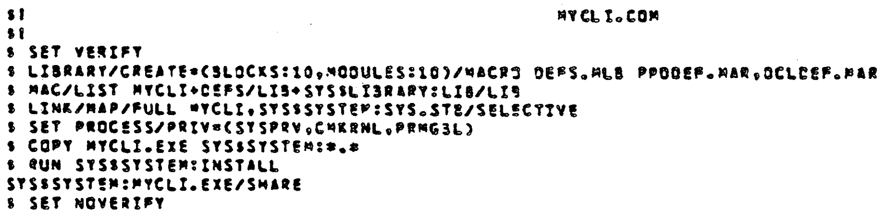
```
$EQU    PRC_M_EOFLOGO   <^X4000>
$EQU    PRC_V_DETACHED  15              ; TERMINAL IS DETACHED FROM THIS PROCESS
$EQU    PRC_M_DETACHED  <^X8000>
$DEF    PRC_W_FLAGS     .BLKW           ; PROCESS LEVEL FLAGS
$DEF    PRC_W_ONLEVEL   .BLKW           ; ON ERROR LEVEL NUMBER
$DEF    PRC_L_ONERROR   .BLKL           ; ADDRESS OF ON CONDITION COMMAND TEXT
$DEF    PRC_L_PPFLIST   .BLKL           ; LISTHEAD OF OPEN PPF RAB'S (VIA OPEN COMMAND)
                                ; BLOCKS; FOR EACH TERMINATION MAILBOX CREATED
$DEF    PRC_L_TMBX      .BLKL           ; LISTHEAD OF TERMINATION MAILBOX STORAGE

                        .BLKB   2
$DEF    PRC_W_ATTMBX    .BLKW           ; CHANNEL TO MAILBOX FOR RE-ATTACH REQUESTS
$DEF    PRC_L_INDCLOCK  .BLKL           ; TOTAL INDIRECT STACKS & UNSTACKS
$DEF    PRC_L_TAB_VEC   .BLKL           ; ADDRESS OF COMMAND TABLE VECTOR
$DEF    PRC_L_EXTBLK    .BLKL           ; EXIT HANDLER CONTROL BLOCK
$DEF    PRC_L_EXTHND    .BLKL           ; EXIT HANDLER ADDRESS
$DEF    PRC_L_EXTARG    .BLKL           ; NUMBER OF EXIT HANDLER ARGUMENTS
$DEF    PRC_L_EXTPRM    .BLKL           ; ADDRESS OF REASON FOR EXIT (BELOW)
$DEF    PRC_L_EXTCOD    .BLKL           ; REASON FOR EXIT
$DEF    PRC_L_STACKPT   .BLKL           ; INDIRECT STACK POINTER
$DEF    PRC_L_STACKLM   .BLKL           ; INDIRECT STACK LIMIT
$DEF    PRC_L_EXMDEPADR .BLKL           ; "DOT" FOR EXAMINE/DEPOSIT
$DEF    PRC_B_EXMDEPWID .BLKB           ; WIDTH DEFAULT, IE: BYTE,WORD,LONGWORD
$DEF    PRC_B_EXMDEPMOD .BLKB           ; MODE DEFAULT, IE: ASCII,HEX,OCTAL
                                        ; PROCESS RADIX TYPES
$EQU    PRC_K_HEX       0               ; HEXIDECIMAL
$EQU    PRC_K_DEC       1               ; DECIMAL
$EQU    PRC_K_OCT       2               ; OCTAL
$DEF    PRC_B_DEFRADIX  .BLKB           ; CURRENT DEFAULT RADIX
$EQU    PRC_V_CMD       0               ; COMMAND CALLBACK HAS BEEN DONE
$EQU    PRC_M_CMD       1
$EQU    PRC_V_CHAIN     1               ; CHAIN CALLBACK HAS BEEN DONE
$EQU    PRC_M_CHAIN     2
$EQU    PRC_V_RUNDEF    2               ; USE RUN DEFAULT (NOT EXTERNAL)
$EQU    PRC_M_RUNDEF    4
$EQU    PRC_V_EXEONLY   3               ; IMAGE IS EXECUTE ONLY
$EQU    PRC_M_EXEONLY   8
$EQU    PRC_V_PRIV      4               ; IMAGE IS PRIVILEGED
$EQU    PRC_M_PRIV      16
$EQU    PRC_V_ONEXIT    5               ; EXIT ALREADY PERFORMED
$EQU    PRC_M_ONEXIT    32
$DEF    PRC_B_FLAGS2    .BLKB           ; PROCESS LEVEL FLAGS (MORE OF THEM)
$DEF    PRC_L_LSTSTATUS .BLKL           ; LAST STATUS SET (LONGWORD VALUE)
                                                ; SKIP UNUSED CHARACTERS
$EQU    PRC_V_CTRLT     20              ; CONTROL T
$EQU    PRC_M_CTRLT     <^X100000>
                                                ; SKIP UNUSED CHARACTERS
$EQU    PRC_V_CTRLY     25              ; CONTROL Y
$EQU    PRC_M_CTRLY     <^X2000000>
$DEF    PRC_L_OUTOFBAND .BLKL           ; OUT OF BAND AST ENABLE BITMASK
$DEF    PRC_L_ONCTLY    .BLKL           ; ADDRESS OF ON CONTROL/Y COMMAND TEXT
$DEF    PRC_L_IDFLNK    .BLKL           ; POINTER TO INDIRECT FILE FRAMES
$DEF    PRC_L_SPWN      .BLKL           ; LISTHEAD OF SPAWN STORAGE BLOCKS FOR EACH
                                ; SUBPROCESS CREATED; SPAWN STORAGE FOR PROCESS.
                                ; CURRENTLY BEING SPAWNED IS FIRST, IF ANY.
                        .BLKL   3
$EQU    PRC_S_IMAGENAME 8
$DEF    PRC_Q_IMAGENAME .BLKQ           ; DESCRIPTOR OF CHAIN IMAGE FILE SPECIFICATION
$EQU    PRC_S_COMMAND   8
$DEF    PRC_Q_COMMAND   .BLKQ           ; DESCRIPTOR OF CHAIN COMMAND LINE FOR LATER
$DEF    PRC_C_LENGTH
```

Listing 3-7: DCLDEF.MAR File (Page 3 of 3)



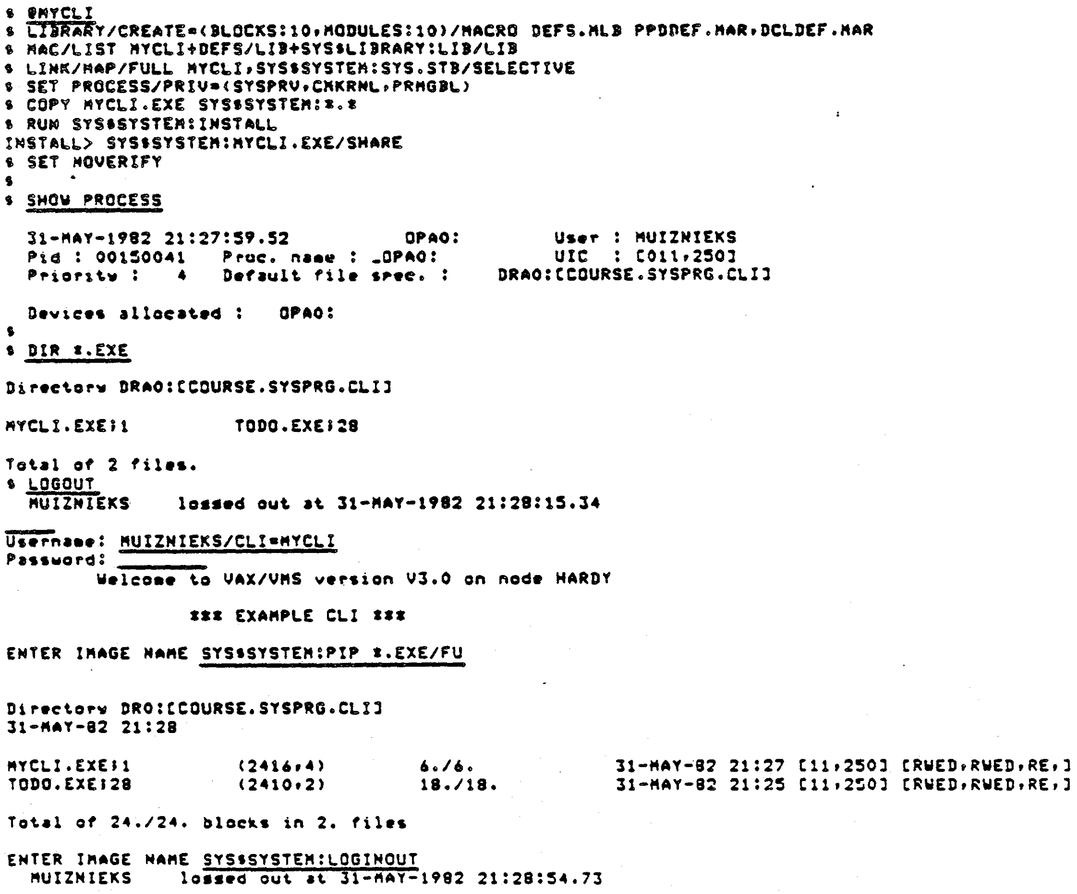
```
$DEF    PRC_K_LENGTH                            ; LENGTH OF PROCESS WORK AREA
        $DEFEND PRC,$GBL,DEF

        .ENDM   PRCDEF

;
;       DEFINE PROCESS RMS DATA AREA
;

        .MACRO  PRODEF,$GBL

        $DEFINI PRO,$GBL

$EQU    PRO_S_FAB         80
$DEF    PRO_G_FAB         .BLKB   80            ; PROCESS FAB
$EQU    PRO_S_NAM         96
$DEF    PRO_G_NAM         .BLKB   96            ; PROCESS NAME BLOCK
$EQU    PRO_S_INPRAB      68
$DEF    PRO_G_INPRAB      .BLKB   68            ; INPUT RAB
$EQU    PRO_S_ALTINPRAB 68
$DEF    PRO_G_ALTINPRAB .BLKB   68            ; ALTERNATE INPUT RAB
$EQU    PRO_S_ALTOUTRAB 68
$DEF    PRO_G_ALTOUTRAB .BLKB   68            ; ALTERNATE OUTPUT RAB
$EQU    PRO_S_OUTRAB      68
$DEF    PRO_G_OUTRAB      .BLKB   68            ; OUTPUT RAB
$DEF    PRO_C_LENGTH
; FOLLOWING EXTENSION USED ON LEVEL 0 TO STORE FID/DID/FNM FOR QUEUING
; THE JOB LOG FILE TO THE JOB CONTROLLER.
$DEF    PRO_K_LENGTH                            ; NORMAL LENGTH
$EQU    PRO_S_OUTDVI      16
$DEF    PRO_T_OUTDVI      .BLKB   16            ; DEVICE FOR OUTPUT FILE
$EQU    PRO_S_OUTFID      6
$DEF    PRO_W_OUTFID      .BLKW   3             ; FILE ID FOR OUTPUT FILE
$EQU    PRO_S_OUTDID      6
$DEF    PRO_W_OUTDID      .BLKW   3             ; DIRECTORY ID FOR OUTPUT FILE
$EQU    PRO_S_OUTFNM      20
$DEF    PRO_T_OUTFNM      .BLKB   20            ; OUTPUT FILE NAME
$DEF    PRO_C_XLENGTH

$DEF    PRO_K_XLENGTH                           ; LENGTH OF EXTENDED BLOCK
        $DEFEND PRO,$GBL,DEF

        .ENDM   PRODEF
```

## Listing 3-8: MYCLI.COM File



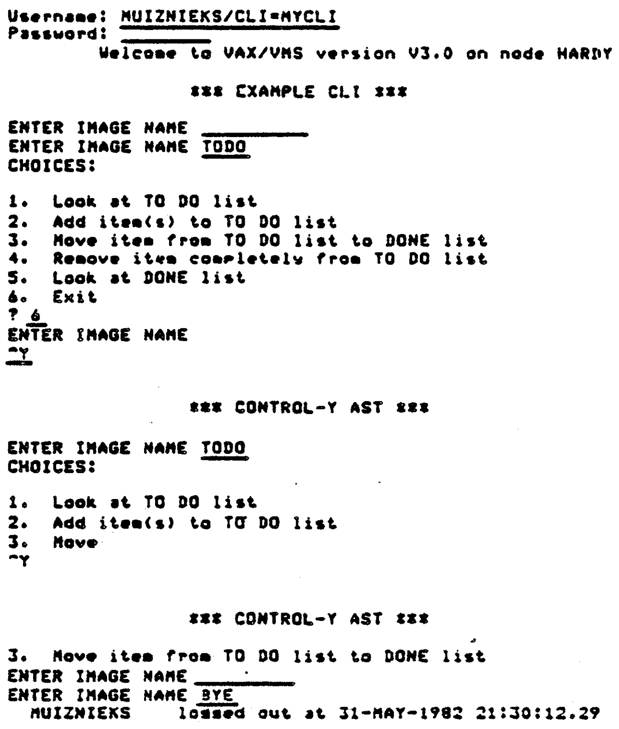
```
$!                                                          MYCLI.COM
$!
$ SET VERIFY
$ LIBRARY/CREATE=(BLOCKS:10,MODULES:10)/MACRO DEFS.MLB PPDDEF.MAR,DCLDEF.MAR
$ MAC/LIST MYCLI+DEFS/LIB+SYS$LIBRARY:LIB/LIB
$ LINK/MAP/FULL MYCLI,SYS$SYSTEM:SYS.STB/SELECTIVE
$ SET PROCESS/PRIV=(SYSPRV,CMKRNL,PRMGBL)
$ COPY MYCLI.EXE SYS$SYSTEM:*.*
$ RUN SYS$SYSTEM:INSTALL
SYS$SYSTEM:MYCLI.EXE/SHARE
$ SET NOVERIFY
```

## Listing 3-9: Sample Run



```
$ @MYCLI
$ LIBRARY/CREATE=(BLOCKS:10,MODULES:10)/MACRO DEFS.MLB PPDDEF.MAR,DCLDEF.MAR
$ MAC/LIST MYCLI+DEFS/LIB+SYS$LIBRARY:LIB/LIB
$ LINK/MAP/FULL MYCLI,SYS$SYSTEM:SYS.STB/SELECTIVE
$ SET PROCESS/PRIV=(SYSPRV,CMKRNL,PRMGBL)
$ COPY MYCLI.EXE SYS$SYSTEM:*.*
$ RUN SYS$SYSTEM:INSTALL
INSTALL> SYS$SYSTEM:MYCLI.EXE/SHARE
$ SET NOVERIFY
$
$ SHOW PROCESS

  31-MAY-1982 21:27:59.52          OPA0:          User : MUIZNIEKS
  Pid : 00150041   Proc. name : _OPA0:            UIC  : [011,250]
  Priority :   4   Default file spec. :    DRA0:[COURSE.SYSPRG.CLI]

  Devices allocated :   OPA0:
$
$ DIR *.EXE

Directory DRA0:[COURSE.SYSPRG.CLI]

MYCLI.EXE;1          TODO.EXE;28

Total of 2 files.
$ LOGOUT
  MUIZNIEKS    logged out at 31-MAY-1982 21:28:15.34

Username: MUIZNIEKS/CLI=MYCLI
Password:
        Welcome to VAX/VMS version V3.0 on node HARDY

                ** EXAMPLE CLI ***

ENTER IMAGE NAME SYS$SYSTEM:PIP *.EXE/FU


Directory DR0:[COURSE.SYSPRG.CLI]
31-MAY-82 21:28

MYCLI.EXE;1          (2416,4)        6./6.            31-MAY-82 21:27 [11,250] [RWED,RWED,RE,]
TODO.EXE;28          (2410,2)        18./18.          31-MAY-82 21:25 [11,250] [RWED,RWED,RE,]

Total of 24./24. blocks in 2. files

ENTER IMAGE NAME SYS$SYSTEM:LOGINOUT
  MUIZNIEKS    logged out at 31-MAY-1982 21:28:54.73
```

## Listing 3-1Ø: Second Sample Run



```
Username: MUIZNIEKS/CLI=MYCLI
Password:
        Welcome to VAX/VMS version V3.0 on node HARDY

                    *** EXAMPLE CLI ***

ENTER IMAGE NAME
ENTER IMAGE NAME TODO
CHOICES:

1.  Look at TO DO list
2.  Add item(s) to TO DO list
3.  Move item from TO DO list to DONE list
4.  Remove item completely from TO DO list
5.  Look at DONE list
6.  Exit
? 6
ENTER IMAGE NAME
^Y

                    *** CONTROL-Y AST ***

ENTER IMAGE NAME TODO
CHOICES:

1.  Look at TO DO list
2.  Add item(s) to TO DO list
3.  Move
^Y

                    *** CONTROL-Y AST ***

3.  Move item from TO DO list to DONE list
ENTER IMAGE NAME
ENTER IMAGE NAME BYE
  MUIZNIEKS     logged out at 31-MAY-1982 21:30:12.29
```

# WRITING A SYMBIONT

# WRITING A SYMBIONT

## INTRODUCTION

Symbiosis is a term normally used to describe a relationship that is beneficial to both entities. The entities (the Job Controller and the symbionts) are performing functions that are beneficial to each other.

The Job Controller performs several related functions:

Management of interactive processes

Management of batch queues and jobs

Management of symbionts

Management of the accounting file

The symbionts (input and output) perform the detailed operations of printing a file or reading in cards to be placed in the batch queue. because the symbionts are subprocesses of the Job Controller, they are created and deleted by the Job Controller. The symbionts also communicate with the the Job Controller via mailboxes. This means that the symbionts must adhere to strict rules of communication in sending to and recieving information from the Job Controller.

Due to the degree of interdependance the symbionts and the Job Controller have with other, both will be discussed in this module.

## OBJECTIVES

- Describe the general contents of messages exchanged between the job controller and symbionts.

- Describe the implementation of the Job Controller's queue file, and discuss the implications of its implementation as a global section.

- Write a symbiont that communicates with the Job Controller.

## RESOURCES

JOB CONTROLLER SOURCE LISTINGS
PRINT SYMBIONT SOURCE LISTINGS
INTERNALS/DATA STRUCTURES MANUAL

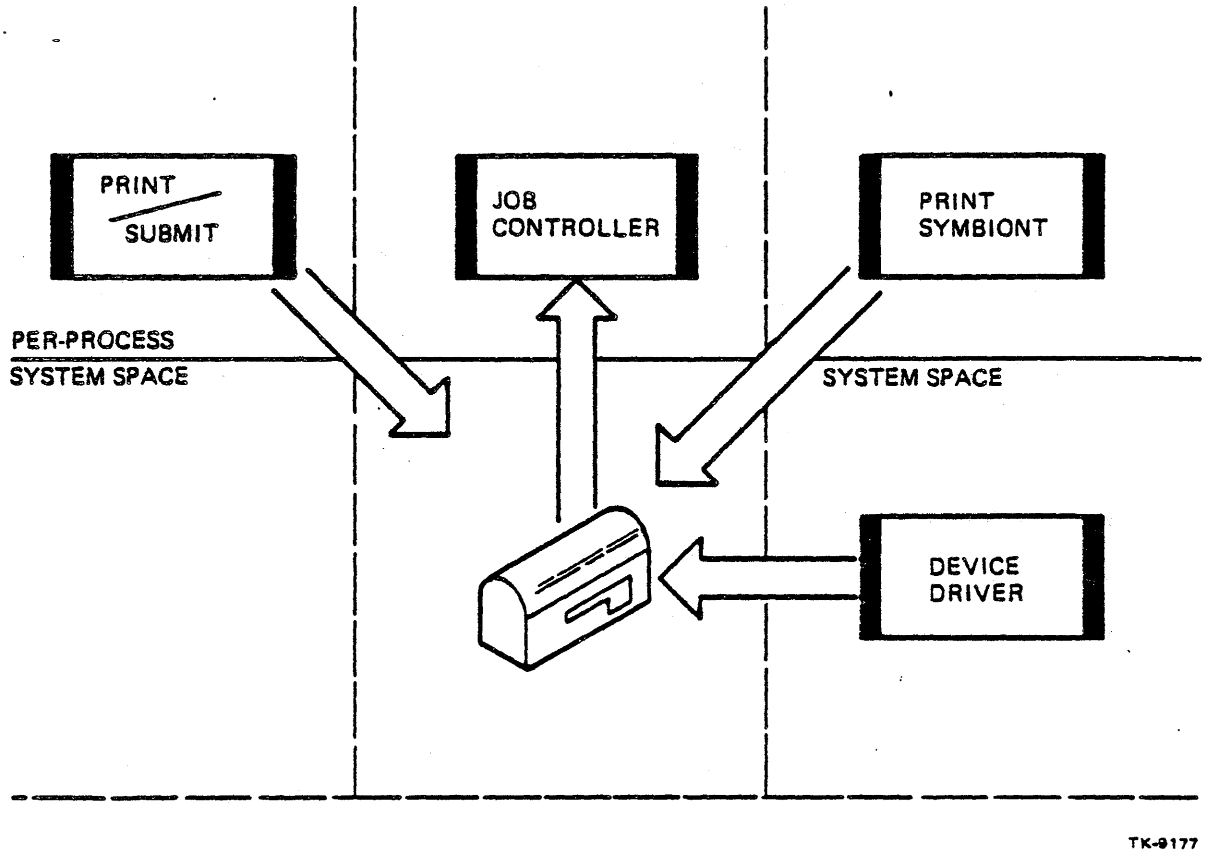


Figure 4-1 COMMUNICATION TO JOB CONTROLLER

- JOB CONTROLLER is a full process
  - event driven
  - responds to information placed in mailbox
  - outstanding $QIO on mailbox

- Mailbox communication with
  - User processes
  - Card readers
  - Symbionts

## JOB CONTROLLER FUNCTIONS

1. Interactive Jobs

   a. Creation
      Responds to unsolicited input message
      Process created running LOGINOUT.EXE

   b. Activities
      Responds to messages from CLI (i.e. PRINT)

   c. Deletion
      Records accounting information

2. Batch Jobs

   a. Creation
      Responds to unsolicited input message
      Process created running INPSMB.EXE

   b. Activities
      Same as for interactive jobs

   c. Deletion
      Same as for interactive jobs

3. Symbiont Manager

   a. Creation
      Symbionts created via operator action

   b. Activities
      Mailboxes messages sent to Symbiont assign jobs to print. Symbionts do not see queue

   c. Deletion
      Symbionts deleted via operator action

4. Accounting Manager

   a. Activities
      - Interactive or batch job termination
      - Print job completion
      - Login failure

   b. Additional DCL commands ($SET ACCOUNTING) invoke the Accounting Manager

## Programmer Interaction with JOB CONTROLLER

| Function | Method |
|---|---|
| Send information to the Accounting Manager | $SNDACC |
| Send information to the Symbiont Manager | $SNDSMB |
| Send a file to be printed | $PRINT |
| Initialize and control of the queues | $INIT/QUEUE<br>$START/QUEUE<br>$STOP/QUEUE<br>$DELETE/QUEUE |

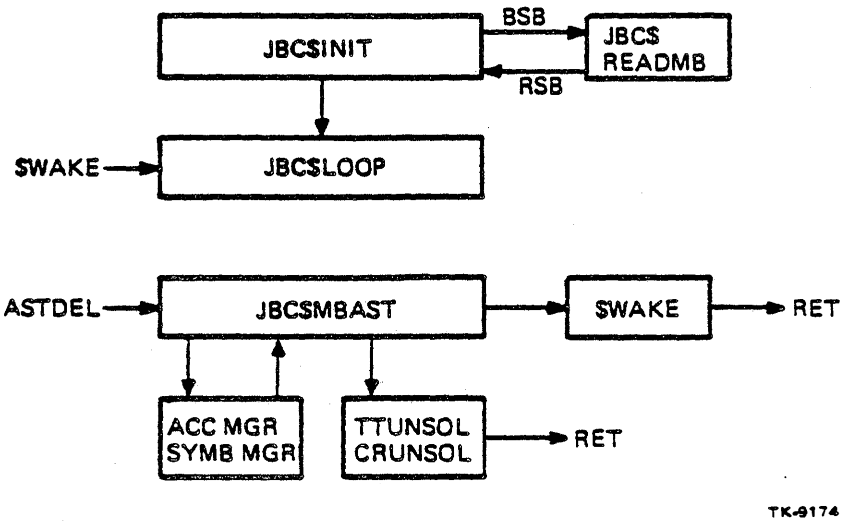


Figure 4-2 Job Controller Code Flow

- Initialization
- Main Routine Loop
- Mailbox AST
  - if unsolicited TTY or CR, $CREPRC
  - else issue $WAKE

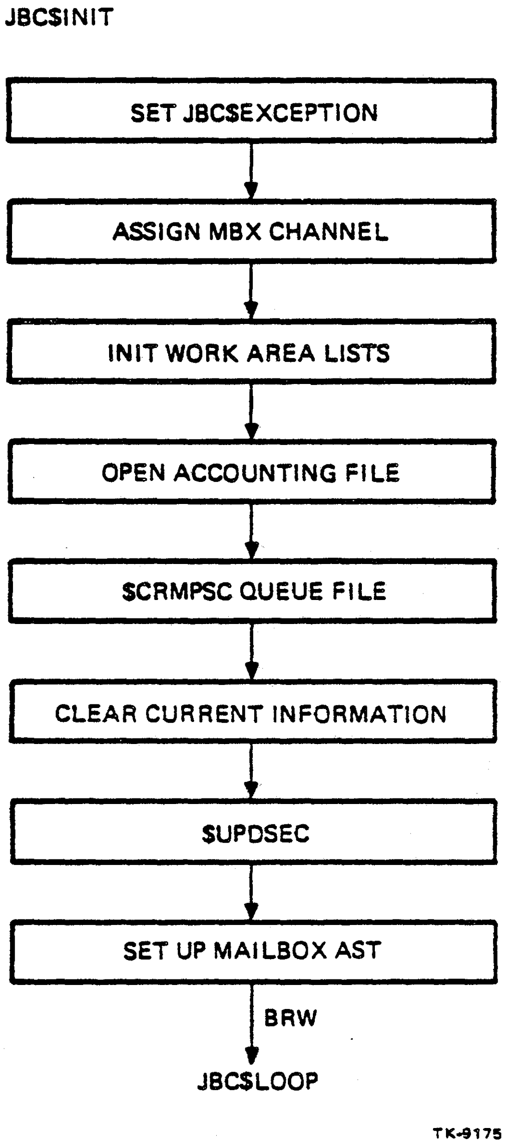
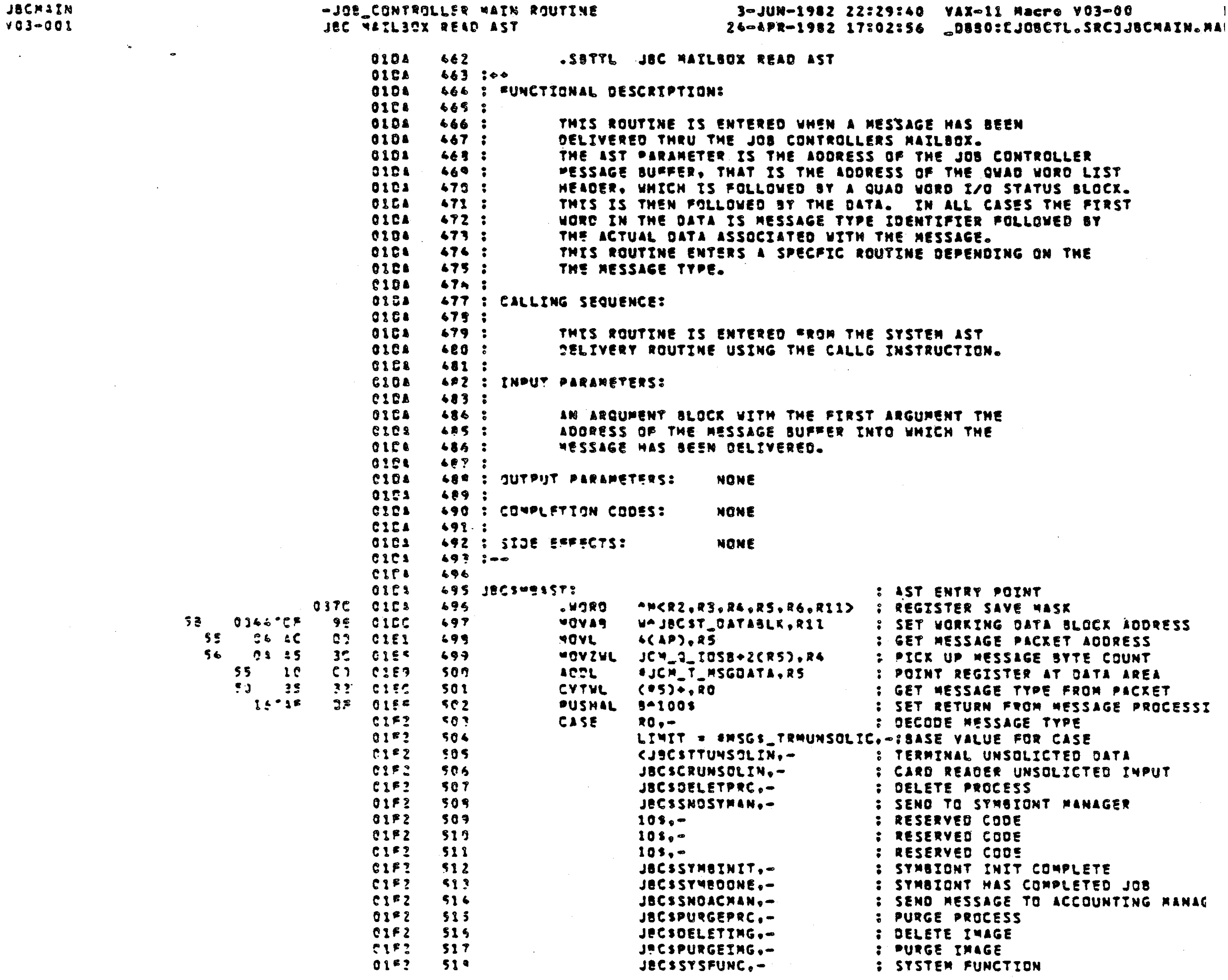


Figure 4-3 Job Controller Initialization Flow

Listing 4-1 Mailbox Ast Code (page 1 of 7)

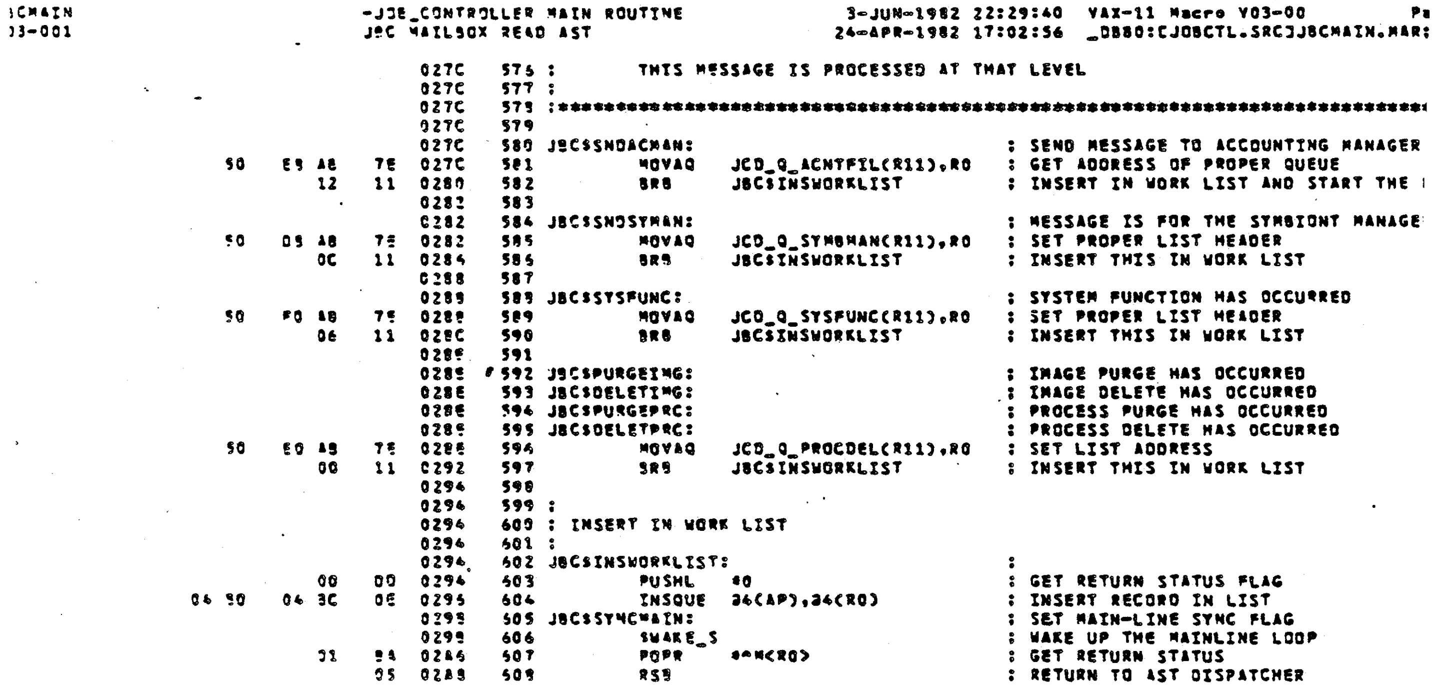


```
JBCMAIN                    -JOB_CONTROLLER MAIN ROUTINE          3-JUN-1982 22:29:40  VAX-11 Macro V03-00
V03-001                    JBC MAILBOX READ AST                 26-APR-1982 17:02:56  _DBB0:[JOBCTL.SRC]JBCMAIN.MA

                   01DA             .SBTTL  JBC MAILBOX READ AST
                   01DA       ;++
                   01DA       ; FUNCTIONAL DESCRIPTION:
                   01DA       ;
                   01DA       ;       THIS ROUTINE IS ENTERED WHEN A MESSAGE HAS BEEN
                   01DA       ;       DELIVERED THRU THE JOB CONTROLLERS MAILBOX.
                   01DA       ;       THE AST PARAMETER IS THE ADDRESS OF THE JOB CONTROLLER
                   01DA       ;       MESSAGE BUFFER, THAT IS THE ADDRESS OF THE QUAD WORD LIST
                   01DA       ;       HEADER, WHICH IS FOLLOWED BY A QUAD WORD I/O STATUS BLOCK.
                   01DA       ;       THIS IS THEN FOLLOWED BY THE DATA.  IN ALL CASES THE FIRST
                   01DA       ;       WORD IN THE DATA IS MESSAGE TYPE IDENTIFIER FOLLOWED BY
                   01DA       ;       THE ACTUAL DATA ASSOCIATED WITH THE MESSAGE.
                   01DA       ;       THIS ROUTINE ENTERS A SPECFIC ROUTINE DEPENDING ON THE
                   01DA       ;       THE MESSAGE TYPE.
                   01DA       ;
                   01DA       ; CALLING SEQUENCE:
                   01DA       ;
                   01DA       ;       THIS ROUTINE IS ENTERED FROM THE SYSTEM AST
                   01DA       ;       DELIVERY ROUTINE USING THE CALLG INSTRUCTION.
                   01DA       ;
                   01DA       ; INPUT PARAMETERS:
                   01DA       ;
                   01DA       ;       AN ARGUMENT BLOCK WITH THE FIRST ARGUMENT THE
                   01DA       ;       ADDRESS OF THE MESSAGE BUFFER INTO WHICH THE
                   01DA       ;       MESSAGE HAS BEEN DELIVERED.
                   01DA       ;
                   01DA       ; OUTPUT PARAMETERS:    NONE
                   01DA       ;
                   01DA       ; COMPLETION CODES:     NONE
                   01DA       ;
                   01DA       ; SIDE EFFECTS:         NONE
                   01DA       ;--
                   01DA
                   01DA       JBC$MBXAST:                                  ; AST ENTRY POINT
              037C 01DA             .WORD   ^M<R2,R3,R4,R5,R6,R11>     ; REGISTER SAVE MASK
5B  0344'CF     9E 01DC             MOVAB   W^JBC$T_DATABLK,R11        ; SET WORKING DATA BLOCK ADDRESS
 55   04 AC     D0 01E1             MOVL    4(AP),R5                   ; GET MESSAGE PACKET ADDRESS
 56   08 A5     3C 01E5             MOVZWL  JCM_Q_IOSB+2(R5),R4        ; PICK UP MESSAGE BYTE COUNT
   55    10     C0 01E9             ADDL    #JCM_T_MSGDATA,R5          ; POINT REGISTER AT DATA AREA
   50    85     32 01EC             CVTWL   (R5)+,R0                   ; GET MESSAGE TYPE FROM PACKET
      16'AF     9F 01EF             PUSHAL  B^100$                     ; SET RETURN FROM MESSAGE PROCESSI
                   01F2             CASE    R0,-                       ; DECODE MESSAGE TYPE
                   01F2                     LIMIT = #MSG$_TRMUNSOLIC,-;BASE VALUE FOR CASE
                   01F2                     <JBC$TTUNSOLIN,-           ; TERMINAL UNSOLICTED DATA
                   01F2                     JBC$CRUNSOLIN,-            ; CARD READER UNSOLICTED INPUT
                   01F2                     JBC$DELETPRC,-             ; DELETE PROCESS
                   01F2                     JBC$SNDSYMAN,-             ; SEND TO SYMBIONT MANAGER
                   01F2                     10$,-                      ; RESERVED CODE
                   01F2                     10$,-                      ; RESERVED CODE
                   01F2                     10$,-                      ; RESERVED CODE
                   01F2                     JBC$SYMBINIT,-             ; SYMBIONT INIT COMPLETE
                   01F2                     JBC$SYMBDONE,-             ; SYMBIONT HAS COMPLETED JOB
                   01F2                     JBC$SNDACMAN,-             ; SEND MESSAGE TO ACCOUNTING MANAG
                   01F2                     JBC$PURGEPRC,-             ; PURGE PROCESS
                   01F2                     JBC$DELETING,-             ; DELETE IMAGE
                   01F2                     JBC$PURGEING,-             ; PURGE IMAGE
                   01F2                     JBC$SYSFUNC,-              ; SYSTEM FUNCTION
```

Listing 4-1 Mailbox Ast Code (page 2 of 7)

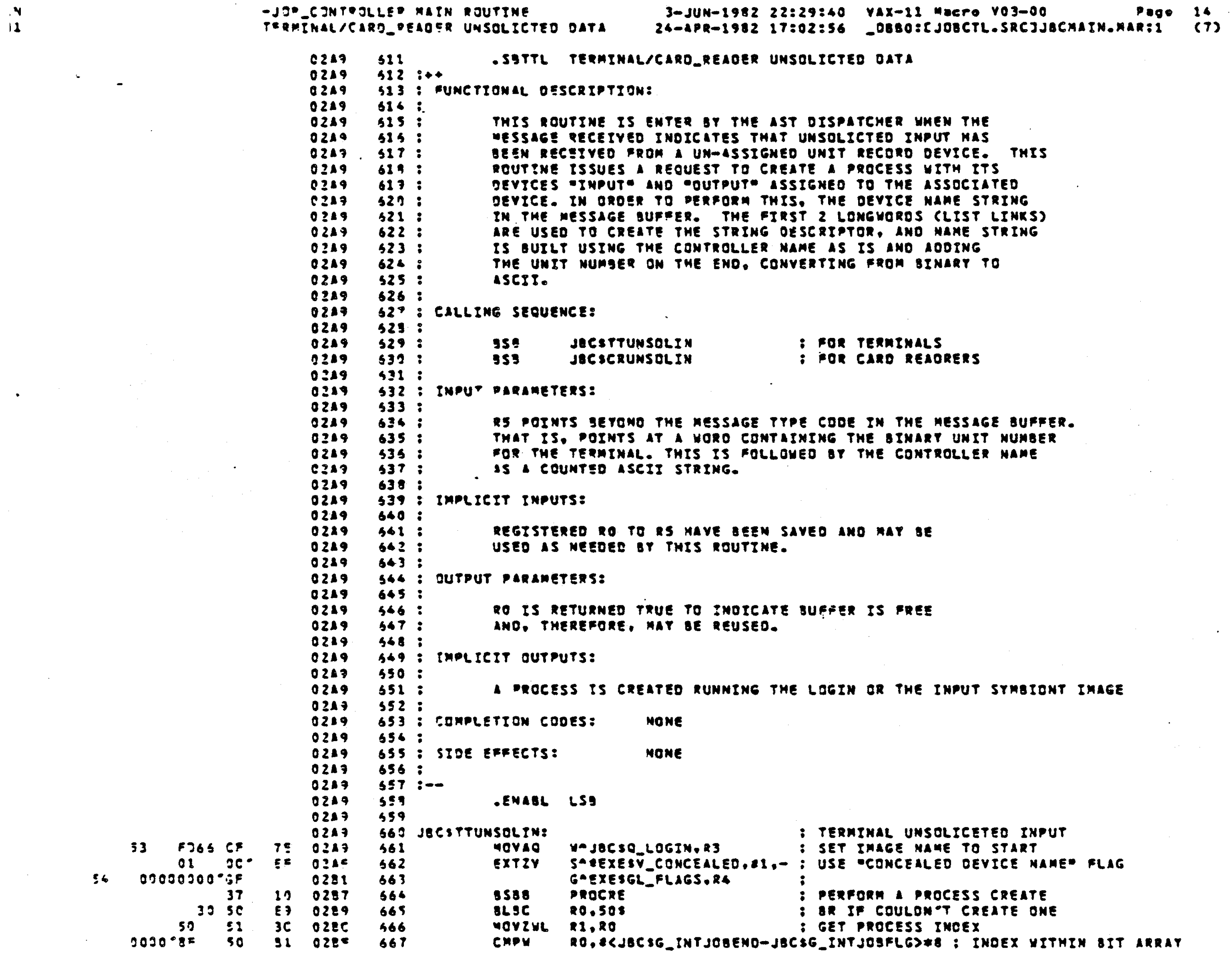
```
IN                  -JOB_CONTROLLER MAIN ROUTINE        3-JUN-1982 22:29:40  VAX-11 Macro V03-00        Page  12
01                  JBC MAILBOX READ AST               24-APR-1982 17:02:56  _DBB0:[JOBCTL.SRC]JBCMAIN.MAR;1    (6)

                        01F2               >
                        0212       10$:    SHOW_ERROR      INVALID_MESG    ; BAD MESSAGE RECEIVED
               03   11  0214               BRB     110$                    ; DEALLOCATE THE MESSAGE BUFFER
            05 50   E9  0216       100$:   BLBC    R0,120$                 ; BR IF MESSAGE BUFFER STILL IN USE
      FC BB  04 BC  0E  0219       110$:   INSQUE  @4(AP),@JCD_Q_FREEBUFR+4(R11) ; RELEASE CURRENT BUFFER IF FREE
             FF7C   30  021E       120$:   BSBW    JBC$READMB              ; RESTART I/O ON MAILBOX
                    04  0221               RET                             ; DISMISS AST
                        0222
                        0222       ;
                        0222       ; MESSAGE IS SYMBIONT DONE
                        0222       ;
                        0222       ;       ADJUST STATE FOR SYMBIONT TO READY FOR NEXT FILE
                        0222       ;       INSERT ITS TABLE IN SEVICE LIST
                        0222       ;       SET THE MAIN LINE SYNC FLAG
                        0222       ;
                        0222       JBC$SYMBDONE:                           ; SYMBIONT IS COMPLETE
               55   DD  0222               PUSHL   R5                      ; SAVE MESSAGE BUFFER ADDRESS
      51    FA A5   D0  0224               MOVL    <JCM_Q_IOSB+4-<JCM_T_MSGDATA+2>>(R5),R1 ; GET SENDER'S ID
            FCC5'   30  0228               BSBW    SYM$FINDSYMCTL          ; LOCATE THE SYMBIONT CONTROL TABLE
               02   BA  022B               POPR    #^M<R1>                 ; POP BUFFER ADDRESS TO WORK REGISTER
      08 A5    07   91  022D               CMPB    #SCT_K_SUSPND,SCT_B_STATE(R5) ; Suspend issued?
               04   12  0231               BNEQ    5$                      ; Br if no
      08 A5    03   90  0233               MOVB    #SCT_K_DEQFIL,SCT_B_STATE(R5) ; Mark as idle
         50    81   3C  0237       5$:     MOVZWL  (R1)+,R0                ; Pick up job status
            04 50   E8  023A               BLBS    R0,6$                   ; Br if successful
      16 A5    50   B0  023D               MOVW    R0,SCT_W_JOBSTAT(R5)    ; Save error status for mainline
      24 A5    81   C0  0241       6$:     ADDL    (R1)+,SCT_L_GETCNT(R5)  ; ACCUMULATE ACCOUNTING
      20 A5    81   C0  0245               ADDL    (R1)+,SCT_L_QIOCNT(R5)  ; INFORMATION ON GET'S ,QIO'S
         50    81   3C  0249               MOVZWL  (R1)+,R0                ; GET PAGES AS A LONG WORD
      1C A5    50   C0  024C               ADDL    R0,SCT_L_PAGCNT(R5)     ; AND ACCOUNT FOR PAGES
                        0250
      56    14 A5   3C  0250               MOVZWL  SCT_W_QINDEX(R5),R6     ; POINT R6 AT QUEUE HEADER
  56   000004AE'EF  C0  0254               ADDL    JBC$Q_RETADR,R6                ; GET ACTUAL ADDR
   05 08 A5    00   E1  025B               BBC     #SMQ$V_STOPPED,SMQ$B_FLAGS(R6),10$      ; BR QUE NOT STOPPED
               50   D4  0260               CLRL    R0                      ; SET TO DEALLOCATE DEVICE
            FD93'   30  0262               BSBW    SYM$ALLODEAL            ; GO DEALLOCATE DEVICE
               0D   11  0265       10$:    BRB     SYMBEXIT                ; EXIT SYMBIONT SERVICE AST
                        0267       ;
                        0267       ; MESSAGE IS A SYMBIONT HAS INITED.
                        0267       ;       SAVE THE MAILBOX UNIT IN THE SYMBIONT CONTROL TABLE
                        0267       ;       AND INSEPT TABLE IN SYMBIONT SERVICE LIST
                        0267       ;       SET THE SYNCRONIZING EVENT FLAG TO GET THE MAIN STARTED.
                        0267       ;
                        0267       JBC$SYMBINIT:                           ;
               65   DD  0267               PUSHL   (R5)                    ; SAVE MAILBOX UNIT NUMBER
      51    FA A5   D0  0269               MOVL    <JCM_Q_IOSB+4-<JCM_T_MSGDATA+2>>(R5),R1 ; PICK UP SENDER ID
            FC80'   30  026D               BSBW    SYM$FINDSYMCTL          ; LOOK FOR SYMBIONT CONTROL TABLE
      3A A5    8E   F7  0270               CVTLW   (SP)+,SCT_W_MBCHAN(R5)  ; STORE MAILBOX UNIT IN CONTROL TABLE
                        0274       SYMBEXIT:                               ; COMMON EXIT FOR SYMBIONT SERVICE
      04 BB    65   0E  0274               INSQUE  (R5),@JCD_Q_SYMBSRV+4(R11) ; PUT THIS SYMBIONT IN SERVICE QUEUE
               01   DD  0278               PUSHL   #1                      ; SET STATUS TO AST DISPATCHER
               1F   11  027A               BRB     JBC$SYNCMAIN            ; SYNCRONIZE WITH MAINLINE
                        027C
                        027C       ;*******************************************************************************
                        027C       ;
                        027C       ;       INSERT MESSAGES IN THE PROPER QUEUE
                        027C       ;       AND SET THE SYNCRONIZING FLAG FOR THE MAIN LINE
```

Listing 4-1 Mailbox Ast Code (page 3 of 7)



```
JCMAIN                            -JOE_CONTROLLER MAIN ROUTINE           3-JUN-1982 22:29:40  VAX-11 Macro V03-00         Pa
03-001                            JBC MAILBOX READ AST                  24-APR-1982 17:02:56  _DB80:[JOBCTL.SRC]JBCMAIN.MAR;

                                  027C    576 ;       THIS MESSAGE IS PROCESSED AT THAT LEVEL
                                  027C    577 ;
                                  027C    578 ;*****************************************************************************
                                  027C    579
                                  027C    580 JBC$SNDACMAN:                                  ; SEND MESSAGE TO ACCOUNTING MANAGER
               50   E8 A8   7E    027C    581         MOVAQ   JCD_Q_ACNTFIL(R11),R0         ; GET ADDRESS OF PROPER QUEUE
                       12   11    0280    582         BRB     JBC$INSWORKLIST               ; INSERT IN WORK LIST AND START THE
                                  0282    583
                                  0282    584 JBC$SNDSYMAN:                                  ; MESSAGE IS FOR THE SYMBIONT MANAGE
               50   D8 A8   7E    0282    585         MOVAQ   JCD_Q_SYMBMAN(R11),R0         ; SET PROPER LIST HEADER
                       0C   11    0286    586         BRB     JBC$INSWORKLIST               ; INSERT THIS IN WORK LIST
                                  0288    587
                                  0288    588 JBC$SYSFUNC:                                   ; SYSTEM FUNCTION HAS OCCURRED
               50   F0 A8   7E    0288    589         MOVAQ   JCD_Q_SYSFUNC(R11),R0         ; SET PROPER LIST HEADER
                       06   11    028C    590         BRB     JBC$INSWORKLIST               ; INSERT THIS IN WORK LIST
                                  028E    591
                                  028E    592 JBC$PURGEIMG:                                  ; IMAGE PURGE HAS OCCURRED
                                  028E    593 JBC$DELETIMG:                                  ; IMAGE DELETE HAS OCCURRED
                                  028E    594 JBC$PURGEPRC:                                  ; PROCESS PURGE HAS OCCURRED
                                  028E    595 JBC$DELETPRC:                                  ; PROCESS DELETE HAS OCCURRED
               50   E0 A8   7E    028E    596         MOVAQ   JCD_Q_PROCDEL(R11),R0         ; SET LIST ADDRESS
                       00   11    0292    597         BRB     JBC$INSWORKLIST               ; INSERT THIS IN WORK LIST
                                  0294    598
                                  0294    599 ;
                                  0294    600 ; INSERT IN WORK LIST
                                  0294    601 ;
                                  0294    602 JBC$INSWORKLIST:                               ;
                       00   DD    0294    603         PUSHL   #0                            ; GET RETURN STATUS FLAG
          04 50     04 3C   0E    0296    604         INSQUE  @4(AP),@4(R0)                 ; INSERT RECORD IN LIST
                                  0298    605 JBC$SYNCMAIN:                                  ; SET MAIN-LINE SYNC FLAG
                                  0298    606         $WAKE_S                               ; WAKE UP THE MAINLINE LOOP
                       01   BA    02A6    607         POPR    #^M<R0>                       ; GET RETURN STATUS
                            05    02A8    608         RSB                                   ; RETURN TO AST DISPATCHER
```

## Listing 4-1 Mailbox Ast Code (page 4 of 7)

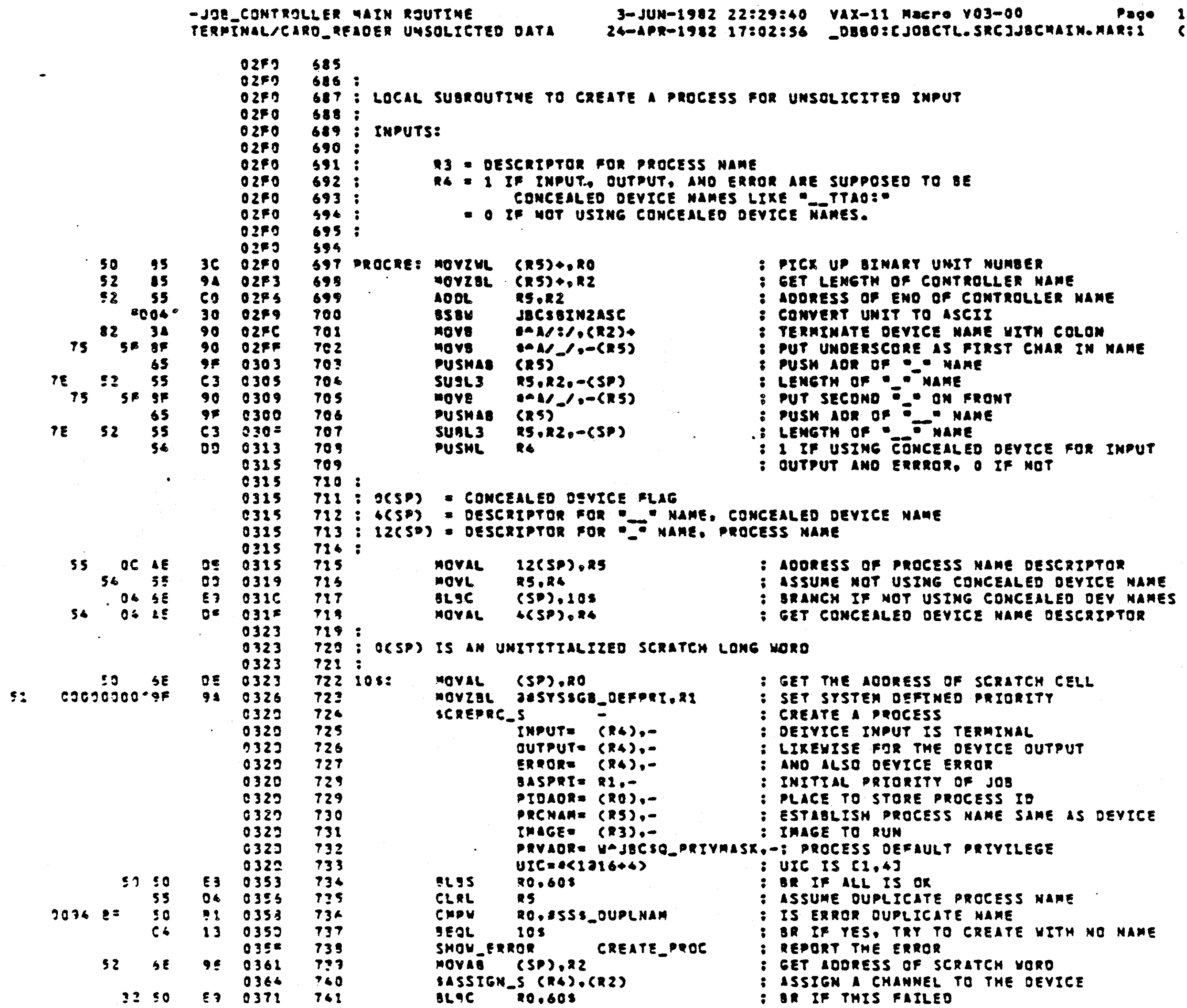


```
.N                      -JOB_CONTROLLER MAIN ROUTINE            3-JUN-1982 22:29:40  VAX-11 Macro V03-00          Page  14
11                      TERMINAL/CARD_READER UNSOLICTED DATA    24-APR-1982 17:02:56  _DBB0:[JOBCTL.SRC]JBCMAIN.MAR;1   (7)

                        02A9          .SBTTL  TERMINAL/CARD_READER UNSOLICTED DATA
                        02A9     ;++
                        02A9     ; FUNCTIONAL DESCRIPTION:
                        02A9     ;
                        02A9     ;        THIS ROUTINE IS ENTER BY THE AST DISPATCHER WHEN THE
                        02A9     ;        MESSAGE RECEIVED INDICATES THAT UNSOLICTED INPUT HAS
                        02A9     ;        BEEN RECEIVED FROM A UN-ASSIGNED UNIT RECORD DEVICE.  THIS
                        02A9     ;        ROUTINE ISSUES A REQUEST TO CREATE A PROCESS WITH ITS
                        02A9     ;        DEVICES "INPUT" AND "OUTPUT" ASSIGNED TO THE ASSOCIATED
                        02A9     ;        DEVICE. IN ORDER TO PERFORM THIS, THE DEVICE NAME STRING
                        02A9     ;        IN THE MESSAGE BUFFER.  THE FIRST 2 LONGWORDS (LIST LINKS)
                        02A9     ;        ARE USED TO CREATE THE STRING DESCRIPTOR, AND NAME STRING
                        02A9     ;        IS BUILT USING THE CONTROLLER NAME AS IS AND ADDING
                        02A9     ;        THE UNIT NUMBER ON THE END, CONVERTING FROM BINARY TO
                        02A9     ;        ASCII.
                        02A9     ;
                        02A9     ; CALLING SEQUENCE:
                        02A9     ;
                        02A9     ;        BSB     JBC$TTUNSOLIN            ; FOR TERMINALS
                        02A9     ;        BSB     JBC$CRUNSOLIN            ; FOR CARD READERS
                        02A9     ;
                        02A9     ; INPUT PARAMETERS:
                        02A9     ;
                        02A9     ;        R5 POINTS BEYOND THE MESSAGE TYPE CODE IN THE MESSAGE BUFFER.
                        02A9     ;        THAT IS, POINTS AT A WORD CONTAINING THE BINARY UNIT NUMBER
                        02A9     ;        FOR THE TERMINAL. THIS IS FOLLOWED BY THE CONTROLLER NAME
                        02A9     ;        AS A COUNTED ASCII STRING.
                        02A9     ;
                        02A9     ; IMPLICIT INPUTS:
                        02A9     ;
                        02A9     ;        REGISTERED R0 TO R5 HAVE BEEN SAVED AND MAY BE
                        02A9     ;        USED AS NEEDED BY THIS ROUTINE.
                        02A9     ;
                        02A9     ; OUTPUT PARAMETERS:
                        02A9     ;
                        02A9     ;        R0 IS RETURNED TRUE TO INDICATE BUFFER IS FREE
                        02A9     ;        AND, THEREFORE, MAY BE REUSED.
                        02A9     ;
                        02A9     ; IMPLICIT OUTPUTS:
                        02A9     ;
                        02A9     ;        A PROCESS IS CREATED RUNNING THE LOGIN OR THE INPUT SYMBIONT IMAGE
                        02A9     ;
                        02A9     ; COMPLETION CODES:      NONE
                        02A9     ;
                        02A9     ; SIDE EFFECTS:          NONE
                        02A9     ;
                        02A9     ;--
                        02A9          .ENABL  LSB
                        02A9
                        02A9     JBC$TTUNSOLIN:                            ; TERMINAL UNSOLICETED INPUT
  53   F366 CF    7E    02A9          MOVAQ   W^JBC$Q_LOGIN,R3          ; SET IMAGE NAME TO START
    01   0C'      EF    02AE          EXTZV   S^#EXE$V_CONCEALED,#1,-   ; USE "CONCEALED DEVICE NAME" FLAG
54  00000000'GF         02B1                  G^EXE$GL_FLAGS,R4         ;
           37     10    02B7          BSBB    PROCRE                    ; PERFORM A PROCESS CREATE
        30 50     E9    02B9          BLBC    R0,50$                    ; BR IF COULDN'T CREATE ONE
      50   51     3C    02BC          MOVZWL  R1,R0                     ; GET PROCESS INDEX
 0000'8F   50     B1    02BF          CMPW    R0,#<JBC$G_INTJOBEND-JBC$G_INTJOBFLG>*8 ; INDEX WITHIN BIT ARRAY
```

Listing 4-1 Mailbox Ast Code (page 5 of 7)

```
JBCMAIN                                   -JOB_CONTROLLER MAIN ROUTINE          3-JUN-1982 22:29:40  VAX-11 Macro V03-00
V03-001                                   TERMINAL/CARD_READER UNSOLICTED DATA  24-APR-1982 17:02:56  _DBB0:[JOBCTL.SRC]JBCMAIN.M

                          26    15  02C4                 BGEQU   50$                        ; IF NO - DON'T COUNT INTERACTIVE .
             20 0000'CF   50    E2  02C6                 BBSS    R0,W^JBC$G_INTJOBFLG,50$: SET FLAG AND BR IF SET, DON,T BU
                                    02CC                 $CMKRNL_S B^20$                    ; ADD 1 TO COUNT OF INTERACTIVE JO
                          12    11  02D8                 BRB     50$                        ;
                              0000  02DA        20$:     .WORD   0                          ; KERNAL ACCESS MODE ENTRY MASK
               00000000'9F      B6  02DC                 INCW    @#SYS$GW_IJOBCNT           ; COUNT NUMBER OF INTERACTIVE JOBS
                                04  02E2                 RET                                ;
                                    02E3
                                    02E3        JBC$CRUNSOLIN:                              ; CARD READER UNSOLICETED INPUT
        53      F343 CF         7E  02E3                 MOVAQ   JBC$Q_INPSMB,R3            ; SET ADDRESS OF IMAGE TO ACTIVATE
                          54    D4  02E8                 CLRL    R4                         ; DON'T USE CONCEALED DEVICE FOR C
                          04    10  02EA                 BSBB    PROCRE                     ; CREATE THE PROCESS
                50        01    D0  02EC        50$:     MOVL    #1,R0                      ; SET RELASE THE BUFFER FLAG
                                05  02EF                 RSB                                ; RETURN TO AST DISPATCHER
                                    02F0
                                    02F0                 .DSABL  LSB
```

## Listing 4-1 Mailbox Ast Code (page 6 of 7)



```
-JOB_CONTROLLER MAIN ROUTINE              3-JUN-1982 22:29:40  VAX-11 Macro V03-00          Page  16
TERMINAL/CARD_READER UNSOLICTED DATA     24-APR-1982 17:02:56  _DBB0:[JOBCTL.SRC]JBCMAIN.MAR;1   (7)

                        02F0   685
                        02F0   686 ;
                        02F0   687 ; LOCAL SUBROUTINE TO CREATE A PROCESS FOR UNSOLICITED INPUT
                        02F0   688 ;
                        02F0   689 ; INPUTS:
                        02F0   690 ;
                        02F0   691 ;        R3 = DESCRIPTOR FOR PROCESS NAME
                        02F0   692 ;        R4 = 1 IF INPUT, OUTPUT, AND ERROR ARE SUPPOSED TO BE
                        02F0   693 ;                CONCEALED DEVICE NAMES LIKE "__TTA0:"
                        02F0   694 ;           = 0 IF NOT USING CONCEALED DEVICE NAMES.
                        02F0   695 ;
                        02F0   696
   50    85    3C       02F0   697 PROCRE: MOVZWL  (R5)+,R0                   ; PICK UP BINARY UNIT NUMBER
   52    85    9A       02F3   698        MOVZBL  (R5)+,R2                   ; GET LENGTH OF CONTROLLER NAME
   52    55    C0       02F6   699        ADDL    R5,R2                      ; ADDRESS OF END OF CONTROLLER NAME
      *0004*   30       02F9   700        BSBW    JBC$BIN2ASC                ; CONVERT UNIT TO ASCII
   82    3A    90       02FC   701        MOVB    #^A/:/,(R2)+               ; TERMINATE DEVICE NAME WITH COLON
 75   5F 8F    90       02FF   702        MOVB    #^A/_/,-(R5)               ; PUT UNDERSCORE AS FIRST CHAR IN NAME
         65    9F       0303   703        PUSHAB  (R5)                       ; PUSH ADR OF "_" NAME
7E   52  55    C3       0305   704        SUBL3   R5,R2,-(SP)                ; LENGTH OF "_" NAME
 75   5F 8F    90       0309   705        MOVB    #^A/_/,-(R5)               ; PUT SECOND "_" ON FRONT
         65    9F       030D   706        PUSHAB  (R5)                       ; PUSH ADR OF "__" NAME
7E   52  55    C3       030F   707        SUBL3   R5,R2,-(SP)                ; LENGTH OF "__" NAME
         54    DD       0313   708        PUSHL   R4                         ; 1 IF USING CONCEALED DEVICE FOR INPUT
                        0315   709                                           ; OUTPUT AND ERRROR, 0 IF NOT
                        0315   710 ;
                        0315   711 ; 0(SP)  = CONCEALED DEVICE FLAG
                        0315   712 ; 4(SP)  = DESCRIPTOR FOR "__" NAME, CONCEALED DEVICE NAME
                        0315   713 ; 12(SP) = DESCRIPTOR FOR "_" NAME, PROCESS NAME
                        0315   714 ;
 55   0C AE    DE       0315   715        MOVAL   12(SP),R5                  ; ADDRESS OF PROCESS NAME DESCRIPTOR
    54   55    D0       0319   716        MOVL    R5,R4                      ; ASSUME NOT USING CONCEALED DEVICE NAME
   04 6E       E9       031C   717        BLBC    (SP),10$                   ; BRANCH IF NOT USING CONCEALED DEV NAMES
 54   04 AE    DE       031F   718        MOVAL   4(SP),R4                   ; GET CONCEALED DEVICE NAME DESCRIPTOR
                        0323   719 ;
                        0323   720 ; 0(SP) IS AN UNITITIALIZED SCRATCH LONG WORD
                        0323   721 ;
   50    6E    DE       0323   722 10$:   MOVAL   (SP),R0                    ; GET THE ADDRESS OF SCRATCH CELL
51   00000000'9F  9A    0326   723        MOVZBL  @#SYS$GB_DEFPRI,R1         ; SET SYSTEM DEFINED PRIORITY
                        032D   724        $CREPRC_S       -                  ; CREATE A PROCESS
                        032D   725                INPUT=  (R4),-             ; DEIVICE INPUT IS TERMINAL
                        032D   726                OUTPUT= (R4),-             ; LIKEWISE FOR THE DEVICE OUTPUT
                        032D   727                ERROR=  (R4),-             ; AND ALSO DEVICE ERROR
                        032D   728                BASPRI= R1,-               ; INITIAL PRIORITY OF JOB
                        032D   729                PIDADR= (R0),-             ; PLACE TO STORE PROCESS ID
                        032D   730                PRCNAM= (R5),-             ; ESTABLISH PROCESS NAME SAME AS DEVICE
                        032D   731                IMAGE=  (R3),-             ; IMAGE TO RUN
                        032D   732                PRVADR= W^JBC$Q_PRIVMASK,-; PROCESS DEFAULT PRIVILEGE
                        032D   733                UIC=#<1@16+4>              ; UIC IS [1,4]
      5D 50    E8       0353   734        BLBS    R0,60$                     ; BR IF ALL IS OK
         55    D4       0356   735        CLRL    R5                         ; ASSUME DUPLICATE PROCESS NAME
0094 8F  50    B1       0358   736        CMPW    R0,#SS$_DUPLNAM            ; IS ERROR DUPLICATE NAME
         C4    13       035D   737        BEQL    10$                        ; BR IF YES, TRY TO CREATE WITH NO NAME
                        035F   738        SHOW_ERROR      CREATE_PROC        ; REPORT THE ERROR
   52    6E    9E       0361   739        MOVAB   (SP),R2                    ; GET ADDRESS OF SCRATCH WORD
                        0364   740        $ASSIGN_S (R4),(R2)                ; ASSIGN A CHANNEL TO THE DEVICE
      32 50    E9       0371   741        BLBC    R0,60$                     ; BR IF THIS FAILED
```

## Listing 4-1 Mailbox Ast Code (page 7 of 7)

```
JBCMAIN                          -JOB_CONTROLLER MAIN ROUTINE          3-JUN-1982 22:29:40  VAX-11 Macro V03-00
V03-001                          TERMINAL/CARD_READER UNSOLICTED DATA  24-APR-1982 17:02:56  _DBB0:[JOBCTL.SRC]JBCMAI

                                  0374              $QIO_S  CHAN=(R2),-                : WRITE MESSAGE TO TERMINAL
                                  0374                      FUNC=#IO$_WRITEVBLK,-      : WRITE THE DATA
                                  0374                      ASTADR=B^TTMSGAST,-        : AST WHEN WRITE IS DONE
                                  0374                      ASTPRM=(R2),-              : CHANNEL NUMBER IS AST PARAMET
                                  0374                      P1=B^100$,-                : ADDRESS OF MESSAGE STRING
                                  0374                      P2=S^#<110$-100$>,-        : LENGTH OF MESSAGE
                                  0374                      P4=#32                     : NORMAL CARRAIGE CONTROL
                    0A 50    E8   0397              BLBS    R0,50$                     : BR IF QIO WAS ACCEPTED
                                  039A              $DASSGN_S (R2)                     : ELSE DEASSIGN TO RE-ENABLE UN
                       50    D4   03A4     50$:     CLRL    R0                         : RO=LBC=PROCESS CREATE ERROR
                       02    BA   03A6     60$:     POPR    #^M<R1>                    : CLEAR SCRATCH WORD OR GET PRO
                 5E    10    C0   03A8              ADDL    #16,SP                     : CLEAN OFF TWO STRING DESCRIPT
                             05   03AB              RSB                                :
                                  03AC
45 52 43 52 50 2D 57 2D 43 42 4A 25   03AC     100$:    .ASCII  \%JBC-W-PRCREAT, process create error\
20 73 73 65 63 6F 72 70 20 2C 54 41   03B8
72 6F 72 72 65 20 65 74 61 65 72 63   03C4
                                  03D0     110$:
                                  03D0
                                  03D0     TTMSGAST:
                         0000     03D0              .WORD   0                          : AST ENTRY POINT
                                  03D2              $DASSGN_S 4(AP)                    : DEASSIGN THE CHANNEL
                             04   03DD              RET                                :
                                  03DE
                                  03DE     ;
                                  03DE     ; LOCAL ROUTINE TO HANDLE HOURLY AST PROCESSING
                                  03DE     ;
                                  03DE     SYM$HOURLY:
            000004E3'EF      D4   03DE              CLRL    JOB_L_POSSIBLE             : CLEAR COUNT OF POSSIBLE PROC
        000004D4'EF   02     9A   03E4              MOVZBL  #2,JOB_L_PCBINDEX          : SET INDEX IN PCB VECTOR
            000004E6'EF      D0   03EB     10$:     MOVL    JOB_L_SAVEMODE,-           : SET ACCESS MODE
            000004E0'EF           03F1                      JOB_L_MODE                 : FOR AST DELIVERY
                                  03F6              $CMKRNL_S       -                  : CALL KERNEL ROUT. TO CHECK A
                                  03F6                      ROUTIN=CHK_HRDAY_BITS,-    : ADDRESS OF ROUTINE
                                  03F6                      ARGLST=JOB_L_COUNT         : ADDRESS OF ARGUEMENT LIST
                    06 50    E9   0409              BLBC    R0,60$                     : IF LBC THEN CONTINUE
            000004E3'EF      D6   040C              INCL    JOB_L_POSSIBLE             : INCREMENT COUNT
            00000000'GF      F3   0412     50$:     AOBLEQ  G^SCH$GL_MAXPIX,-
         CD 000004D4'EF           0419                      JOB_L_PCBINDEX,10$         : LOOP THRU ALL
            000004E3'EF      D5   041E              TSTL    JOB_L_POSSIBLE             : FIND ANY TO BE DELETED?
                     10      13   0424              BEQL    100$                       : IF EQL THEN NO
            000004E6'EF      D7   0426              DECL    JOB_L_SAVEMODE             : DECREMENT AST ACCESS MODE
                     15      19   042C              BLSS    100$                       : IF LSS THEN DONE THEM ALL
                                  042E
                                  042E              $SETIMR_S       -                  : SET TIMER FOR CHECK IN A FEW
                                  042E                      ASTADR=SYM$CHKLOGINS,-     : AST ADDRESS
                                  042E                      DAYTIM=JBC$Q_NXTMINS,-     : N MINUTES LATER
                                  042E                      REQIDT=#JBC_K_MINUTES      : INDICATE NOT AN HOURLY AST
                             05   0443     100$:    RSB                                : RETURN
                                  0444
                                  0444
```

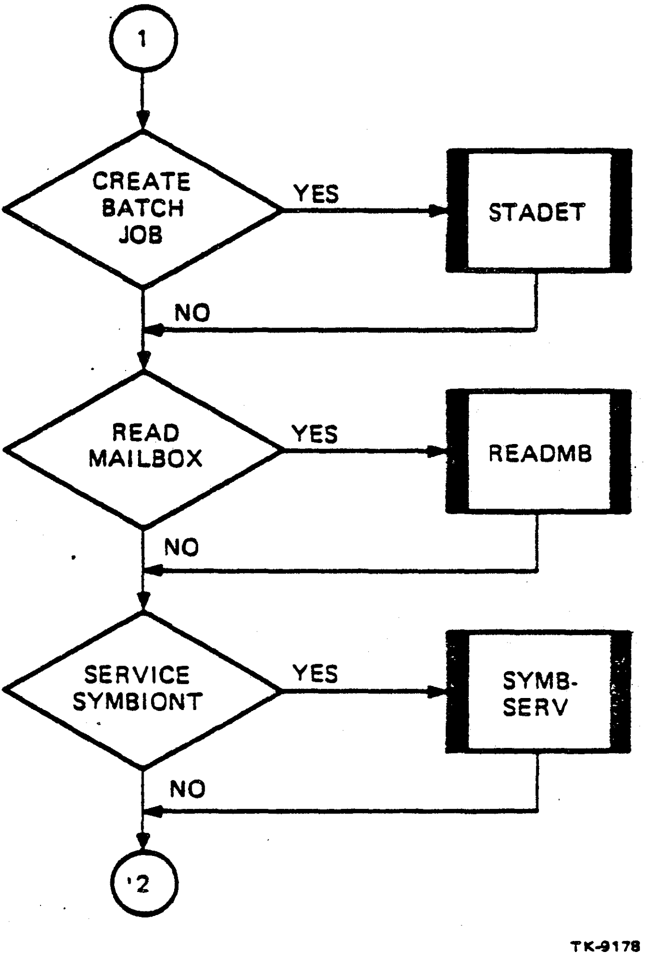


Figure 4-4 Job Controller Main Loop (page 1 of 3)

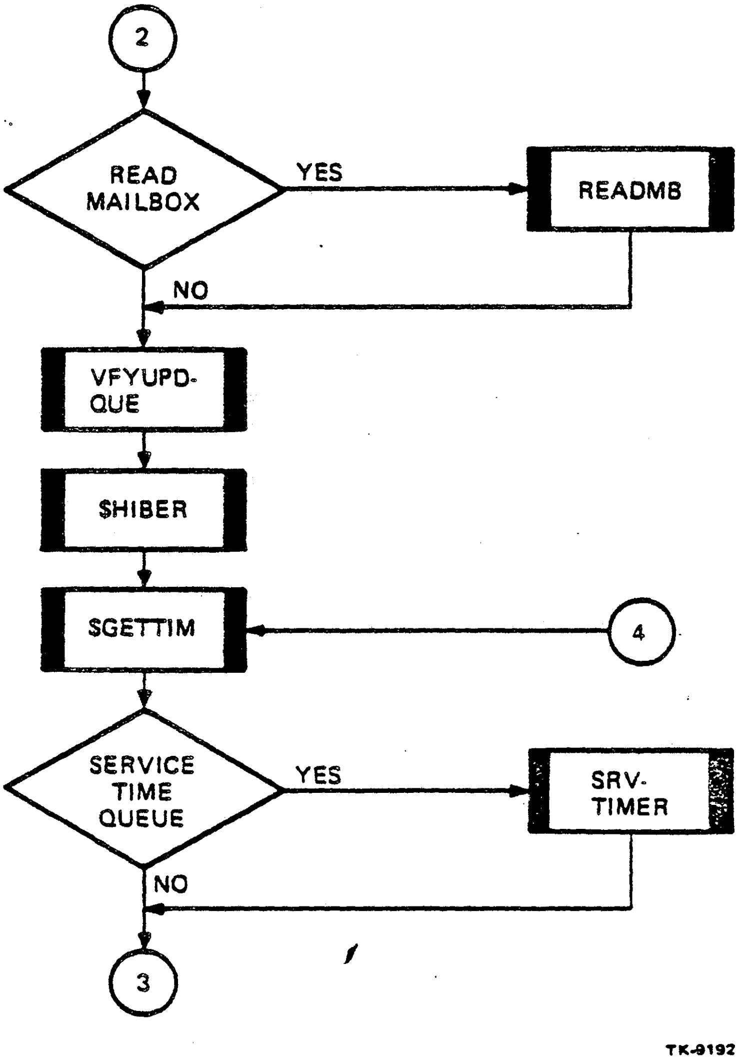


Figure 4-5 Job Controller Main Loop (page 2 of 3)

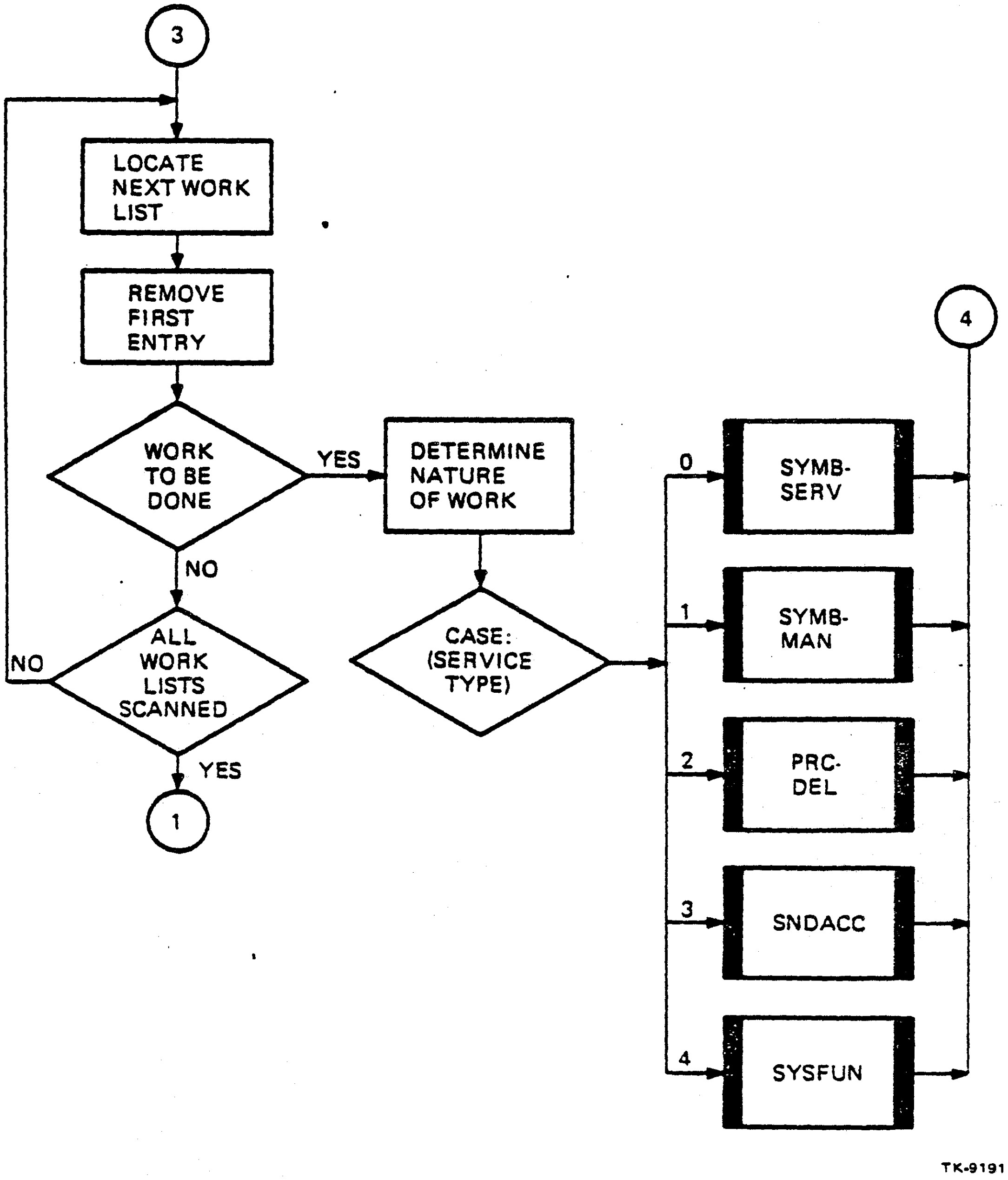
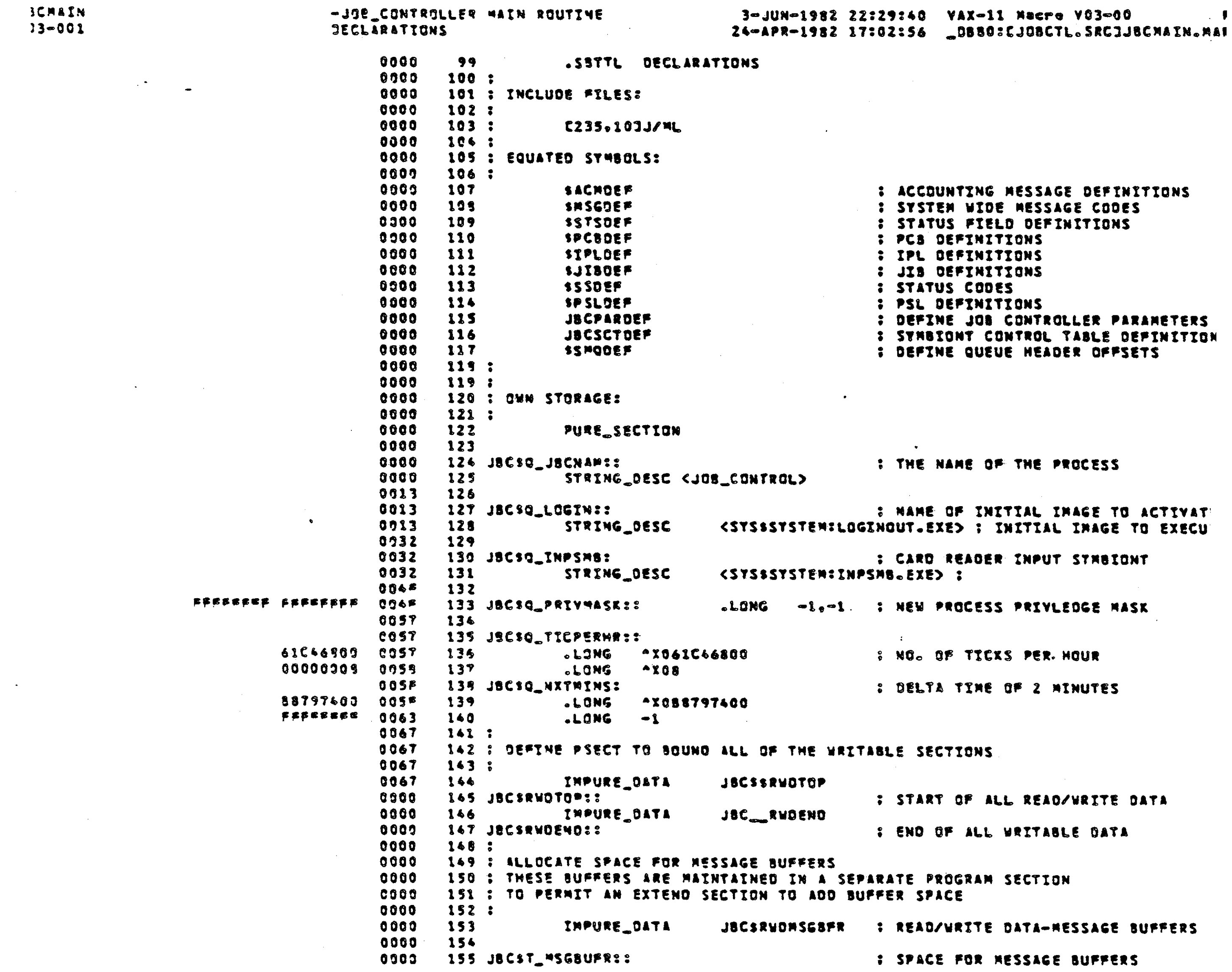


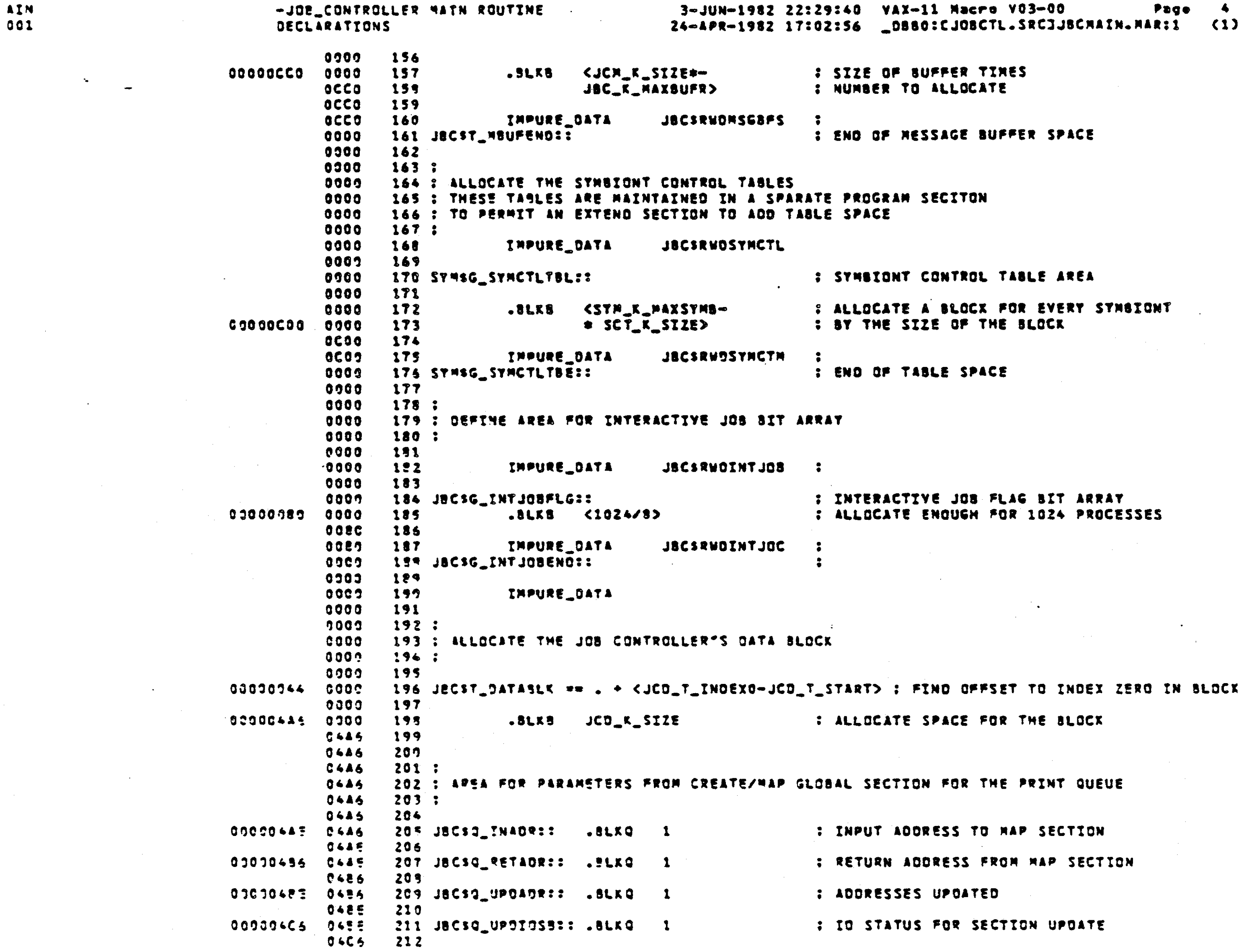
Figure 4-6 Job Controller Main Loop (page 3 of 3)

## Listing 4-2 Main Loop Code (page 1 of 8)



```
BCMAIN                       -JOB_CONTROLLER MAIN ROUTINE          3-JUN-1982 22:29:40  VAX-11 Macro V03-00
03-001                       DECLARATIONS                         24-APR-1982 17:02:56  _DBB0:[JOBCTL.SRC]JBCMAIN.MAI

                          0000         .SBTTL  DECLARATIONS
                          0000     ;
                          0000     ; INCLUDE FILES:
                          0000     ;
                          0000     ;       [235,10]J/ML
                          0000     ;
                          0000     ; EQUATED SYMBOLS:
                          0000     ;
                          0000          $ACMDEF                       ; ACCOUNTING MESSAGE DEFINITIONS
                          0000          $MSGDEF                       ; SYSTEM WIDE MESSAGE CODES
                          0000          $STSDEF                       ; STATUS FIELD DEFINITIONS
                          0000          $PCBDEF                       ; PCB DEFINITIONS
                          0000          $IPLDEF                       ; IPL DEFINITIONS
                          0000          $JIBDEF                       ; JIB DEFINITIONS
                          0000          $SSDEF                        ; STATUS CODES
                          0000          $PSLDEF                       ; PSL DEFINITIONS
                          0000          JBCPARDEF                     ; DEFINE JOB CONTROLLER PARAMETERS
                          0000          JBCSCTDEF                     ; SYMBIONT CONTROL TABLE DEFINITION
                          0000          $SMQDEF                       ; DEFINE QUEUE HEADER OFFSETS
                          0000     ;
                          0000     ;
                          0000     ; OWN STORAGE:
                          0000     ;
                          0000          PURE_SECTION
                          0000
                          0000     JBC$Q_JBCNAM::                     ; THE NAME OF THE PROCESS
                          0000          STRING_DESC <JOB_CONTROL>
                          0013
                          0013     JBC$Q_LOGIN::                      ; NAME OF INITIAL IMAGE TO ACTIVAT
                          0013          STRING_DESC     <SYS$SYSTEM:LOGINOUT.EXE> ; INITIAL IMAGE TO EXECU
                          0032
                          0032     JBC$Q_INPSMB:                      ; CARD READER INPUT SYMBIONT
                          0032          STRING_DESC     <SYS$SYSTEM:INPSMB.EXE> ;
                          004F
       FFFFFFFF FFFFFFFF  004F     JBC$Q_PRIVMASK::        .LONG   -1,-1   ; NEW PROCESS PRIVLEDGE MASK
                          0057
                          0057     JBC$Q_TICPERHR::
                61C46800  0057          .LONG   ^X061C46800           ; NO. OF TICKS PER HOUR
                00000008  005B          .LONG   ^X08
                          005F     JBC$Q_NXTMINS:                     ; DELTA TIME OF 2 MINUTES
                88797400  005F          .LONG   ^X088797400
                FFFFFFFF  0063          .LONG   -1
                          0067     ;
                          0067     ; DEFINE PSECT TO BOUND ALL OF THE WRITABLE SECTIONS
                          0067     ;
                          0067          IMPURE_DATA     JBC$$RWDTOP
                          0000     JBC$RWDTOP::                       ; START OF ALL READ/WRITE DATA
                          0000          IMPURE_DATA     JBC__RWDEND
                          0000     JBC$RWDEND::                       ; END OF ALL WRITABLE DATA
                          0000     ;
                          0000     ; ALLOCATE SPACE FOR MESSAGE BUFFERS
                          0000     ; THESE BUFFERS ARE MAINTAINED IN A SEPARATE PROGRAM SECTION
                          0000     ; TO PERMIT AN EXTEND SECTION TO ADD BUFFER SPACE
                          0000     ;
                          0000          IMPURE_DATA     JBC$RWDMSGBFR   ; READ/WRITE DATA-MESSAGE BUFFERS
                          0000
                          0000     JBC$T_MSGBUFR::                    ; SPACE FOR MESSAGE BUFFERS
```

Listing 4-2 Main Loop Code (page 2 of 8)

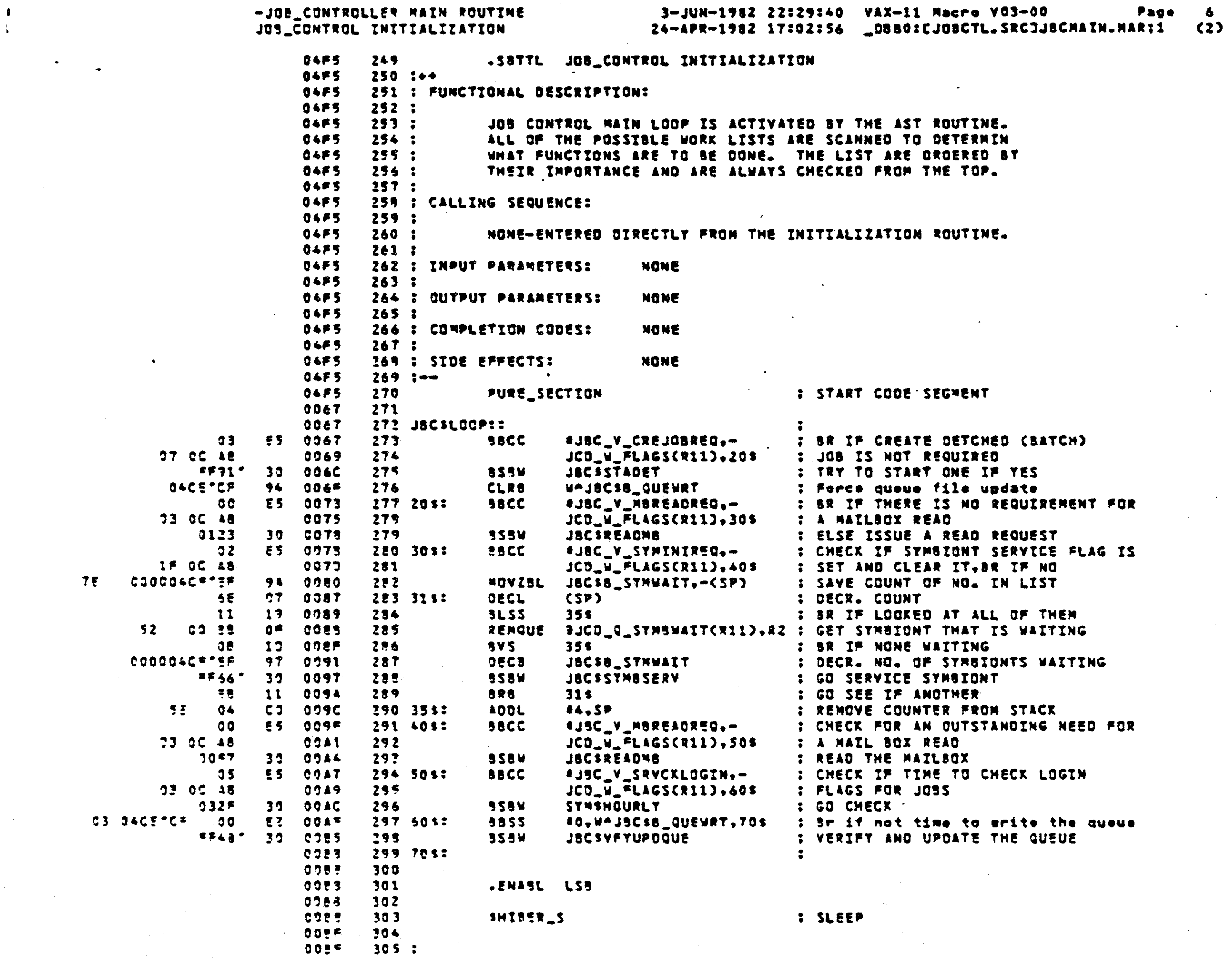


```
AIN           -JOB_CONTROLLER MAIN ROUTINE             3-JUN-1982 22:29:40  VAX-11 Macro V03-00          Page   4
001           DECLARATIONS                            24-APR-1982 17:02:56  _DB80:[JOBCTL.SRC]JBCMAIN.MAR;1   (1)

          0000
00000CC0  0000          .BLKB   <JCM_K_SIZE*-           ; SIZE OF BUFFER TIMES
          0CC0                  JBC_K_MAXBUFR>          ; NUMBER TO ALLOCATE
          0CC0
          0CC0          IMPURE_DATA     JBC$RWOMSGBFS   ;
          0000  JBC$T_MBUFEND::                         ; END OF MESSAGE BUFFER SPACE
          0000
          0000  ;
          0000  ; ALLOCATE THE SYMBIONT CONTROL TABLES
          0000  ; THESE TABLES ARE MAINTAINED IN A SPARATE PROGRAM SECITON
          0000  ; TO PERMIT AN EXTEND SECTION TO ADD TABLE SPACE
          0000  ;
          0000          IMPURE_DATA     JBC$RWOSYMCTL
          0000
          0000  SYM$G_SYMCTLTBL::                       ; SYMBIONT CONTROL TABLE AREA
          0000
          0000          .BLKB   <STM_K_MAXSYMB-         ; ALLOCATE A BLOCK FOR EVERY SYMBIONT
00000C00  0000                  * SCT_K_SIZE>           ; BY THE SIZE OF THE BLOCK
          0C00
          0C00          IMPURE_DATA     JBC$RWOSYMCTM   ;
          0000  SYM$G_SYMCTLTBE::                       ; END OF TABLE SPACE
          0000
          0000  ;
          0000  ; DEFINE AREA FOR INTERACTIVE JOB BIT ARRAY
          0000  ;
          0000
          0000          IMPURE_DATA     JBC$RWOINTJOB   ;
          0000
          0000  JBC$G_INTJOBFLG::                       ; INTERACTIVE JOB FLAG BIT ARRAY
00000080  0000          .BLKB   <1024/8>                ; ALLOCATE ENOUGH FOR 1024 PROCESSES
          0080
          0080          IMPURE_DATA     JBC$RWOINTJOC   ;
          0000  JBC$G_INTJOBEND::                       ;
          0000
          0000          IMPURE_DATA
          0000
          0000  ;
          0000  ; ALLOCATE THE JOB CONTROLLER'S DATA BLOCK
          0000  ;
          0000
00000344  0000  JBC$T_DATABLK == . + <JCD_T_INDEX0-JCD_T_START> ; FIND OFFSET TO INDEX ZERO IN BLOCK
          0000
000004A6  0000          .BLKB   JCD_K_SIZE              ; ALLOCATE SPACE FOR THE BLOCK
          04A6
          04A6
          04A6  ;
          04A6  ; AREA FOR PARAMETERS FROM CREATE/MAP GLOBAL SECTION FOR THE PRINT QUEUE
          04A6  ;
          04A6
000004AE  04A6  JBC$Q_INADR::   .BLKQ   1               ; INPUT ADDRESS TO MAP SECTION
          04AE
000004B6  04AE  JBC$Q_RETADR::  .BLKQ   1               ; RETURN ADDRESS FROM MAP SECTION
          04B6
000004BE  04B6  JBC$Q_UPDADR::  .BLKQ   1               ; ADDRESSES UPDATED
          04BE
000004C6  04BE  JBC$Q_UPDIOSB:: .BLKQ   1               ; IO STATUS FOR SECTION UPDATE
          04C6
```

Listing 4-2 Main Loop Code (page 3 of 8)

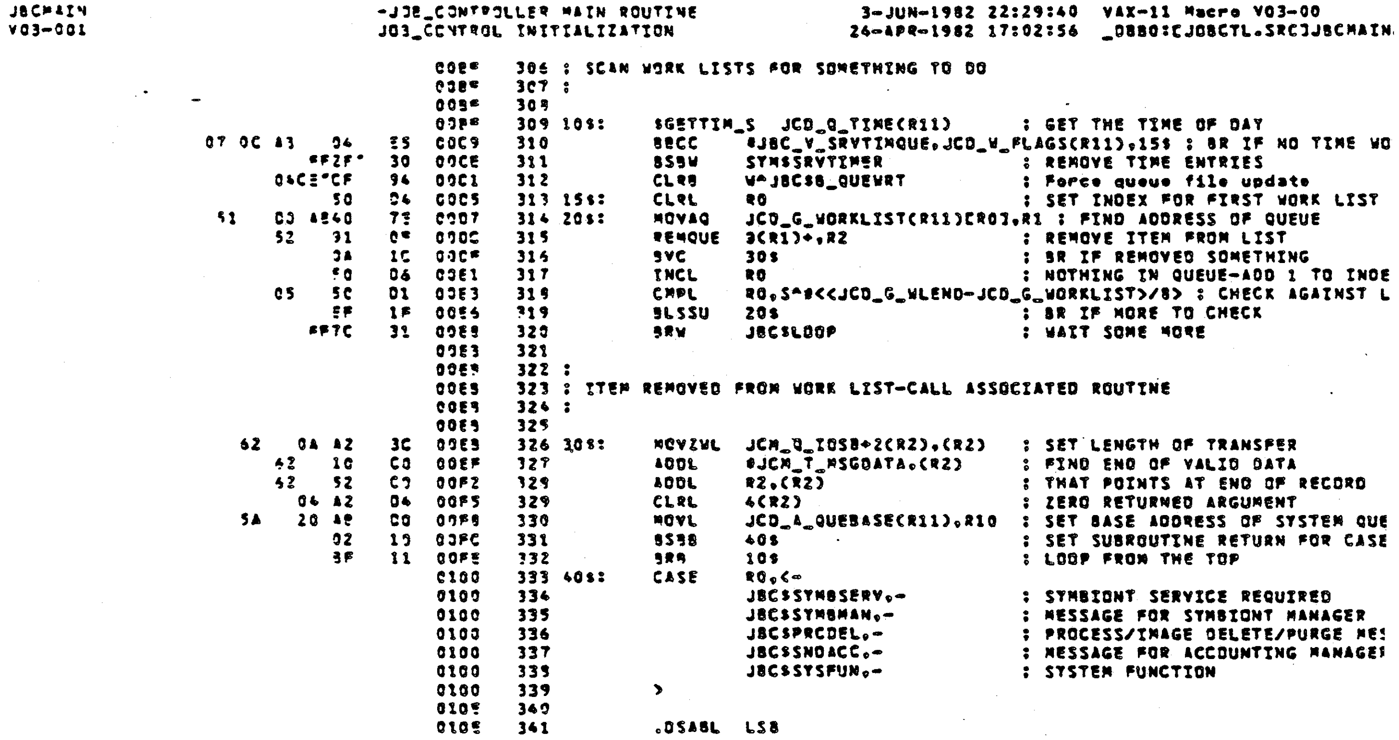
```
JBCMAIN                   -JOB_CONTROLLER MAIN ROUTINE          3-JUN-1982 22:29:40  VAX-11 Macro V03-00
V03-001                   DECLARATIONS                          24-APR-1982 17:02:56  _DBB0:[JOBCTL.SRC]JBCMAIN.MA

000004C8  04C0   213 JBC$Q_NXTAST::  .BLKQ   1                  ; TIME FOR NEXT HOURLY AST
          04C8   214
      00  04C8   215 JBC$B_QUEWRT::  .BYTE   0                  ; FLAG FOR QUEUE UPDATING
      00  04C9   216 JBC$B_SYMWAIT:: .BYTE   0                  ; COUNT OF NO. SYMBIONTS IN WAITLI
          04D0   217
          04D0   218 ;*****
          04D0   219 ;       FOLLOWING LOCATIONS ASSUMED TO BE AJACENT
          04D0   220
          04D0   221 JOB_L_COUNT:
00000004  04D0   222                 .LONG   4                  ; ARGUMENT COUNT
          04D4   223 JOB_L_PCBINDEX:
00000000  04D4   224                 .LONG   0                  ; INDEX INTO PCB VECTOR
          04D8   225 JOB_L_HOUR:
00000000  04D8   226                 .LONG   0                  ; HOUR OF DAY
          04DC   227 JOB_L_DAY:
00000000  04DC   228                 .LONG   0                  ; DAY OF WEEK
          04E0   229 JOB_L_MODE:
00000000  04E0   230                 .LONG   0                  ; ACCESS MODE FOR AST
          04E4   231
          04E4   232 ;*****
          04E4   233
          04E4   234 JOB_L_SAVEMODE:
00000000  04E4   235                 .LONG   0                  ; SAVE ACCESS MODE FOR LOOP
          04E8   236
          04E8   237 JOB_L_POSSIBLE:
00000000  04E8   238                 .LONG   0
          04EC   239 ;
          04EC   240 ; PROTOTYPE NAME FOR CONSTRUCTION OF MAILBOX RESPONCE DEVICE NAMES
          04EC   241 ;
          04EC   242
          04EC   243 JBC$T_MBXNAME::
      00  04EC   244         .BYTE   0                          ; COUNT GOES HERE
42 4D 5F  04ED   245         .ASCII  \_MB\                      ; NAME WITH "_"TO PREVENT SUBSTIT
          04F0   246 JBC$T_MBXUNIT::                            ; THE UNIT NUMBER GOES HERE
000004F5  04F0   247         .BLKB   5                          ; BIGGEST UNIT IS 65K
```

Listing 4-2 Main Loop Code (page 4 of 8)



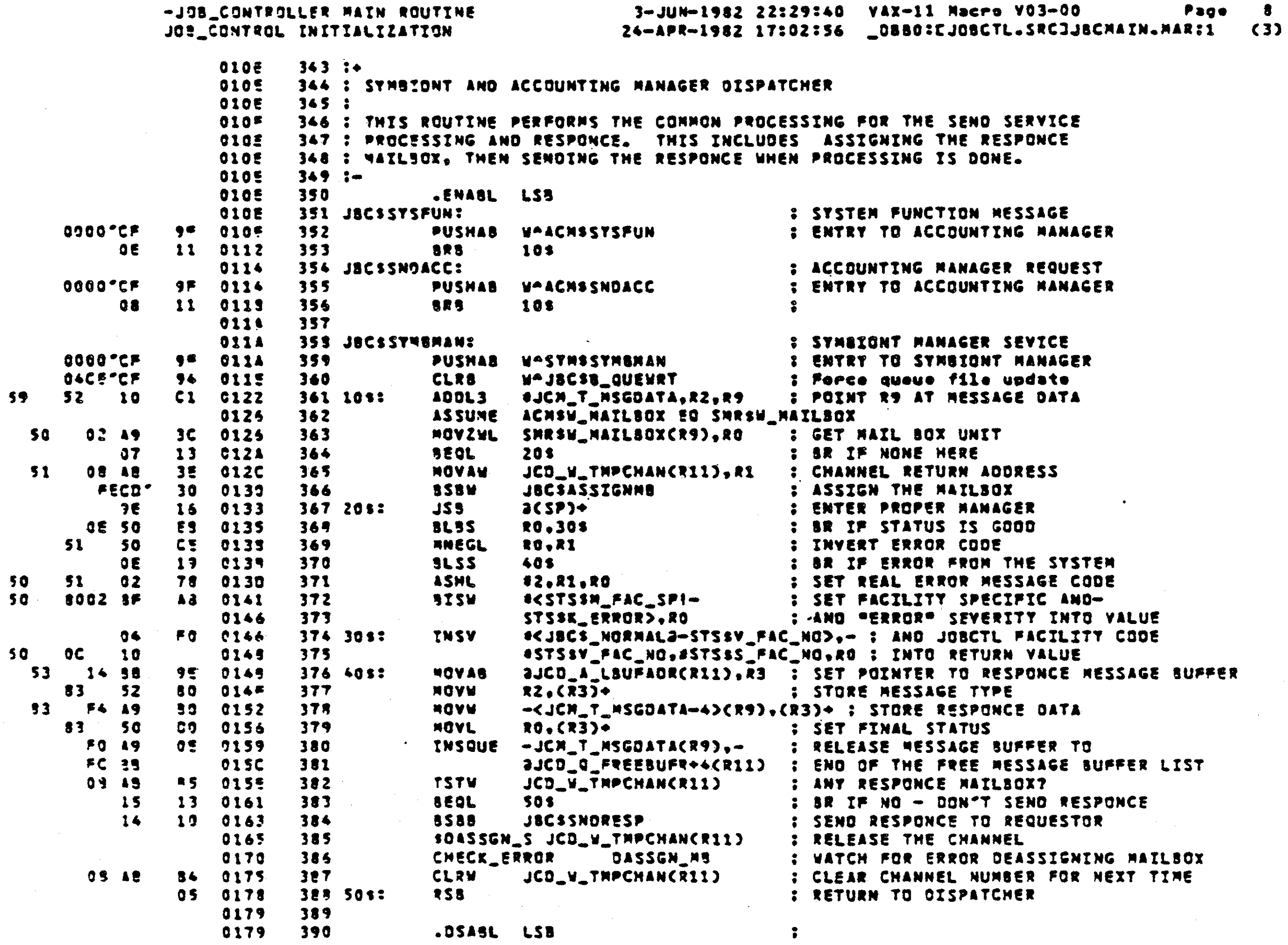
```
                      -JOB_CONTROLLER MAIN ROUTINE          3-JUN-1982 22:29:40  VAX-11 Macro V03-00         Page   6
                      JOB_CONTROL INITIALIZATION           24-APR-1982 17:02:56  _DBB0:[JOBCTL.SRC]JBCMAIN.MAR;1   (2)

                              04F5          .SBTTL  JOB_CONTROL INITIALIZATION
                              04F5      ;++
                              04F5      ; FUNCTIONAL DESCRIPTION:
                              04F5      ;
                              04F5      ;       JOB CONTROL MAIN LOOP IS ACTIVATED BY THE AST ROUTINE.
                              04F5      ;       ALL OF THE POSSIBLE WORK LISTS ARE SCANNED TO DETERMIN
                              04F5      ;       WHAT FUNCTIONS ARE TO BE DONE.  THE LIST ARE ORDERED BY
                              04F5      ;       THEIR IMPORTANCE AND ARE ALWAYS CHECKED FROM THE TOP.
                              04F5      ;
                              04F5      ; CALLING SEQUENCE:
                              04F5      ;
                              04F5      ;       NONE-ENTERED DIRECTLY FROM THE INITIALIZATION ROUTINE.
                              04F5      ;
                              04F5      ; INPUT PARAMETERS:     NONE
                              04F5      ;
                              04F5      ; OUTPUT PARAMETERS:    NONE
                              04F5      ;
                              04F5      ; COMPLETION CODES:     NONE
                              04F5      ;
                              04F5      ; SIDE EFFECTS:         NONE
                              04F5      ;--
                              04F5              PURE_SECTION                    ; START CODE SEGMENT
                              0067
                              0067      JBC$LOOP::                              ;
              03      E5      0067              BBCC    #JBC_V_CREJOBREQ,-      ; BR IF CREATE DETCHED (BATCH)
           07 0C A8           0069                      JCD_W_FLAGS(R11),20$    ; JOB IS NOT REQUIRED
            FF31'     30      006C              BSBW    JBC$STADET              ; TRY TO START ONE IF YES
          04CE'CF     94      006F              CLRB    W^JBC$B_QUEWRT          ; Force queue file update
              00      E5      0073      20$:    BBCC    #JBC_V_MBREADREQ,-      ; BR IF THERE IS NO REQUIREMENT FOR
           03 0C A8           0075                      JCD_W_FLAGS(R11),30$    ; A MAILBOX READ
            0123      30      0078              BSBW    JBC$READMB              ; ELSE ISSUE A READ REQUEST
              02      E5      007B      30$:    BBCC    #JBC_V_SYMINIREQ,-      ; CHECK IF SYMBIONT SERVICE FLAG IS
           1F 0C A8           007D                      JCD_W_FLAGS(R11),40$    ; SET AND CLEAR IT,BR IF NO
 7E   000004CE'EF     9A      0080              MOVZBL  JBC$B_SYMWAIT,-(SP)     ; SAVE COUNT OF NO. IN LIST
              6E      D7      0087      31$:    DECL    (SP)                    ; DECR. COUNT
              11      19      0089              BLSS    35$                     ; BR IF LOOKED AT ALL OF THEM
    52     C0 38      0F      008B              REMQUE  @JCD_Q_SYMBWAIT(R11),R2 ; GET SYMBIONT THAT IS WAITING
              08      1D      008F              BVS     35$                     ; BR IF NONE WAITING
     000004CE'EF      97      0091              DECB    JBC$B_SYMWAIT           ; DECR. NO. OF SYMBIONTS WAITING
            FF56'     30      0097              BSBW    JBC$SYMBSERV            ; GO SERVICE SYMBIONT
              E8      11      009A              BRB     31$                     ; GO SEE IF ANOTHER
         5E   04      C0      009C      35$:    ADDL    #4,SP                   ; REMOVE COUNTER FROM STACK
              00      E5      009F      40$:    BBCC    #JBC_V_MBREADREQ,-      ; CHECK FOR AN OUTSTANDING NEED FOR
           03 0C A8           00A1                      JCD_W_FLAGS(R11),50$    ; A MAIL BOX READ
            00F7      30      00A4              BSBW    JBC$READMB              ; READ THE MAILBOX
              05      E5      00A7      50$:    BBCC    #JBC_V_SRVCKLOGIN,-     ; CHECK IF TIME TO CHECK LOGIN
           03 0C A8           00A9                      JCD_W_FLAGS(R11),60$    ; FLAGS FOR JOBS
            032F      30      00AC              BSBW    SYM$HOURLY              ; GO CHECK
 03 04CE'CF   00      E2      00AF      60$:    BBSS    #0,W^JBC$B_QUEWRT,70$   ; Br if not time to write the queue
            FF48'     30      00B5              BSBW    JBC$VFYUPDQUE           ; VERIFY AND UPDATE THE QUEUE
                              00B8      70$:                                    ;
                              00B8
                              00B8              .ENABL  LSB
                              00B8
                              00B8              $HIBER_S                        ; SLEEP
                              00BF
                              00BF      ;
```

Listing 4-2 Main Loop Code (page 5 of 8)



```
JBCMAIN                                -JOB_CONTROLLER MAIN ROUTINE           3-JUN-1982 22:29:40  VAX-11 Macro V03-00
V03-001                                JOB_CONTROL INITIALIZATION            26-APR-1982 17:02:56  _DBB0:[JOBCTL.SRC]JBCMAIN.

                                  00BE    306 ; SCAN WORK LISTS FOR SOMETHING TO DO
                                  00BE    307 ;
                                  00BE    308
                                  00BE    309 10$:    $GETTIM_S  JCD_Q_TIME(R11)       ; GET THE TIME OF DAY
         07 0C A3   04      E5    00C9    310         BBCC     #JBC_V_SRVTIMQUE,JCD_W_FLAGS(R11),15$ ; BR IF NO TIME WO
               FF2F'        30    00CE    311         BSBW     SYN$SRVTIMER             ; REMOVE TIME ENTRIES
            04CE'CF         94    00D1    312         CLRB     W^JBC$B_QUEWRT           ; Force queue file update
                  50        D4    00D5    313 15$:    CLRL     R0                       ; SET INDEX FOR FIRST WORK LIST
      51    03 AB40         7E    00D7    314 20$:    MOVAQ    JCD_G_WORKLIST(R11)[R0],R1 ; FIND ADDRESS OF QUEUE
            52    91        0F    00DC    315         REMQUE   @(R1)+,R2                ; REMOVE ITEM FROM LIST
                  0A        1C    00DF    316         BVC      30$                      ; BR IF REMOVED SOMETHING
                  50        D6    00E1    317         INCL     R0                       ; NOTHING IN QUEUE-ADD 1 TO INDE
            05    50        D1    00E3    318         CMPL     R0,S^#<<JCD_G_WLEND-JCD_G_WORKLIST>/8> ; CHECK AGAINST L
                  EF        1F    00E6    319         BLSSU    20$                      ; BR IF MORE TO CHECK
               FF7C         31    00E8    320         BRW      JBC$LOOP                 ; WAIT SOME MORE
                                  00EB    321
                                  00EB    322 ;
                                  00EB    323 ; ITEM REMOVED FROM WORK LIST-CALL ASSOCIATED ROUTINE
                                  00EB    324 ;
                                  00EB    325
         62    0A A2        3C    00EB    326 30$:    MOVZWL   JCM_Q_IOSB+2(R2),(R2)    ; SET LENGTH OF TRANSFER
            62    10        C0    00EF    327         ADDL     #JCM_T_MSGDATA,(R2)      ; FIND END OF VALID DATA
            62    52        C0    00F2    328         ADDL     R2,(R2)                  ; THAT POINTS AT END OF RECORD
               04 A2        D4    00F5    329         CLRL     4(R2)                    ; ZERO RETURNED ARGUMENT
         5A    20 AB        D0    00F8    330         MOVL     JCD_A_QUEBASE(R11),R10   ; SET BASE ADDRESS OF SYSTEM QUE
                  02        10    00FC    331         BSBB     40$                      ; SET SUBROUTINE RETURN FOR CASE
                  BE        11    00FE    332         BRB      10$                      ; LOOP FROM THE TOP
                                  0100    333 40$:    CASE     R0,<-
                                  0100    334                  JBC$SYMBSERV,-           ; SYMBIONT SERVICE REQUIRED
                                  0100    335                  JBC$SYMBMAN,-            ; MESSAGE FOR SYMBIONT MANAGER
                                  0100    336                  JBC$PRCDEL,-             ; PROCESS/IMAGE DELETE/PURGE MES
                                  0100    337                  JBC$SNDACC,-             ; MESSAGE FOR ACCOUNTING MANAGE
                                  0100    338                  JBC$SYSFUN,-             ; SYSTEM FUNCTION
                                  0100    339         >
                                  010D    340
                                  010D    341         .DSABL   LSB
```

Listing 4-2 Main Loop Code (page 6 of 8)

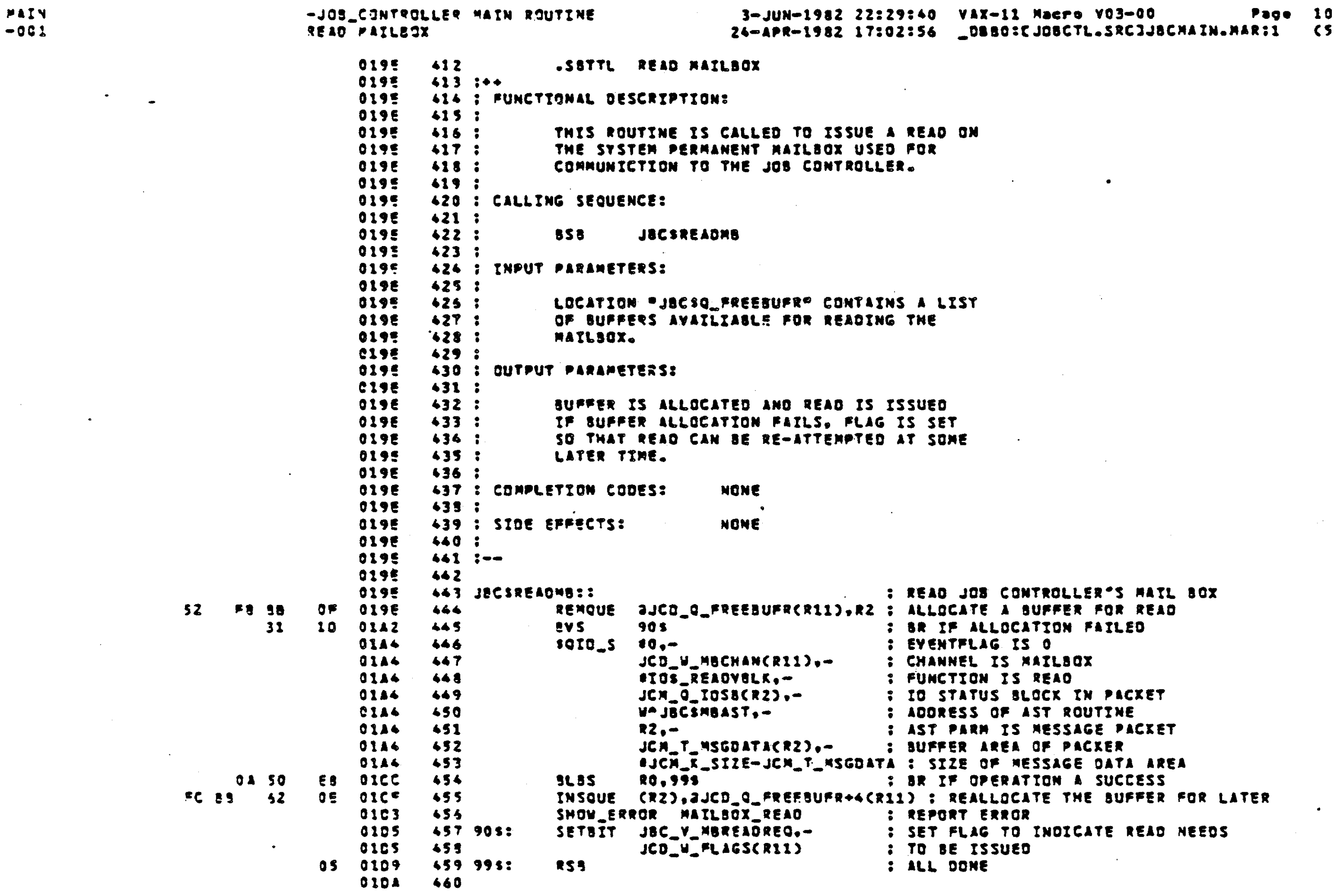


```
            -JOB_CONTROLLER MAIN ROUTINE             3-JUN-1982 22:29:40  VAX-11 Macro V03-00         Page   8
            JOB_CONTROL INITIALIZATION              24-APR-1982 17:02:56  _DBB0:[JOBCTL.SRC]JBCMAIN.MAR;1   (3)

                         010E    343 ;+
                         010E    344 ; SYMBIONT AND ACCOUNTING MANAGER DISPATCHER
                         010E    345 ;
                         010E    346 ; THIS ROUTINE PERFORMS THE COMMON PROCESSING FOR THE SEND SERVICE
                         010E    347 ; PROCESSING AND RESPONCE.  THIS INCLUDES  ASSIGNING THE RESPONCE
                         010E    348 ; MAILBOX, THEN SENDING THE RESPONCE WHEN PROCESSING IS DONE.
                         010E    349 ;-
                         010E    350            .ENABL  LSB
                         010E    351 JBC$SYSFUN:                              ; SYSTEM FUNCTION MESSAGE
    0000'CF         9F   010E    352            PUSHAB  W^ACM$SYSFUN         ; ENTRY TO ACCOUNTING MANAGER
         0E         11   0112    353            BRB     10$
                         0114    354 JBC$SNDACC:                              ; ACCOUNTING MANAGER REQUEST
    0000'CF         9F   0114    355            PUSHAB  W^ACM$SNDACC         ; ENTRY TO ACCOUNTING MANAGER
         08         11   0118    356            BRB     10$                  ;
                         011A    357
                         011A    358 JBC$SYMBMAN:                             ; SYMBIONT MANAGER SEVICE
    0000'CF         9F   011A    359            PUSHAB  W^SYM$SYMBMAN        ; ENTRY TO SYMBIONT MANAGER
    04CE'CF         94   011E    360            CLRB    W^JBC$B_QUEWRT       ; Force queue file update
59    52    10      C1   0122    361 10$:       ADDL3   #JCM_T_MSGDATA,R2,R9 ; POINT R9 AT MESSAGE DATA
                         0126    362            ASSUME  ACM$W_MAILBOX EQ SMR$W_MAILBOX
  50    02 A9       3C   0126    363            MOVZWL  SMR$W_MAILBOX(R9),R0   ; GET MAIL BOX UNIT
            07      13   012A    364            BEQL    20$                    ; BR IF NONE HERE
  51    08 AB       3E   012C    365            MOVAW   JCD_W_TMPCHAN(R11),R1  ; CHANNEL RETURN ADDRESS
          FECD'     30   0130    366            BSBW    JBC$ASSIGNMB           ; ASSIGN THE MAILBOX
            9E      16   0133    367 20$:       JSB     @(SP)+                 ; ENTER PROPER MANAGER
         0E 50      E9   0135    368            BLBS    R0,30$                 ; BR IF STATUS IS GOOD
       51   50      CE   0138    369            MNEGL   R0,R1                  ; INVERT ERROR CODE
            0E      19   013B    370            BLSS    40$                    ; BR IF ERROR FROM THE SYSTEM
50     51   02      78   013D    371            ASHL    #2,R1,R0               ; SET REAL ERROR MESSAGE CODE
50     8002 8F      A8   0141    372            BISW    #<STS$M_FAC_SP1-       ; SET FACILITY SPECIFIC AND-
                         0146    373                    STS$K_ERROR>,R0        ; AND "ERROR" SEVERITY INTO VALUE
            04      F0   0146    374 30$:       INSV    #<JBC$_NORMAL@-STS$V_FAC_NO>,- ; AND JOBCTL FACILITY CODE
50     0C   10           014B    375                    #STS$V_FAC_NO,#STS$S_FAC_NO,R0 ; INTO RETURN VALUE
  53    14 AB       9E   014B    376 40$:       MOVAB   @JCD_A_LBUFADR(R11),R3 ; SET POINTER TO RESPONCE MESSAGE BUFFER
       83   52      B0   014F    377            MOVW    R2,(R3)+               ; STORE MESSAGE TYPE
  83    F4 A9       B0   0152    378            MOVW    -<JCM_T_MSGDATA-4>(R9),(R3)+ ; STORE RESPONCE DATA
       83   50      D0   0156    379            MOVL    R0,(R3)+               ; SET FINAL STATUS
         F0 A9      0E   0159    380            INSQUE  -JCM_T_MSGDATA(R9),-   ; RELEASE MESSAGE BUFFER TO
         FC BB           015C    381                    @JCD_Q_FREEBUFR+4(R11) ; END OF THE FREE MESSAGE BUFFER LIST
         08 AB      B5   015E    382            TSTW    JCD_W_TMPCHAN(R11)     ; ANY RESPONCE MAILBOX?
            15      13   0161    383            BEQL    50$                    ; BR IF NO - DON'T SEND RESPONCE
            14      10   0163    384            BSBB    JBC$SNDRESP            ; SEND RESPONCE TO REQUESTOR
                         0165    385            $DASSGN_S JCD_W_TMPCHAN(R11)   ; RELEASE THE CHANNEL
                         0170    386            CHECK_ERROR     DASSGN_MB      ; WATCH FOR ERROR DEASSIGNING MAILBOX
         08 AB      B4   0175    387            CLRW    JCD_W_TMPCHAN(R11)     ; CLEAR CHANNEL NUMBER FOR NEXT TIME
                    05   0178    388 50$:       RSB                            ; RETURN TO DISPATCHER
                         0179    389
                         0179    390            .DSABL  LSB                    ;
```

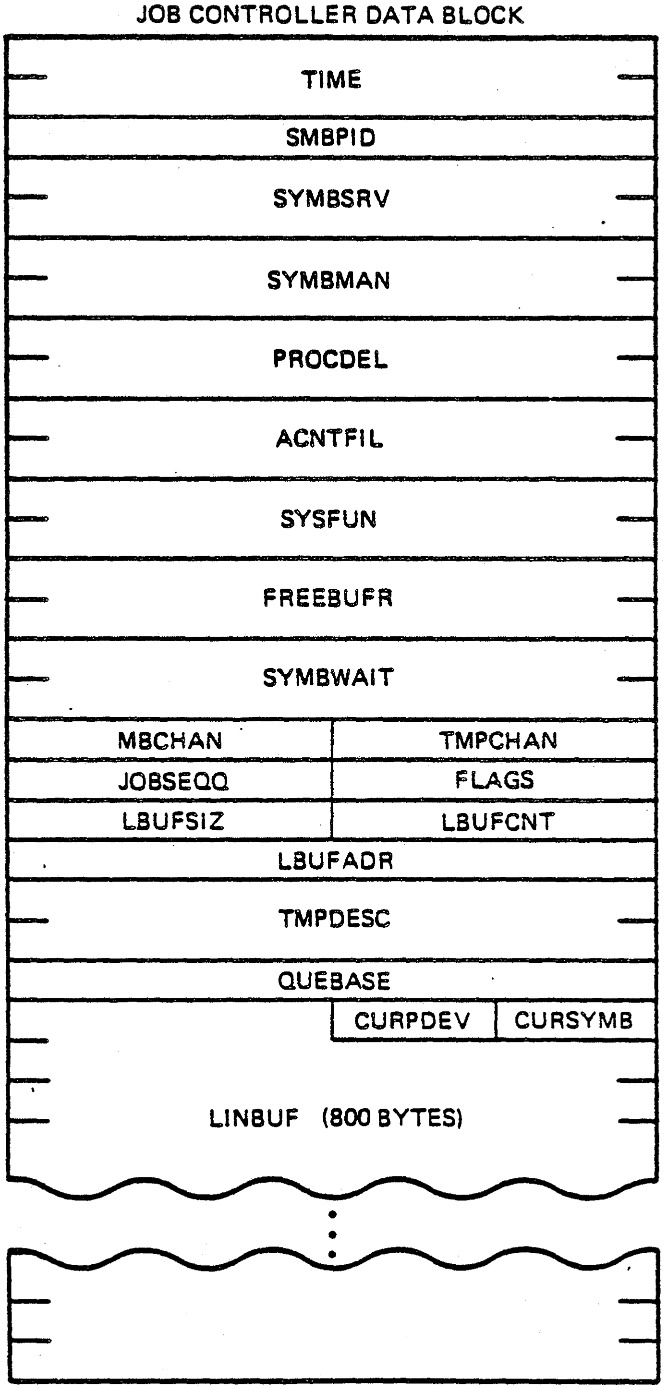
## Listing 4-2 Main Loop Code (page 7 of 8)

```
JBCMAIN                              -JOB_CONTROLLER MAIN ROUTINE         3-JUN-1982 22:29:40  VAX-11 Macro V03-00
V03-001                              JOB_CONTROL INITIALIZATION          24-APR-1982 17:02:56  _DBB0:[JOBCTL.SRC]JBCMAIN.

                                0179
                                0179  ;+
                                0179  ; JBC$SNDRESP - SEND RESPONCE
                                0179  ;
                                0179  ;       THIS SUBROUTINE IS CALLED TO SEND A MESSAGE TO THE REQUESTING
                                0179  ;       PROCESS VIA THE SUPPLIED MAILBOX.
                                0179  ;
                                0179  ; INPUTS:        R3 IS END OF MESSAGE IN JBC LINE BUFFER
                                0179  ;
                                0179  ;-
                                0179
                                0179  JBC$SNDRESP::                                  ;
      51    53    14 AB    C3   0179          SUBL3   JCD_A_LBUFADR(R11),R3,R1 ; FIND LENGTH
                                017E          $QIO_S  EFN = #0,-               ; WRITE MAILBOX, EVENT FLAG IS 0
                                017E                  CHAN = JCD_W_TMPCHAN(R11),-; CHANNEL NUMBER
                                017E                  FUNC = #<IO$_WRITEVBLK!IO$M_NOW>,- ; OPERATION IS WRITE-
                                017E                  P1 = @JCD_A_LBUFADR(R11),- ; BUFFER ADDRESS
                                017E                  P2 = R1                  ; LENGTH
                           05   0190          RSB                              ;
```

Listing 4-2 Main Loop Code (page 8 of 8)



```
MAIN                          -JOB_CONTROLLER MAIN ROUTINE          3-JUN-1982 22:29:40  VAX-11 Macro V03-00          Page  10
-001                          READ MAILBOX                         24-APR-1982 17:02:56  _DBB0:[JOBCTL.SRC]JBCMAIN.MAR;1   (5

                       019E            .SBTTL  READ MAILBOX
                       019E      ;++
                       019E      ; FUNCTIONAL DESCRIPTION:
                       019E      ;
                       019E      ;       THIS ROUTINE IS CALLED TO ISSUE A READ ON
                       019E      ;       THE SYSTEM PERMANENT MAILBOX USED FOR
                       019E      ;       COMMUNICTION TO THE JOB CONTROLLER.
                       019E      ;
                       019E      ; CALLING SEQUENCE:
                       019E      ;
                       019E      ;       BSB     JBC$READMB
                       019E      ;
                       019E      ; INPUT PARAMETERS:
                       019E      ;
                       019E      ;       LOCATION "JBC$Q_FREEBUFR" CONTAINS A LIST
                       019E      ;       OF BUFFERS AVAILIABLE FOR READING THE
                       019E      ;       MAILBOX.
                       019E      ;
                       019E      ; OUTPUT PARAMETERS:
                       019E      ;
                       019E      ;       BUFFER IS ALLOCATED AND READ IS ISSUED
                       019E      ;       IF BUFFER ALLOCATION FAILS, FLAG IS SET
                       019E      ;       SO THAT READ CAN BE RE-ATTEMPTED AT SOME
                       019E      ;       LATER TIME.
                       019E      ;
                       019E      ; COMPLETION CODES:     NONE
                       019E      ;
                       019E      ; SIDE EFFECTS:         NONE
                       019E      ;
                       019E      ;--
                       019E
                       019E      JBC$READMB::                                   ; READ JOB CONTROLLER'S MAIL BOX
52   F8 9B   0F        019E            REMQUE  @JCD_Q_FREEBUFR(R11),R2 ; ALLOCATE A BUFFER FOR READ
        31   1D        01A2            BVS     90$                     ; BR IF ALLOCATION FAILED
                       01A4            $QIO_S  #0,-                    ; EVENTFLAG IS 0
                       01A4                    JCD_W_MBCHAN(R11),-     ; CHANNEL IS MAILBOX
                       01A4                    #IO$_READVBLK,-         ; FUNCTION IS READ
                       01A4                    JCM_Q_IOSB(R2),-        ; IO STATUS BLOCK IN PACKET
                       01A4                    W^JBC$MBAST,-           ; ADDRESS OF AST ROUTINE
                       01A4                    R2,-                    ; AST PARM IS MESSAGE PACKET
                       01A4                    JCM_T_MSGDATA(R2),-     ; BUFFER AREA OF PACKER
                       01A4                    #JCM_K_SIZE-JCM_T_MSGDATA : SIZE OF MESSAGE DATA AREA
     0A 50   E8        01CC            BLBS    R0,99$                  ; BR IF OPERATION A SUCCESS
FC BB   52   0E        01CF            INSQUE  (R2),@JCD_Q_FREEBUFR+4(R11) ; REALLOCATE THE BUFFER FOR LATER
                       01D3            SHOW_ERROR  MAILBOX_READ        ; REPORT ERROR
                       01D5      90$:  SETBIT  JBC_V_MBREADREQ,-       ; SET FLAG TO INDICATE READ NEEDS
                       01D5                    JCD_W_FLAGS(R11)        ; TO BE ISSUED
             05        01D9      99$:  RSB                             ; ALL DONE
                       01DA
```



TK-9181

Figure 4-7 Job Controller Data Block

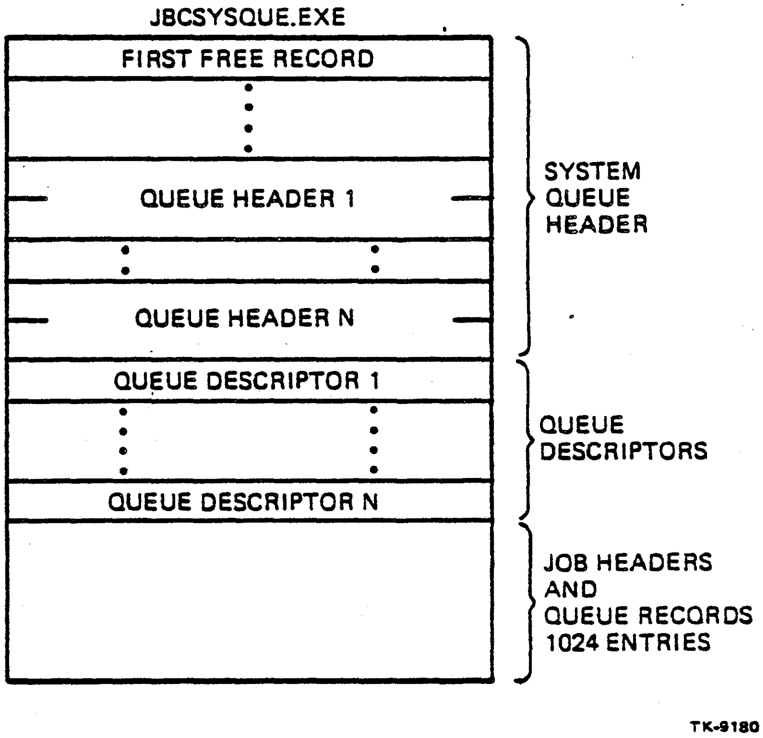


Figure 4-8 File Format for JBCSYSQUE.EXE

- System Queue Header (SQH)
- Symbiont Manager Queue Desctiptors (SMQ)
- Symbiont Job Header (SJH)
- Symbiont Queue Record (SQR)

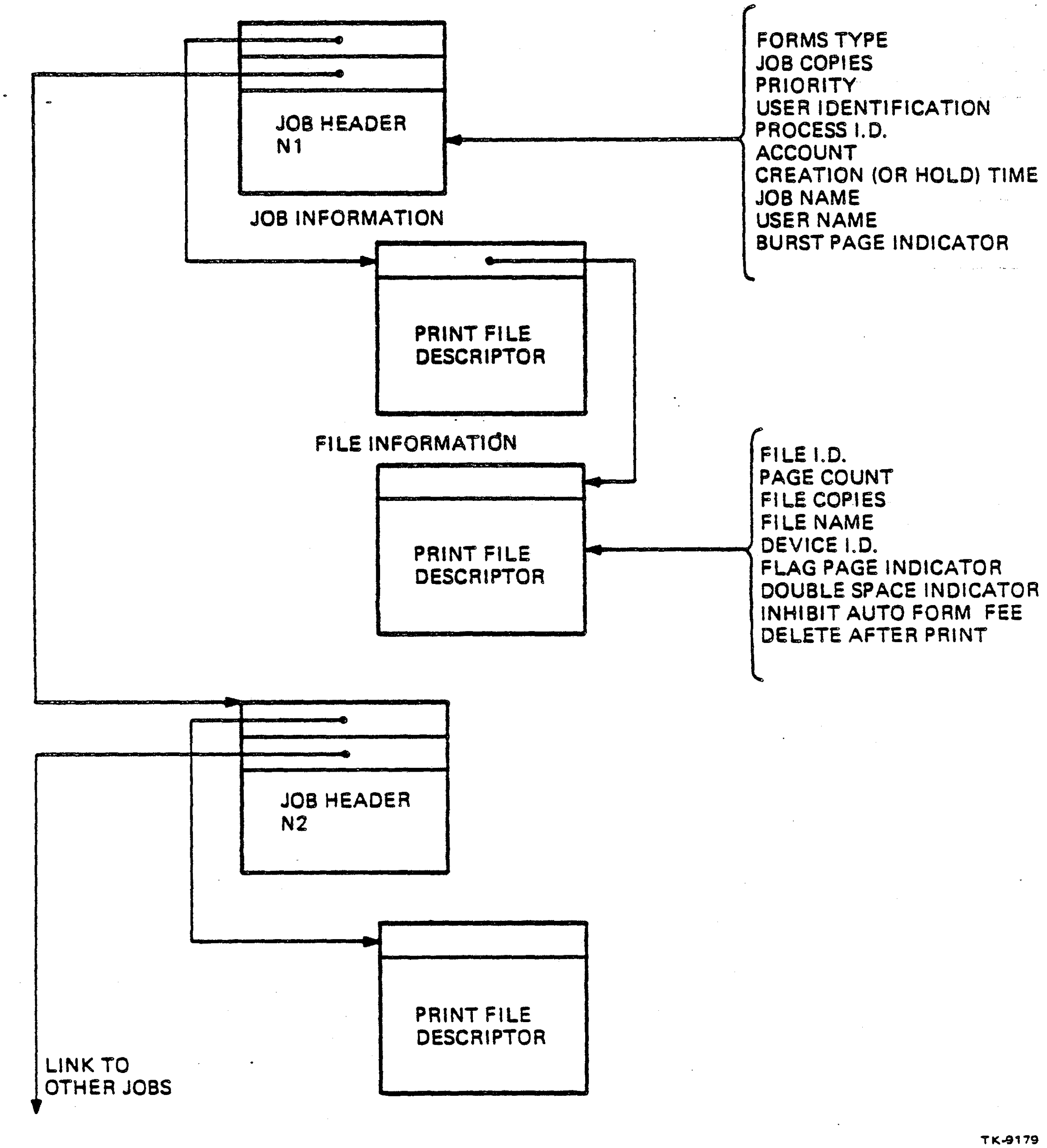


Figure 4-9 Print Job Structure

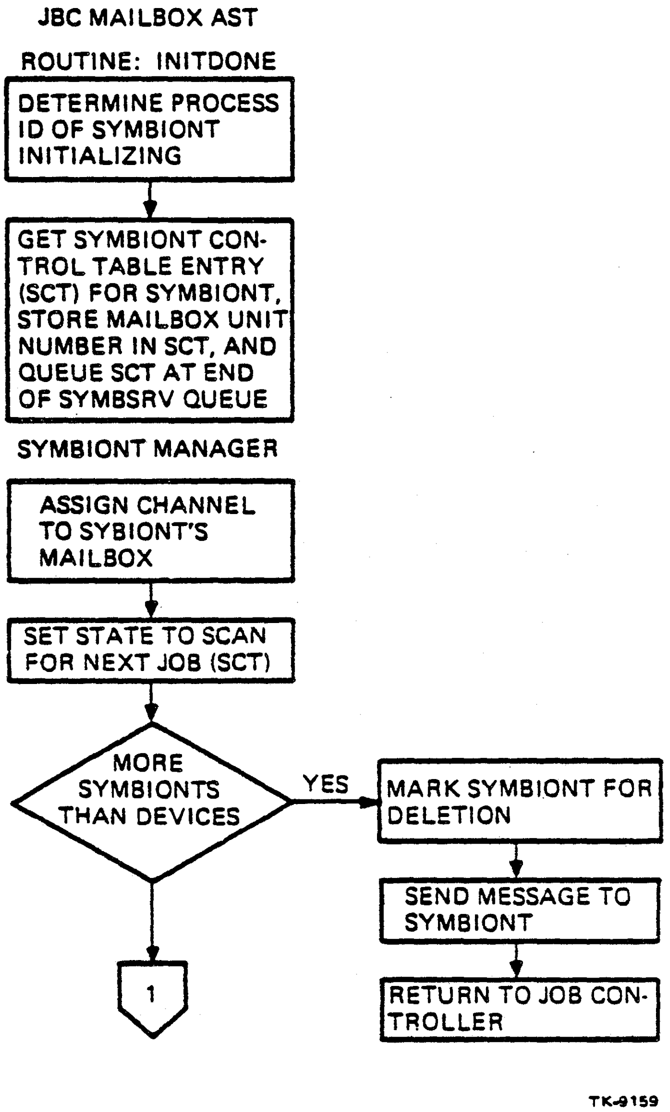

JBC MAILBOX AST
ROUTINE: INITDONE
DETERMINE PROCESS ID OF SYMBIONT INITIALIZING
GET SYMBIONT CONTROL TABLE ENTRY (SCT) FOR SYMBIONT, STORE MAILBOX UNIT NUMBER IN SCT, AND QUEUE SCT AT END OF SYMBSRV QUEUE
SYMBIONT MANAGER
ASSIGN CHANNEL TO SYBIONT'S MAILBOX
SET STATE TO SCAN FOR NEXT JOB (SCT)
MORE SYMBIONTS THAN DEVICES
YES
MARK SYMBIONT FOR DELETION
SEND MESSAGE TO SYMBIONT
RETURN TO JOB CONTROLLER
1
TK-9159

Figure 4-10 Assign Job to Symbiont

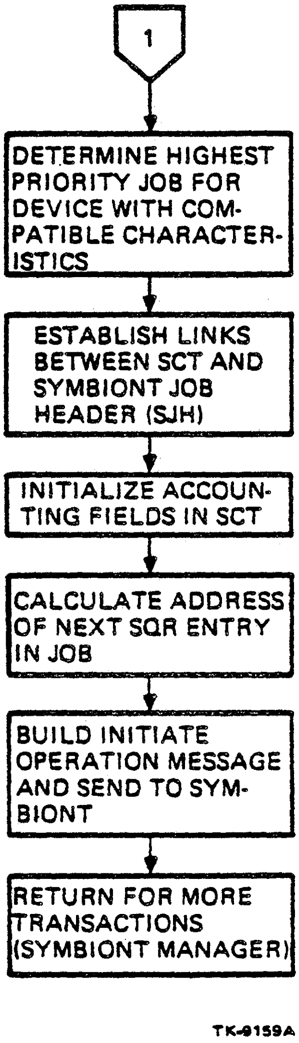

1
DETERMINE HIGHEST PRIORITY JOB FOR DEVICE WITH COMPATIBLE CHARACTERISTICS
ESTABLISH LINKS BETWEEN SCT AND SYMBIONT JOB HEADER (SJH)
INITIALIZE ACCOUNTING FIELDS IN SCT
CALCULATE ADDRESS OF NEXT SQR ENTRY IN JOB
BUILD INITIATE OPERATION MESSAGE AND SEND TO SYMBIONT
RETURN FOR MORE TRANSACTIONS (SYMBIONT MANAGER)
TK-9159A

Figure 4-11 Assign Job to Symbiont

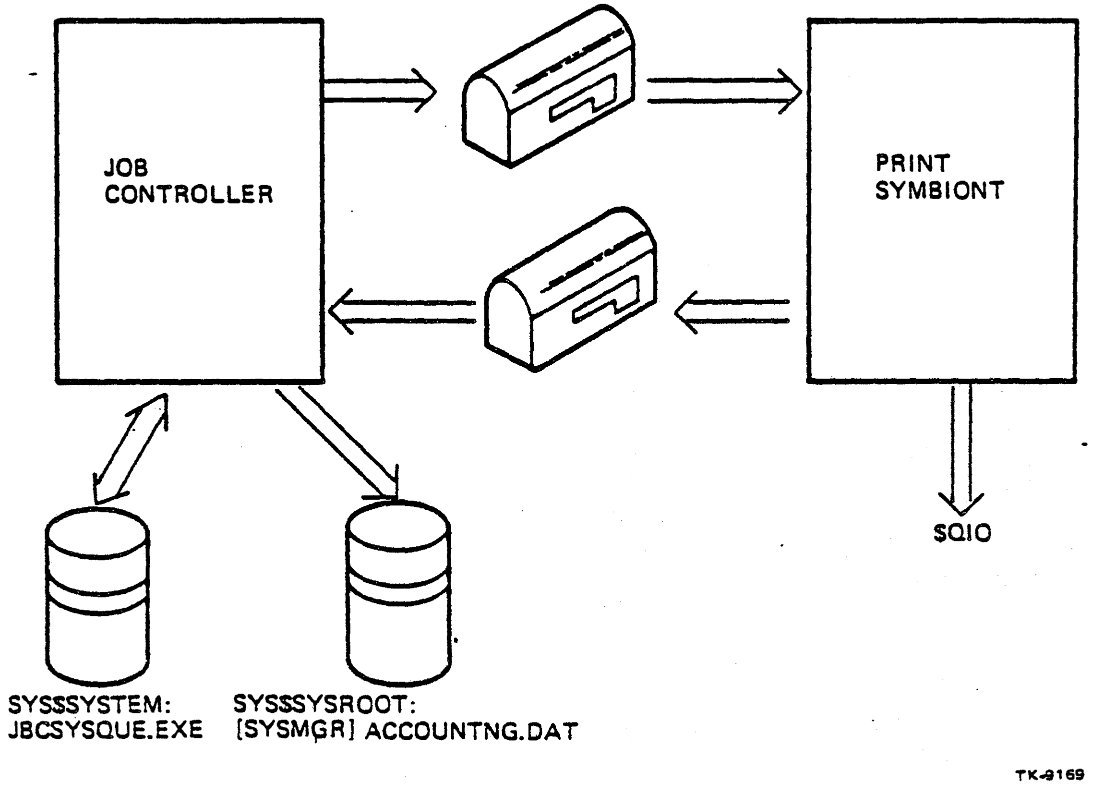


Figure 4-12 Job Controller vs Symbionts

- Job Controller
  - Accounting Manager
  - Queue Manager
  - Symbiont Manager

- Symbionts
  - Subprocess
  - Communication via mailboxes
  - Print symbionts 'formats' print jobs
  - Default print symbiont is SYS$SYSTEM:PRTSMB.EXE

## WRITING A SYMBIONT

- Symbiont Creation
  - INITIALIZE/QUEUE
    Creates the queue descriptor
    Inserts the queue options
  - START/QUEUE
    Increments count of print queues
    Creates a symbiont table entry
    Creates symbiont ($CREPRC)

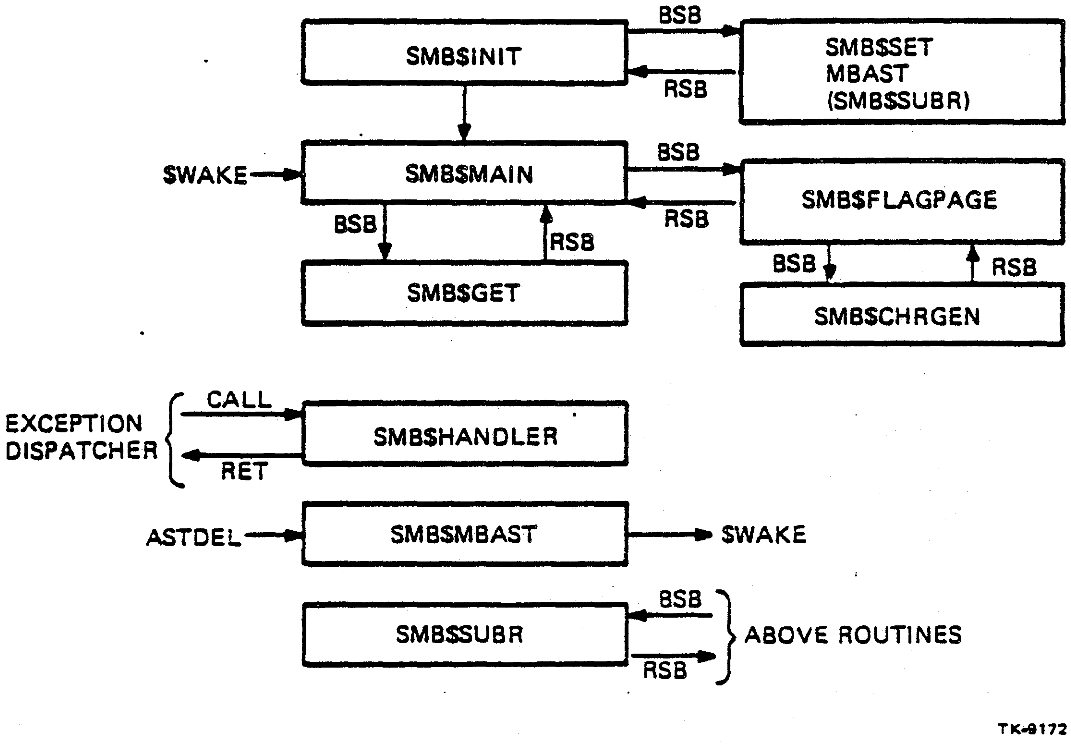


Figure 4-13 Symbiont Code Flow

- Initialization
- Main Routine Loop
- Condition Handler
- Flag page and character generator

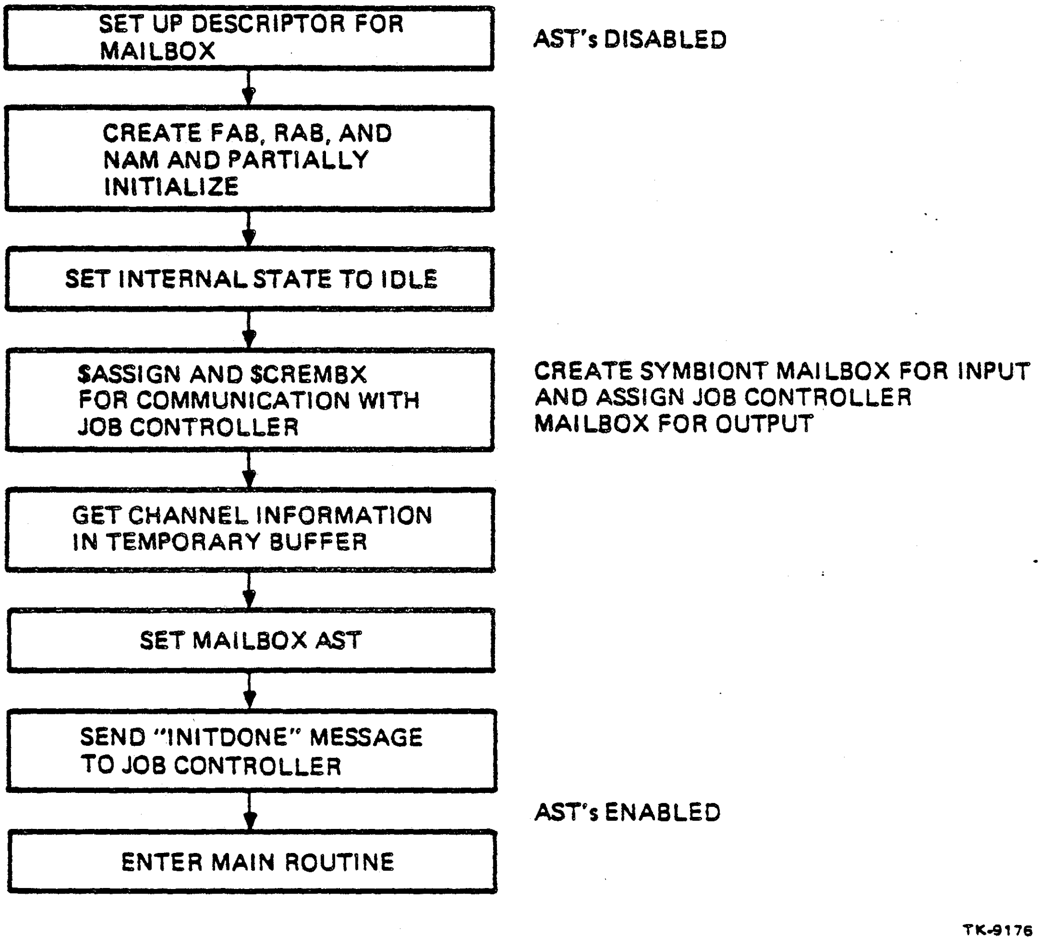

SET UP DESCRIPTOR FOR MAILBOX
AST's DISABLED
CREATE FAB, RAB, AND NAM AND PARTIALLY INITIALIZE
SET INTERNAL STATE TO IDLE
$ASSIGN AND $CREMBX FOR COMMUNICATION WITH JOB CONTROLLER
CREATE SYMBIONT MAILBOX FOR INPUT AND ASSIGN JOB CONTROLLER MAILBOX FOR OUTPUT
GET CHANNEL INFORMATION IN TEMPORARY BUFFER
SET MAILBOX AST
SEND "INITDONE" MESSAGE TO JOB CONTROLLER
AST's ENABLED
ENTER MAIN ROUTINE
TK-9176

Figure 4-14 Initialization Activity

| | |
|---|---|
| MBUNIT | MSGTYPE |

Figure 4-15  INITDONE Message to Job Controller

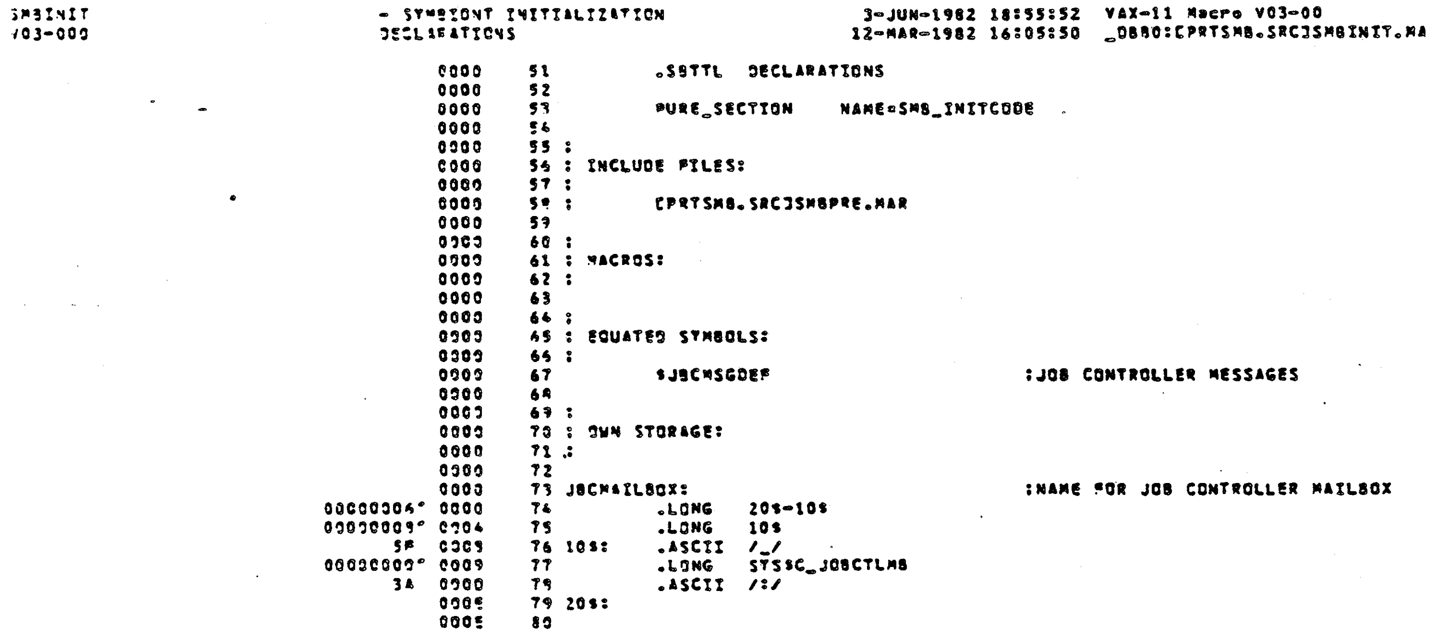
MBUNIT  (Mailbox unit number from $GETCHN service for symbiont mailbox)

MSGTYPE  (=8 = INITDONE) (MSG$_SMBINI)

Note:

Initialization activity (output symbiont)
Executed once during life of symbiont

## Listing 4-3 Initialization Code (page 1 of 4)



```
SMBINIT                          - SYMBIONT INITIALIZATION          3-JUN-1982 18:55:52  VAX-11 Macro V03-00
V03-000                          DECLARATIONS                      12-MAR-1982 16:05:50  _DBB0:[PRTSMB.SRC]SMBINIT.MA

                      0000          .SBTTL  DECLARATIONS
                      0000
                      0000          PURE_SECTION    NAME=SMB_INITCODE
                      0000
                      0000     ;
                      0000     ; INCLUDE FILES:
                      0000     ;
                      0000     ;        [PRTSMB.SRC]SMBPRE.MAR
                      0000
                      0000     ;
                      0000     ; MACROS:
                      0000     ;
                      0000
                      0000     ;
                      0000     ; EQUATED SYMBOLS:
                      0000     ;
                      0000          $JBCMSGDEF                         ;JOB CONTROLLER MESSAGES
                      0000
                      0000     ;
                      0000     ; OWN STORAGE:
                      0000     ;
                      0000
                      0000     JBCMAILBOX:                             ;NAME FOR JOB CONTROLLER MAILBOX
            00000006' 0000          .LONG   20$-10$
            00000008' 0004          .LONG   10$
                  5F  0008     10$: .ASCII  /_/
            00000000' 0009          .LONG   SYS$C_JOBCTLMB
                  3A  000D          .ASCII  /:/
                      000E     20$:
                      000E
```

## Listing 4-3 Initialization Code (page 2 of 4)

```
                  - SYMBIONT INITIALIZATION             3-JUN-1982 18:55:52  VAX-11 Macro V03-00        Page   3
                  SYMBIONT INITIALIZATION ROUTINE      12-MAR-1982 16:05:50  _DBB0:[PRTSMB.SRC]SMBINIT.MAR;1   (1)

                        000E     82          .SBTTL  SYMBIONT INITIALIZATION ROUTINE
                        000E     83 ;++
                        000E     84 ; FUNCTIONAL DESCRIPTION:
                        000E     85 ;
                        000E     86 ;        THIS ROUTINE PERFORMS ALL ONE TIME FUNCTIONS FOR THE
                        000E     87 ;        PRINT SYMBIONT.
                        000E     88 ;
                        000E     89 ; CALLING SEQUENCE:
                        000E     90 ;
                        000E     91 ;        MAIN ENTRY POINT OF SYMBIONT
                        000E     92 ;
                        000E     93 ; INPUT PARAMETERS:
                        000E     94 ;
                        000E     95 ;        NONE
                        000E     96 ;
                        000E     97 ; IMPLICIT INPUTS:
                        000E     98 ;
                        000E     99 ;        NONE
                        000E    100 ;
                        000E    101 ; OUTPUT PARAMETERS:
                        000E    102 ;
                        000E    103 ;        R11 CONTAINS THE ADDRESS OF THE IMPURE DATA BLOCK
                        000E    104 ;        INIT DONE MESSAGE SENT TO SYMBIONT MANAGER
                        000E    105 ;
                        000E    106 ; IMPLICIT OUTPUTS:
                        000E    107 ;
                        000E    108 ;        CHANNEL ASSIGNED TO SYMBIONT MANAGER MAILBOX
                        000E    109 ;        MAILBOX CREATED FOR RECEIPT OF MANAGER MESSAGES
                        000E    110 ;
                        000E    111 ; COMPLETION CODES:
                        000E    112 ;
                        000E    113 ;        NONE
                        000E    114 ;
                        000E    115 ; SIDE EFFECTS:
                        000E    116 ;
                        000E    117 ;        THIS ROUTINE DISPATCHES DIRECTLY TO THE SYMBIONT IDLE LOOP
                        000E    118 ;
                        000E    119 ;--
                        000E    120
                        000E    121
                        000E    122
                        000E    123 SMB_START:                                 ;SYMBIONT INITIAL ENTRY
                  0000  000E    124          .WORD   0                         ;ENTRY MASK
                        0010    125
   5C     0000'CF   DE  0010    126          MOVAL   W^SMB$HANDLER,(FP)        ;SET CONDITION HANDLER ADDRESS
                        0015    127
                        0015    128 ;
                        0015    129 ; DISABLE ASTS UNTIL MESSAGE IS SENT
                        0015    130 ;
                        0015    131
                        0015    132          $SETAST_S       #0                ;DISABLE ASTS
                        0015    133
   5B     0000'CF   DE  0015    134          MOVAL   W^SMB$G_DATA,R11          ;SET ADDR OF IMPURE DATA BLOCK
                        0023    135
                        0023    136 ;
                        0023    137 ; RUN-TIME INITIALIZATION OF DATA FIELDS
                        0023    138 ;
```

## Listing 4-3 Initialization Code (page 3 of 4)

```
SMBINIT                  - SYMBIONT INITIALIZATION              3-JUN-1982 18:55:52  VAX-11 Macro V03-00      |
V03-000                  SYMBIONT INITIALIZATION ROUTINE       12-MAR-1982 16:05:50  _DBB0:[PRTSMB.SRC]SMBINIT.MA

                                0023
        03 AB     0070 8F   B0  0023            MOVW    #SIM$K_SIZE,SD_W_MBREADLEN(R11) ;SET INITIAL MB READ LENGTH
                                0029
        27 AB     0200 8F   B0  0029            MOVW    #SMB$K_TBUFSIZ,SD_W_TBUFCNT(R11) ;SET LENGTH OF TEMP BUFFER
        29 AB     0200 8F   B0  002F            MOVW    #SMB$K_TBUFSIZ,SD_W_TBUFSIZ(R11) ;SET LENGTH OF TEMP BUFFER
        25 AB     01F2 CB   DE  0035            MOVAL   SD_T_TBUF(R11),SD_A_TBUFADR(R11) ;SET ADDRESS OF TEMP BUFFE
                                0039
           56     00F3 CB   DE  0039            MOVAL   SD_G_FAB(R11),R6          ;GET ADDRESS OF FAB
           57     0148 CB   DE  0040            MOVAL   SD_G_RAB(R11),R7          ;GET ADDRESS OF RAB
           58     018C CB   DE  0045            MOVAL   SD_G_NAM(R11),R8          ;GET ADDRESS OF NAM BLK
                                004A
                                004A            ASSUME  FAB$B_BID+1 EQ FAB$B_BLN
           56     5003 8F   B0  004A            MOVW    #FAB$C_BID+<FAB$C_BLN@8>,FAB$B_BID(R6) ;CREATE FAB
                                004F            ASSUME  RAB$B_BID+1 EQ RAB$B_BLN
           57     4401 8F   B0  004F            MOVW    #RAB$C_BID+<RAB$C_BLN@8>,RAB$B_BID(R7) ;CREATE RAB
                                0054            ASSUME  NAM$B_BID+1 EQ NAM$B_BLN
           58     6002 8F   B0  0054            MOVW    #NAM$C_BID+<NAM$C_BLN@8>,NAM$B_BID(R8) ;CREATE NAM BLK
                                0059
             28 A6     58   D0  0059            MOVL    R8,FAB$L_NAM(R6)          ;SET NAME BLOCK ADDRESS IN FAB
             3C A7     56   D0  005D            MOVL    R6,RAB$L_FAB(R7)          ;SET FAB ADDRESS IN RAB
        20 A7     0200 8F   B0  0061            MOVW    #SMB$K_LBUFSIZ,RAB$W_USZ(R7) ;SET RECORD BUFFER SIZE
                                0067            SETBIT  RAB$V_RAH,RAB$L_ROP(R7) ;USE READ-AHEAD
                                006C
           00C4 CB     0C   9A  006C            MOVZBL  #QIO$_NARGS,SD_G_QIOBLK(R11) ;SET QIO BLOCK LENGTH
   0000 CB        0000'8F   3C  0071            MOVZWL  #IO$_WRITELBLK,SD_G_QIOBLK+QIO$_FUNC(R11) ;SET I/O FUNCTIO
                    02 AB   B4  0078            CLRW    SD_B_ERR_FLAGS(R11)       ;CLEAR BOTH SETS OF FLAGS
   0632 CB        01F2 CB   DE  007B            MOVAL   SD_T_LBUF(R11),SD_Q_BUFPNT(R11) ; SET FIRST ADDRESS
   0636 CB        0632 CB   DE  0082            MOVAL   SD_T_LBUF1(R11),SD_Q_BUFPNT+4(R11) ; SET SECOND ADDRESS
                                0089
                                0089            ASSUME  STATE$_IDLE EQ 0
                                0089
                  04 AB     94  0089            CLRB    SD_B_STATE(R11)           ;SET INITIAL STATE TO IDLE
                                008C
                                008C     ;
                                008C     ; ASSIGN THE SYMBIONT MANAGERS MAILBOX
                                008C     ;
                                008C
                                008C            $ASSIGN_S       JBCMAILBOX,-      ;ASSIGN CHANNEL TO THE JOB CONTROL
                                008C                    SD_W_JBCCHAN(R11)         ;MAILBOX-CHANNEL NUMBER STORED HE
                  11 50     E9  009C            BLBS    R0,10$                    ;BR IF NO ERROR
                                009F            SIGNAL  JBC$_MBASGN,#0,R0         ;SIGNAL THE ERROR
                     33     11  00AE            BRB     20$                       ;EXIT
                                00B0
                                00B0     ;
                                00B0     ; CREATE SYMBIONT'S MAILBOX
                                00B0     ;
                                00B0
                                00B0     10$:
                                00B0            $CREMBX_S       -                 ;CREATE A MAIL BOX FOR COMMANDS
                                00B0                    PROMSK = #^X0FF0F,-       ; PROTECTION
                                00B0                    MAXMSG = #SIM$K_SIZE,-    ;MAXIMUM MESSAGE SIZE
                                00B0                    BUFQUO = #2*SIM$K_SIZE,-; 2 MESSAGES MAX
                                00B0                    CHAN   = SD_W_MBCHAN(R11) ; CHANNEL OF CREATED MAILBOX GO
                  19 50     E9  00D1            BLBS    R0,30$                    ;BR IF NO ERROR
                                00D4            SIGNAL  JBC$_SYMBCRE,#0,R0        ;SIGNAL THE ERROR
                                00E3     20$:   $EXIT_S                           ;FORCE IMAGE EXIT
                                00EC
```

Listing 4-3 Initialization Code (page 4 of 4)

```
IT                    - SYMBIONT INITIALIZATION            3-JUN-1982 18:55:52  VAX-11 Macro V03-00          Page   5
00                    SYMBIONT INITIALIZATION ROUTINE     12-MAR-1982 16:05:50  _DBB0:[PRTSMB.SRC]SMBINIT.MAR;1   (1)

                           00EC    196 ;
                           00EC    197 ; SET MAILBOX CHANNEL INFO
                           00EC    198 ;
                           00EC    199
                           00EC    200 30$:
           50    65    DE  00EC    201          MOVAL   SD_W_MBCHAN(R11),R0      ;SET ADDR OF CHANNEL
              FF0E'    30  00EF    202          BSBW    SMB$GETCHAN              ;GET CHANNEL INFO
                           00F2    203
                           00F2    204 ;
                           00F2    205 ; SET UNSOLICITED AST FOR MY MAILBOX
                           00F2    206 ;
                           00F2    207
              FFC8'    30  00F2    208          BSBW    SMB$SETMBAST             ;SET THE MAILBOX AST
      50   C1FE CB     B0  00F5    209          MOVW    SD_T_TBUF+12(R11),R0     ;SET MAILBOX UNIT NUMBER FOR INIT MSG
        39 AB    50    B0  00FA    210          MOVW    R0,SD_W_MBUNIT(R11)      ;SAVE UNIT FOR $DELMBX
              FEFF'    30  00FE    211          BSBW    SMB$INIT_DONE            ;SEND MGR THE INIT DONE MESSAGE
                           0101    212
                           0101    213 ;
                           0101    214 ; ENABLE ASTS NOW
                           0101    215 ;
                           0101    216
                           0101    217          $SETAST_S       #1               ;ENABLE ASTS
                           010A    218
              FEF3'    31  010A    219          BRW     SMB$MAIN                 ;GOTO MAIN LOOP
                           010D    220
                           010D    221          .END    SMB_START
```

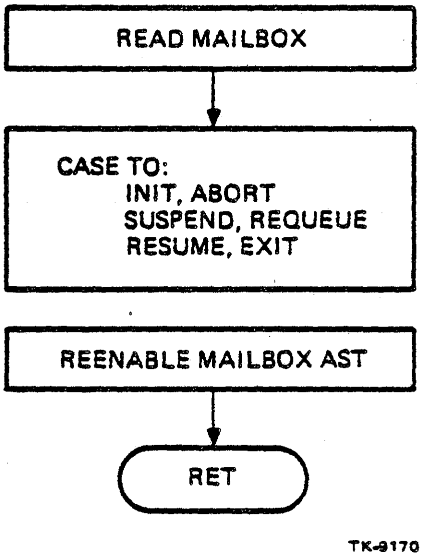
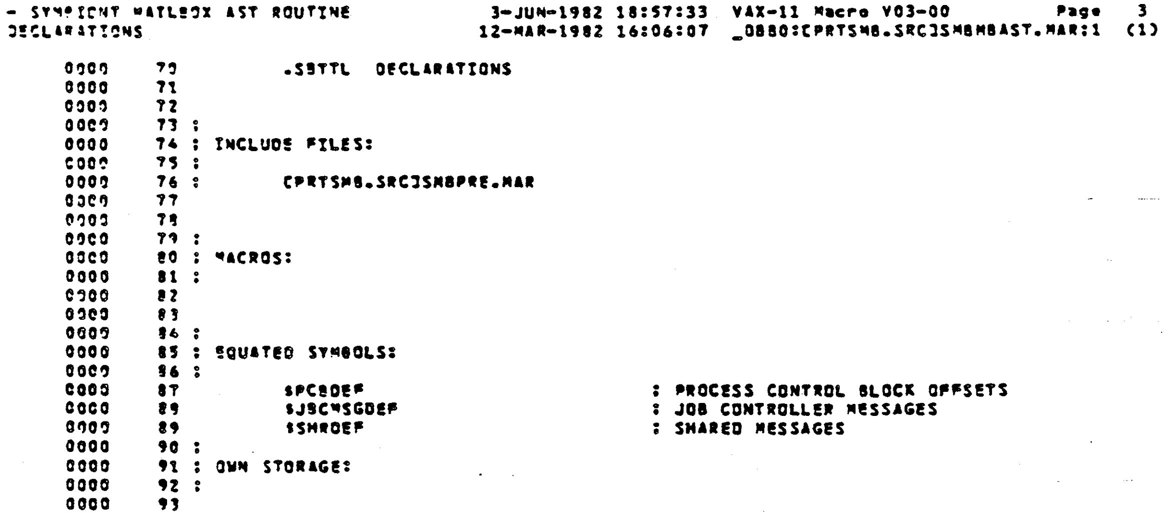


Figure 4-16 Mailbox AST Code Flow

| Init | Abort/Requeue | Suspend | Resume | Exit |
|---|---|---|---|---|
| Assigns Device | $Cancel I/O | Set state suspend | Restore state | $Exit_s |
| Get Channel Info | | Read MB | | |
| $OPEN<br>$CONNECT | | | | |
| Set wake bit<br>$WAKE | | | | |

Listing 4-4 Mailbox Ast (page 1 of 8)

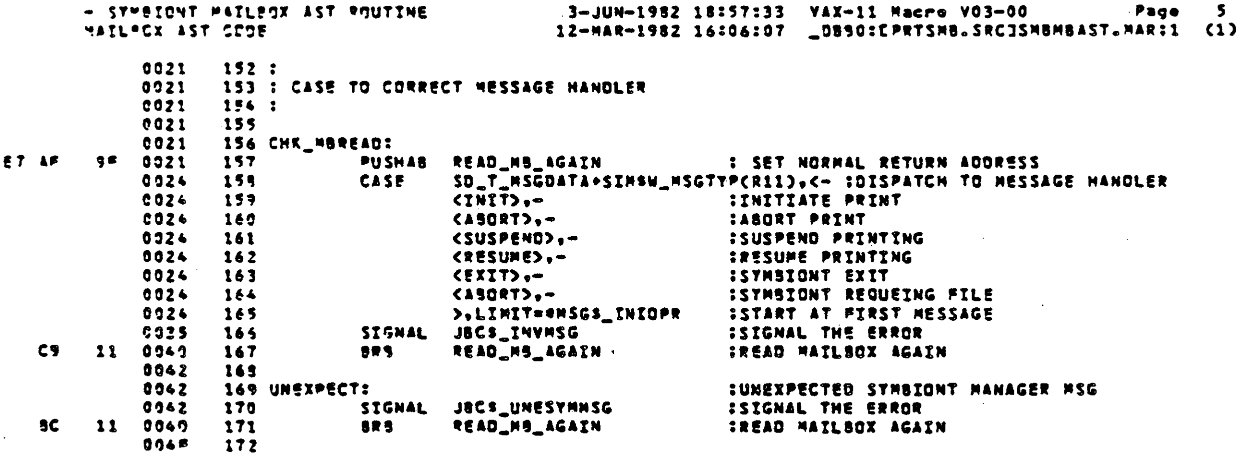


```
- SYMBIONT MAILBOX AST ROUTINE           3-JUN-1982 18:57:33  VAX-11 Macro V03-00           Page   3
DECLARATIONS                            12-MAR-1982 16:06:07  _DBB0:[PRTSMB.SRC]SMBMBAST.MAR;1  (1)

     0000            .SBTTL  DECLARATIONS
     0000
     0000
     0000      ;
     0000      ; INCLUDE FILES:
     0000      ;
     0000      ;       [PRTSMB.SRC]SMBPRE.MAR
     0000
     0000
     0000      ;
     0000      ; MACROS:
     0000      ;
     0000
     0000
     0000      ;
     0000      ; EQUATED SYMBOLS:
     0000      ;
     0000            $PCBDEF                         ; PROCESS CONTROL BLOCK OFFSETS
     0000            $JBCMSGDEF                      ; JOB CONTROLLER MESSAGES
     0000            $SMRDEF                         ; SHARED MESSAGES
     0000      ;
     0000      ; OWN STORAGE:
     0000      ;
     0000
```

## Listing 4-4 Mailbox Ast (page 2 of 8)

```
MBMBAST                        - SYMBIONT MAILBOX AST ROUTINE           3-JUN-1982 18:57:33  VAX-11 Macro V03-00          F
03-000                         MAILBOX AST CODE                        12-MAR-1982 16:06:07  _DBB0:[PRTSMB.SRC]SMBMBAST.MA

                                0000     95          .SBTTL  MAILBOX AST CODE
                                0000     96 ;++
                                0000     97 ; FUNCTIONAL DESCRIPTION:
                                0000     98 ;
                                0000     99 ;
                                0000    100 ; CALLING SEQUENCE:
                                0000    101 ;
                                0000    102 ;        CALLED AT AST LEVEL WHEN SOMETHING IS PUT IN THE MAILBOX
                                0000    103 ;
                                0000    104 ; INPUT PARAMETERS:
                                0000    105 ;
                                0000    106 ;        NONE
                                0000    107 ;
                                0000    108 ; IMPLICIT INPUTS:
                                0000    109 ;
                                0000    110 ;        MESSAGE IN THE MAILBOX
                                0000    111
                                0000    112 ;
                                0000    113 ; OUTPUT PARAMETERS:
                                0000    114 ;
                                0000    115 ;        NONE
                                0000    116 ;
                                0000    117 ; IMPLICIT OUTPUTS:
                                0000    118 ;
                                0000    119 ;        SEE EACH MSG HANDLER
                                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 ;--
                                0000    130
                                0000    131          PURE_SECTION
                                0000    132
                                0000    133
                                0000    134 SMB$MBAST::
                          0E3C  0000    135          .WORD   ^M<R2,R3,R4,R5,R9,R10,R11> ;ENTRY MASK
         5B   0000'CF     DE    0002    136          MOVAL   W^SMB$G_DATA,R11          ;SET IMPURE DATA BLOCK
          5A    04 AB     9A    0007    137          MOVZBL  SD_B_STATE(R11),R10       ;GET CURRENT STATE
                                000B    138 READ_MB_AGAIN:
             5A    01     91    000B    139          CMPB    #STATE$_ASNDEV,R10        ;TRYING TO ASSIGN THE PRINTER
                   03     12    000E    140          BNEQ    10$                       ;BR IF NO
                 0108     30    0010    141          BSBW    ASNDEV                    ;TRY IT AGAIN
                 00CA     30    0013    142 10$:     BSBW    READ_MB_NOW               ;READ THE MAILBOX
                03 50     E8    0016    143          BLBS    R0,CHK_MBREAD             ;BR IF OK
                                0019    144 ;
                                0019    145 ; RE-ENABLE AST
                                0019    146 ;
                                0019    147
                FFE4'     30    0019    148          BSBW    SMB$SETMBAST              ;REENABLE THE AST
      04 AB    5A         90    001C    149          MOVB    R10,SD_B_STATE(R11)       ;SET CURRENT STATE
                          04    0020    150          RET                               ;EXIT THE AST
                                0021    151
```

Listing 4-4 Mailbox Ast (page 3 of 8)



```
- SYMBIONT MAILBOX AST ROUTINE            3-JUN-1982 18:57:33  VAX-11 Macro V03-00          Page   5
MAILBOX AST CODE                         12-MAR-1982 16:06:07  _DBA0:[PRTSMB.SRC]SMBMBAST.MAR;1   (1)

             0021     :
             0021     : CASE TO CORRECT MESSAGE HANDLER
             0021     :
             0021
             0021     CHK_MBREAD:
E7 AF    9F  0021             PUSHAB  READ_MB_AGAIN           : SET NORMAL RETURN ADDRESS
             0024             CASE    SD_T_MSGDATA+SIN$W_MSGTYP(R11),<- ;DISPATCH TO MESSAGE HANDLER
             0024                     <INIT>,-                ;INITIATE PRINT
             0024                     <ABORT>,-               ;ABORT PRINT
             0024                     <SUSPEND>,-             ;SUSPEND PRINTING
             0024                     <RESUME>,-              ;RESUME PRINTING
             0024                     <EXIT>,-                ;SYMBIONT EXIT
             0024                     <ABORT>,-               ;SYMBIONT REQUEING FILE
             0024                     >,LIMIT=#MSG$_INIOPR    ;START AT FIRST MESSAGE
             0035             SIGNAL  JBC$_INVMSG             ;SIGNAL THE ERROR
   C9    11  0040             BRB     READ_MB_AGAIN           ;READ MAILBOX AGAIN
             0042
             0042     UNEXPECT:                               ;UNEXPECTED SYMBIONT MANAGER MSG
             0042             SIGNAL  JBC$_UNESYMMSG          ;SIGNAL THE ERROR
   BC    11  004D             BRB     READ_MB_AGAIN           ;READ MAILBOX AGAIN
             004F
```

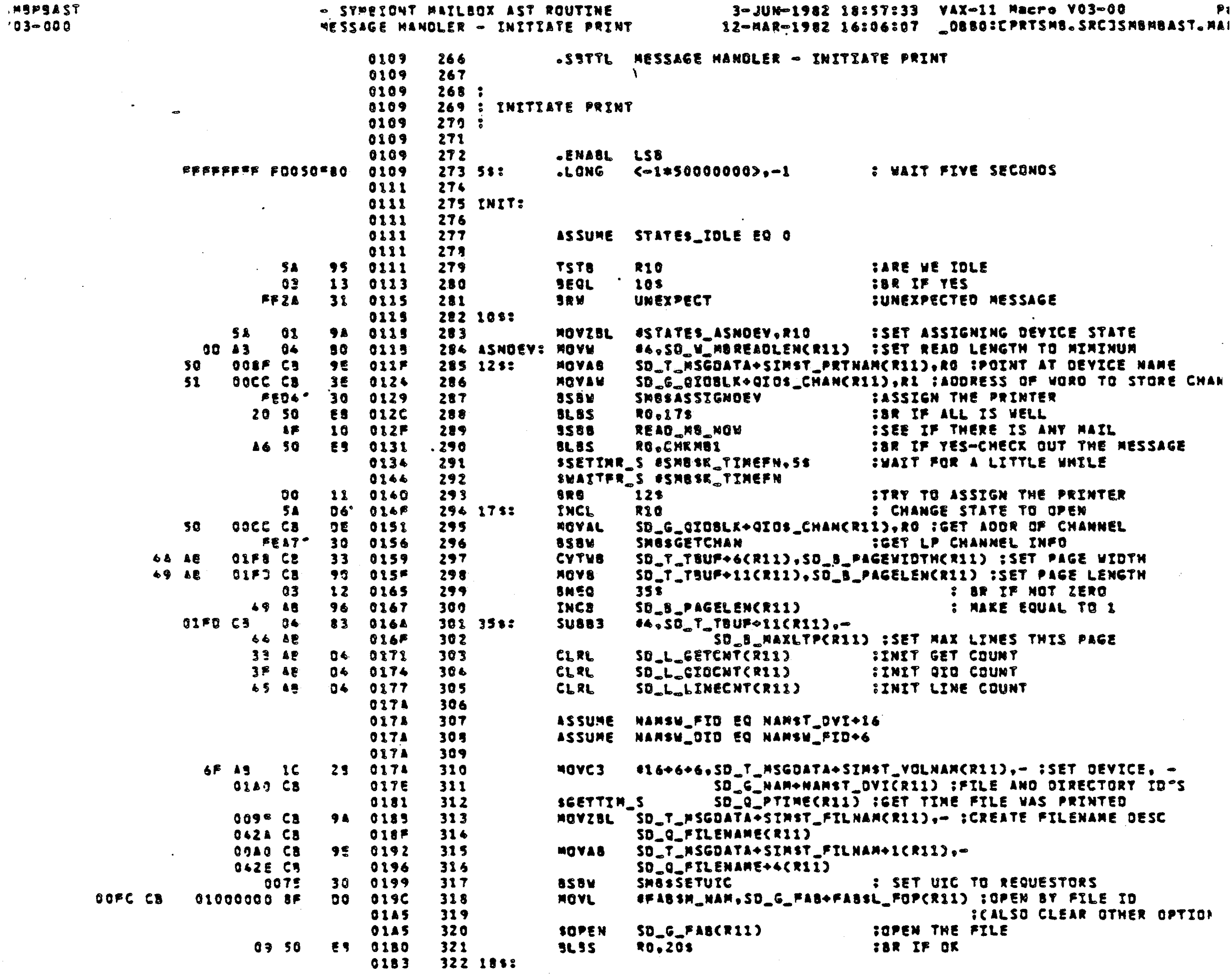
## Listing 4-4 Mailbox Ast (page 4 of 8)

```
BMBAST                           - SYMBIONT MAILBOX AST ROUTINE          3-JUN-1982 18:57:33  VAX-11 Macro V03-00          P
3-000                            MESSAGE HANDLER - RESUME               12-MAR-1982 16:06:07  _DBB0:[PRTSMB.SRC]SMBMBAST.MA

                              004F   174          .SBTTL  MESSAGE HANDLER - RESUME
                              004F   175                  .
                              004F   176 ;
                              004F   177 ; RESUME PRINTING
                              004F   178 ;
                              004F   179
                              004F   180          .ENABL  LSB
                              004F   181
                              004F   182 RESUME:
            5A    05    91    004F   183          CMPB    #STATES_EOF_CLOS,R10     ;ARE WE DONE
                  6F    13    0052   184          BEQL    30$                      ;BR IF YES - IGNORE IT
                  5A    95    0054   185          TSTB    R10                      ;ARE WE IDLE
                  6B    13    0056   186          BEQL    30$                      ;BR IF YES - IGNORE IT
            5A    06    91    0059   187          CMPB    #STATES_SUSPEND,R10      ;ARE WE SUSPENDED
                  65    12    005C   188          BNEQ    UNEXPECT                 ;BR IF NO
            5A    59    9A    005E   189          MOVZBL  R9,R10                   ;RESTORE PREVIOUS STATE
      50    55 AB       B0    0060   190          MOVW    SD_T_MSGDATA+SMB$W_REST(R11),R0 ;GET INDICATOR
                  5D    13    0064   191          BEQL    30$                      ;BR IF NO INDICATOR - JUST RESUME
                              0066   192
                              0066   193 ;
                              0066   194 ; BACKWARD SPACE FILE
                              0066   195 ;
                              0066   196
            5A    03    91    0066   197          CMPB    #STATES_FLAGPAGE,R10     ;PRINTING FLAG PAGE
                  58    13    0069   198          BEQL    30$                      ;BR IF YES - GET OUT
      50    8000 8F     B1    006B   199          CMPW    #^X8000,R0               ;IS IT TOP OF FILE
                  10    12    0070   200          BNEQ    10$                      ;BR IF NO
                              0072   201 7$:
                              0072   202          $REWIND SD_G_RAB(R11)            ;REWIND THE FILE
         39 50          E9    007D   203          BLBS    R0,20$                   ;BR IF OK
                  1F    11    0080   204          BRB     12$
                              0082   205 10$:
         49 AB          D5    0082   206          TSTL    SD_Q_TOP_FORMS(R11)      ;DO WE HAVE A TOP OF FORMS YET
                  E3    13    0085   207          BEQL    7$                       ;BR IF NO - REWIND
      48 AB       06    28    0087   208          MOVC3   #6,SD_Q_TOP_FORMS(R11),-
         0158 CB              008B   209                          SD_G_RAB+RAB$W_RFA(R11) ;SET RFA OF LAST FORM FEED
   0166 CB        02    90    008E   210          MOVB    #RAB$C_RFA,SD_G_RAB+RAB$B_RAC(R11) ;SET RFA MODE
                              0093   211          $FIND   RAB=SD_G_RAB(R11)        ;FIND THE FORM FEED RECORD
         14 50          E9    009E   212          BLBS    R0,15$                   ;BR IF $FIND OK
                              00A1   213 12$:     SETBIT  SD_V_GETERR,SD_B_ERR_FLAGS(R11) ;SET GET ERROR
                              00A5   214          SIGNAL  SHR$_RMSERROR!<4@16>,#0,R0 ;SIGNAL THE ERROR
                        05    00B4   215          RSB                              ;
                              00B5   216
                              00B5   217          ASSUME  RAB$C_SEQ EQ 0
                              00B5   218
                              00B5   219 15$:
         0166 CB        94    00B5   220          CLRB    SD_G_RAB+RAB$B_RAC(R11) ;SET ACCESS BACK TO SEQUENTIAL
                              00B9   221 20$:
43 AB    44 AB    01    81    00B9   222          ADDB3   #1,SD_B_MAXLTP(R11),SD_B_LTPCNT(R11) ;FORCE FORM FEED IF NE
                              00BF   223          SETBIT  SD_V_FFREQ,SD_B_GEN_FLAGS(R11) ; SET FF REQUIRED BIT
                        05    00C3   224 30$:     RSB                              ;
                              00C4   225
                              00C4   226          .DSABL  LSB                      ;
                              00C4   227
                              00C4   228          .DSABL  LSB
```

## Listing 4-4 Mailbox Ast (page 5 of 8)

```
                - SYMBIONT MAILBOX AST ROUTINE           3-JUN-1982 18:57:33  VAX-11 Macro V03-00         Page   7
                MESSAGE HANDLER - RESUME                12-MAR-1982 16:06:07  _DBB0:[PRTSMB.SRC]SMBMBAST.MAR;1  (1)

                          00C4    230          .ENABL  LSB
                          00C4    231
                          00C4    232 SUSPEND:
        5A    05    91    00C4    233          CMPB    #STATE$_EOF_CLOS,R10    ;ARE WE DONE
              16    13    00C7    234          BEQL    20$                     ;BR IF YES - IGNORE IT
              5A    95    00C9    235          TSTB    R10                     ;ARE WE IDLE
              12    13    00CB    236          BEQL    20$                     ;BR IF YES - IGNORE IT
        59    5A    9A    00CD    237          MOVZBL  R10,R9                  ;SAVE CURRENT STATE FOR RESUME
        5A    06    9A    00D0    238          MOVZBL  #STATE$_SUSPEND,R10     ;SET SUSPEND STATE
 50   0000'8F    3C    00D3    239          MOVZWL  #IO$_READVBLK,R0        ;SET FUNCTION CODE
              0B    10    00D8    240          BSBB    READ_MB                 ;READ THE MAILBOX
 6E   FF43 CF    9E    00DA    241 CHKMB1: MOVAB   CHK_MBREAD,(SP)         ; SET NEW RETURN ADDRESS
                    05    00DF    242 20$:    RSB                             ;
                          00E0    243
                          00E0    244          .DSABL  LSB
                          00E0    245
                          00E0    246 ;
                          00E0    247 ; LOCAL SUBROUTINE TO READ THE MAILBOX
                          00E0    248 ;
                          00E0    249 ; INPUT - AT READ_MB WITH  R0 = FUNCTION CODE
                          00E0    250 ;
                          00E0    251          .ENABL  LSB
                          00E0    252 READ_MB_NOW:                            ; ENTER FOR READ-NOW
 50   0000'8F    3C    00E0    253          MOVZWL  #IO$_READVBLK!IO$M_NOW,R0 ; SET FUNCTION OF READ WITH NO WAIT
                          00E5    254 READ_MB:
        51    7E    7E    00E5    255          MOVAQ   -(SP),R1                ;CREATE SPACE FOR IOSB
                          00E8    256          $QIOW_S         -               ;READ THE MAILBOX
                          00E8    257          EFN=#SMB$K_MBEFN,-              ;EVENT FLAG
                          00E8    258          CHAN=SD_W_MBCHAN(R11),-         ;MAILBOX CHANNEL
                          00E8    259          FUNC=R0,-                       ;FUNCTION
                          00E8    260          IOSB=(R1),-                     ;I/O STATUS BLOCK
                          00E8    261          P1=SD_T_MSGDATA(R11),-          ;DATA BUFFER ADDRESS
                          00E8    262          P2=SD_W_MBREADLEN(R11)          ;READ SIZE
        50    8E    7D    0105    263          MOVQ    (SP)+,R0                ;GET I/O STATUS
                    05    0108    264 10$:    RSB
```

Listing 4-4 Mailbox Ast (page 6 of 8)



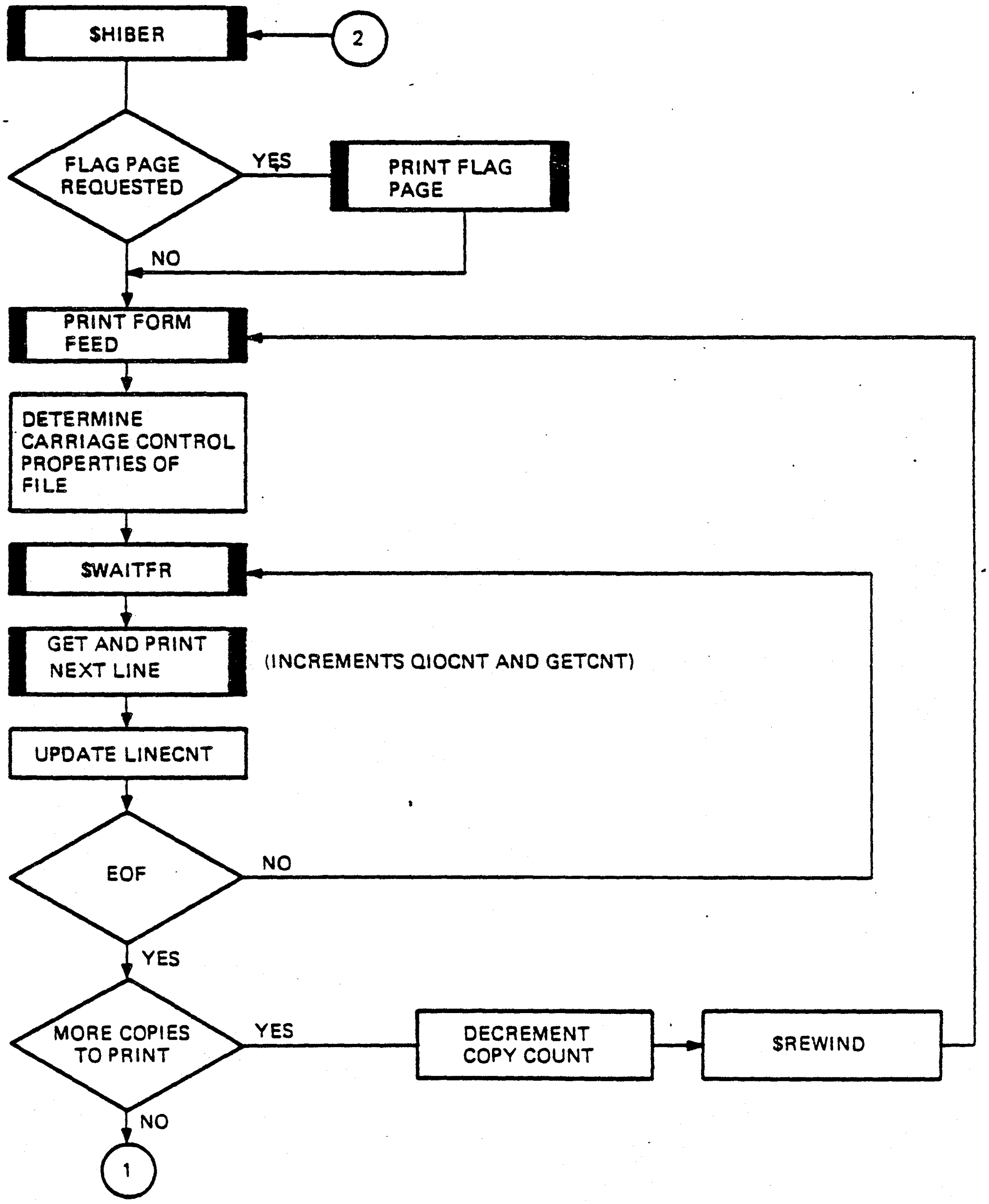
```
.MBPSAST                          - SYMBIONT MAILBOX AST ROUTINE          3-JUN-1982 18:57:33  VAX-11 Macro V03-00          P
'03-000                           MESSAGE HANDLER - INITIATE PRINT       12-MAR-1982 16:06:07  _DBB0:[PRTSMB.SRC]SMBMBAST.MA

                                  0109   266          .SBTTL  MESSAGE HANDLER - INITIATE PRINT
                                  0109   267                  \
                                  0109   268 ;
                                  0109   269 ; INITIATE PRINT
                                  0109   270 ;
                                  0109   271
                                  0109   272          .ENABL  LSB
          FFFFFFFF FD050F80       0109   273 5$:      .LONG   <-1*50000000>,-1        ; WAIT FIVE SECONDS
                                  0111   274
                                  0111   275 INIT:
                                  0111   276
                                  0111   277          ASSUME  STATES_IDLE EQ 0
                                  0111   278
                        5A    95  0111   279          TSTB    R10                     ;ARE WE IDLE
                        03    13  0113   280          BEQL    10$                     ;BR IF YES
                      FF2A    31  0115   281          BRW     UNEXPECT                ;UNEXPECTED MESSAGE
                                  0118   282 10$:
                   5A    01   9A  0118   283          MOVZBL  #STATES_ASNDEV,R10      ;SET ASSIGNING DEVICE STATE
                00 A3    04   B0  011B   284 ASNDEV:  MOVW    #4,SD_W_MBREADLEN(R11)  ;SET READ LENGTH TO MINIMUM
              50   008F CB    9E  011F   285 12$:     MOVAB   SD_T_MSGDATA+SIM$T_PRTNAM(R11),R0 ;POINT AT DEVICE NAME
              51   00CC CB    3E  0124   286          MOVAW   SD_G_QIOBLK+QIO$_CHAN(R11),R1 ;ADDRESS OF WORD TO STORE CHAN
                      FED4'   30  0129   287          BSBW    SMB$ASSIGNDEV           ;ASSIGN THE PRINTER
                     20 50    E8  012C   288          BLBS    R0,17$                  ;BR IF ALL IS WELL
                        4F    10  012F   289          BSBB    READ_MB_NOW             ;SEE IF THERE IS ANY MAIL
                     A6 50    E9  0131   290          BLBS    R0,CHKMB1               ;BR IF YES-CHECK OUT THE MESSAGE
                                  0134   291          $SETIMR_S #SMB$K_TIMEFN,5$      ;WAIT FOR A LITTLE WHILE
                                  0144   292          $WAITFR_S #SMB$K_TIMEFN
                        D0    11  014D   293          BRB     12$                     ;TRY TO ASSIGN THE PRINTER
                        5A    D6' 014F   294 17$:     INCL    R10                     ; CHANGE STATE TO OPEN
              50   00CC CB    DE  0151   295          MOVAL   SD_G_QIOBLK+QIO$_CHAN(R11),R0 ;GET ADDR OF CHANNEL
                      FEA7'   30  0156   296          BSBW    SMB$GETCHAN             ;GET LP CHANNEL INFO
         64 AB     01F8 CB    33  0159   297          CVTWB   SD_T_TBUF+6(R11),SD_B_PAGEWIDTH(R11) ;SET PAGE WIDTH
         49 AB     01F3 CB    90  015E   298          MOVB    SD_T_TBUF+11(R11),SD_B_PAGELEN(R11) ;SET PAGE LENGTH
                        03    12  0165   299          BNEQ    35$                             ; BR IF NOT ZERO
                     49 AB    96  0167   300          INCB    SD_B_PAGELEN(R11)               ; MAKE EQUAL TO 1
            01FD C3    04     83  016A   301 35$:     SUBB3   #4,SD_T_TBUF+11(R11),-
                     66 AB        016F   302                          SD_B_MAXLTP(R11) ;SET MAX LINES THIS PAGE
                     33 AB    D4  0171   303          CLRL    SD_L_GETCNT(R11)        ;INIT GET COUNT
                     3F AB    D4  0174   304          CLRL    SD_L_QIOCNT(R11)        ;INIT QIO COUNT
                     45 AB    D4  0177   305          CLRL    SD_L_LINECNT(R11)       ;INIT LINE COUNT
                                  017A   306
                                  017A   307          ASSUME  NAM$W_FID EQ NAM$T_DVI+16
                                  017A   308          ASSUME  NAM$W_DID EQ NAM$W_FID+6
                                  017A   309
            6F A9    1C       28  017A   310          MOVC3   #16+6+6,SD_T_MSGDATA+SIM$T_VOLNAM(R11),- ;SET DEVICE, -
                   01A0 CB        017E   311                          SD_G_NAM+NAM$T_DVI(R11) ;FILE AND DIRECTORY ID'S
                                  0181   312          $GETTIM_S       SD_Q_PTIME(R11) ;GET TIME FILE WAS PRINTED
                   009E CB    9A  0189   313          MOVZBL  SD_T_MSGDATA+SIM$T_FILNAM(R11),- ;CREATE FILENAME DESC
                   042A CB        018F   314                  SD_Q_FILENAME(R11)
                   00A0 CB    9E  0192   315          MOVAB   SD_T_MSGDATA+SIM$T_FILNAM+1(R11),-
                   042E C9        0196   316                  SD_Q_FILENAME+4(R11)
                      0075    30  0199   317          BSBW    SMB$SETUIC                      ; SET UIC TO REQUESTORS
   00FC CB    01000000 8F     D0  019C   318          MOVL    #FAB$M_NAM,SD_G_FAB+FAB$L_FOP(R11) ;OPEN BY FILE ID
                                  01A5   319                                                  ;(ALSO CLEAR OTHER OPTIO
                                  01A5   320          $OPEN   SD_G_FAB(R11)           ;OPEN THE FILE
                     09 50    E9  01B0   321          BLBS    R0,20$                  ;BR IF OK
                                  01B3   322 18$:
```

## Listing 4-4 Mailbox Ast (page 7 of 8)

```
                        - SYMBIONT MAILBOX AST ROUTINE          3-JUN-1982 18:57:33  VAX-11 Macro V03-00          Page   9
                        MESSAGE HANDLER - INITIATE PRINT        12-MAR-1982 16:06:07  _DBB0:[PRTSMB.SRC]SMBMBAST.MAR;1  (1)

                        01B3   323          SETBIT  SD_V_OPENERR,SD_B_ERR_FLAGS(R11) ;SET OPEN ERROR
           5A   05   9A 01B7   324          MOVZBL  #STATE$_EOF_CLOS,R10     ;SET EOF_CLOSE STATE
                29   11 01BA   325          BRB     30$                      ;GET OUT
                        01BC   326 20$:                                      ;
                        01BC   327          $CONNECT        RAB=SD_G_RAB(R11);CONNECT THE RAB
             09 50   E9 01C7   328          BLBS    R0,25$                   ;BR IF OK
                        01CA   329          ASSUME  FAB$L_STV EQ FAB$L_STS+4
                        01CA   330          ASSUME  RAB$L_STV EQ RAB$L_STS+4
           0150 C8   7D 01CA   331          MOVQ    SD_G_RAB+RAB$L_STS(R11),-  ;SAVE ERROR STATUS VALUES
           0100 C8      01CE   332                  SD_G_FAB+FAB$L_STS(R11)
                E0   11 01D1   333          BRB     18$                      ;QUIT
                        01D3   334 25$:
 0174 C8    01EC C8  9E 01D3   335          MOVAB   SD_T_SQBUF(R11),SD_G_RAB+RAB$L_RHB(R11) ;SET HEADER BUF ADDR
      0137 C8   02   91 01DA   336          CMPB    #2,SD_G_FAB+FAB$B_FSZ(R11) ;IS THE FSZ OK
                04   1E 01DF   337          BGEQU   30$                      ;BR IF YES
           0174 C8   D4 01E1   338          CLRL    SD_G_RAB+RAB$L_RHB(R11) ;DON'T GET THE RECORD HEADER
                30   10 01E5   339 30$:     BSBB    SMB$RSTUIC               ; RESTORE UIC TO DEFAULT
             48 AB   D4 01E7   340          CLRL    SD_Q_TOP_FORMS(R11)      ;SHOW NO TOP OF FORM YET FOR RESUME
     0117 C8    03   91 01EA   341          CMPB    #FAB$C_VFC,SD_G_FAB+FAB$B_RFM(R11) ;SEQUENCE NO. OR PRINT FORMAT?
                0C   12 01EF   342          BNEQ    WAKUP                    ;BR IF NOT
   06 0116 C8   02   E0 01F1   343          BBS     #FAB$V_PRN,SD_G_FAB+FAB$B_RAT(R11),WAKUP ;BR IF PRINT FILE FORMAT
                        01F7   344          CLRBIT  RAB$V_LOC,SD_G_RAB+RAB$L_ROP(R11) ;USE MOVE MODE
                        01FD   345 WAKUP:   SETBIT  SD_V_GOOD_WAKE,SD_B_GEN_FLAGS(R11) ;SET GOOD WAKE FOR MAIN CODE
                        0201   346          $WAKE_S                          ;WAKE THE SYMBIONT
                     05 020C   347          RSB                              ;
                        020D   348
                        020D   349          .DSABL  LSB                      ;
                        020D   350
                        020D   351 ;+
                        020D   352 ; SETUIC - SET UIC TO THAT OF THE REQUESTOR
                        020D   353 ; RSTUIC - RESET UIC TO NORMAL
                        020D   354 ;-
                        020D   355          .ENABL  LSB
               00010004 020D   356 DEFUIC:  .LONG   <^O1016+^O4>             ; DEFAULT UIC = [1,4]
                        0211   357
                        0211   358 SMB$SETUIC::                              ; SET UIC TO THAT OF REQUESTOR
        50   57 AB   DE 0211   359          MOVAL   SD_T_MSGDATA+SIM$L_UIC(R11),R0 ; UIC IS HERE
                04   11 0215   360          BRB     10$
                        0217   361 SMB$RSTUIC::                              ; RESTORE UIC
        50   F3 AF   9E 0217   362          MOVAB   DEFUIC,R0                ; DEFAULT UIC
                        021B   363 10$:     $CMKRNL_S  B^100$,(R0)           ; EXECUTE KERANL MODE ROUTINE
                     05 0227   364          RSB                              ;
                        0228   365
                        0229   366 ;
                        0229   367 ; KERNAL ACCESS MODE ROUTINE TO SET SYMBIONT UIC
                        0229   368 ;
                   0000 0229   369 100$:    .WORD   0                        ; ENTRY
   50   00000000'9F  D0 022B   370          MOVL    @#SCH$GL_CURPCB,R0       ; GET CURRENT PROCESS CONTROL BLOCK
      0098 C0   6C   D0 0232   371          MOVL    (AP),PCB$L_UIC(R0)       ; SET THE UIC
                50   D6 0236   372          INCL    R0                       ; MAKE AN ODD VALUE
                     04 0238   373          RET                              ;
                        0239   374
                        0239   375          .DSABL  LSB
```

## Listing 4-4 Mailbox Ast (page 8 of 8)

```
MBMBAST                      - SYMBIONT MAILBOX AST ROUTINE         3-JUN-1982 18:57:33  VAX-11 Macro V03-00
03-000                       MESSAGE HANDLER - ABORT AND EXIT       12-MAR-1982 16:06:07  _DBB0:[PRTSMB.SRC]SMBMBAST.

                   0239   377          .SBTTL  MESSAGE HANDLER - ABORT AND EXIT
                   0239   378 ;
                   0239   379 ; ABORT PRINT
                   0239   380 ;
                   0239   381
                   0239   382          .ENABL  LSB
                   0239   383
                   0239   384 ABORT:
  5A     05    91  0239   385          CMPB    #STATE$_EOF_CLOS,R10     ;ARE WE DONE
         2E    13  023C   386          BEQL    30$                      ;BR IF YES - IGNORE IT
         5A    95  023E   387          TSTB    R10                      ;ARE WE IDLE
         2A    13  0240   388          BEQL    30$                      ;BR IF YES - IGNORE IT
                   0242   389          SETBIT  SD_V_ABORT,SD_B_ERR_FLAGS(R11) ;SET ABORT PRINT BIT
                   0246   390          $CANCEL_S SD_G_QIOBLK+QIO$_CHAN(R11) ;CANCEL PRINTER I/O
      53 A8    81  0252   391          CMPW    SD_T_MSGDATA+SIM$W_MSGTYP(R11),-
         15        0255   392                  #MSG$_REQUE              ; IS THIS A REQUEUE REQUEST?
         05    12  0256   393          BNEQU   15$                      ; IF NEQ - THEN NO
                   0258   394          CLRBIT  PQR$V_DELETE,-           ; YES - CLEAR DELETE BIT
                   0258   395                  SD_T_MSGDATA+SIM$B_FLAGS(R11) ; IN FLAGS BYTE
  5A     06    91  025D   396 15$:     CMPB    #STATE$_SUSPEND,R10      ;WERE WE SUSPENDED
         03    12  0260   397          BNEQ    20$                      ;BR IF NO
  5A     59    9A  0262   398          MOVZBL  R9,R10                   ;RESTORE PREVIOUS STATE
  5A     01    91  0265   399 20$:     CMPB    #STATE$_ASNDEV,R10       ; TRYING TO ASSIGN THE PRINTER?
         02    12  0268   400          BNEQ    30$                      ; BR IF NO
         5A    D4  026A   401          CLRL    R10                      ; SET IDLE
               05  026C   402 30$:     RSB                              ;
                   026D   403
                   026D   404          .DSABL  LSB
                   026D   405
                   026D   406 ;
                   026D   407 ; SYMBIONT EXIT
                   026D   408 ;
                   026D   409
                   026D   410          .ENABL  LSB
                   026D   411
                   026D   412 EXIT:
  5A     00    91  026D   413          CMPB    #STATE$_IDLE,R10         ;ARE WE IDLE
         03    13  0270   414          BEQL    10$                      ;BR IF YES
       FDCD    31  0272   415          BRW     UNEXPECT                 ;UNEXPECTED MESSAGE
                   0275   416 10$:
                   0275   417          $DASSGN_S       SD_W_JBCCHAN(R11) ;DEASSIGN THE MGR'S MB
      0D 50    E9  0280   418          BLBC    R0,20$                   ;BR IF ERROR
                   0283   419          $DASSGN_S       SD_W_MBCHAN(R11) ;DEASSIGN MY MB
      08 50    E8  028E   420          BLBS    R0,30$                   ;BR IF OK
                   0290   421 20$:     SIGNAL  JBC$_MBDEAS,#0,R0        ;SIGNAL THE ERROR
                   029F   422 30$:     $DELMBX_S       SD_W_MBUNIT(R11) ;DELETE MY MB (JUST IN CASE)
                   02AA   423          $EXIT_S                          ;DONE - GET OUT
                   02B3   424
                   02B3   425          .DSABL  LSB
                   02B3   426
                   02B3   427          .END
```



TK-9190

Figure 4-17 Print Symbiont Main Loop

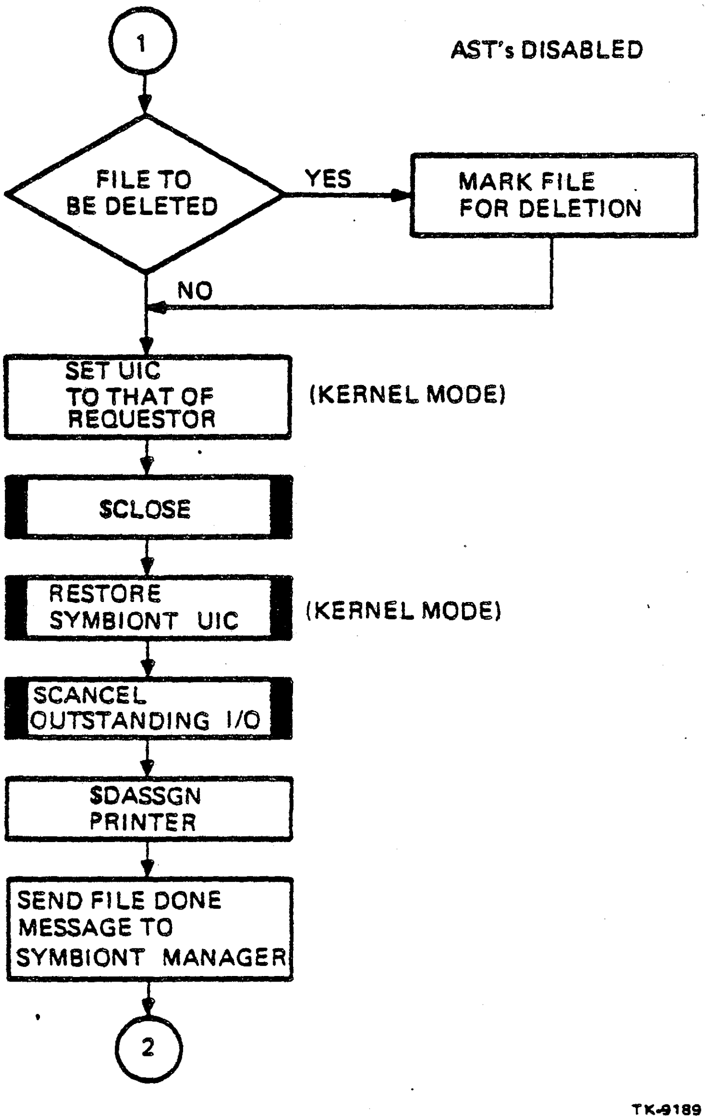
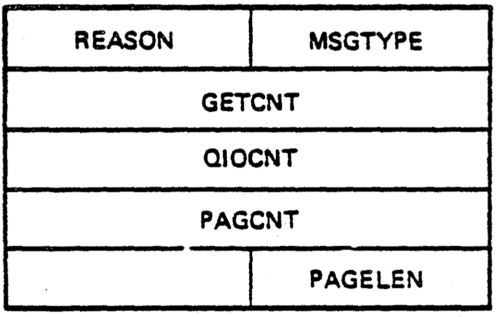


Figure 4-18 Print Symbiont Main Loop



TK-9171

Figure 4-19 FILE DONE Message send to Symbiont Manager

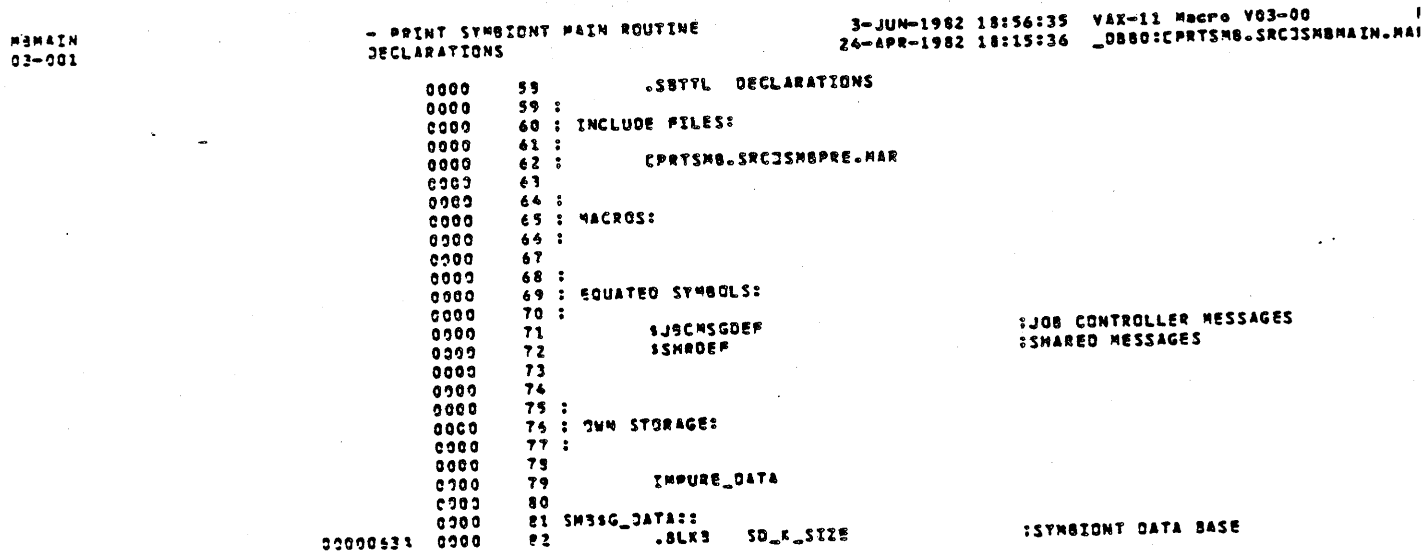
MSG TYPE=9   Symbiont Done

Reason

| | |
|---|---|
| 1 | Success |
| 4 | Aborted |
| 12 | Input Error |
| 2Ø | Print Error |
| 28 | Open Error |

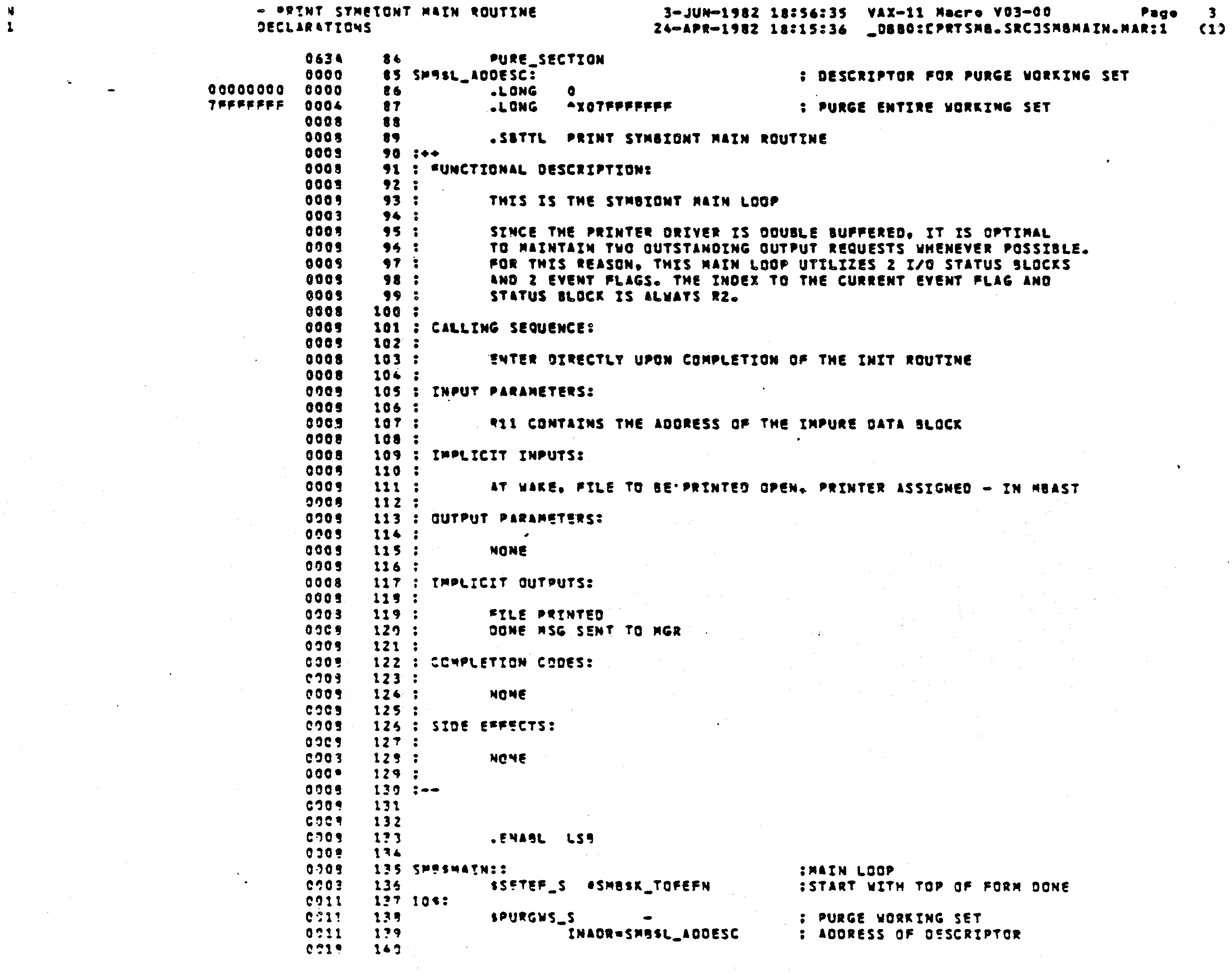
PAGCNT=LINECNT/PAGELEN

## Listing 4-5 Main Loop (page 1 of 6)



```
SMBMAIN               - PRINT SYMBIONT MAIN ROUTINE              3-JUN-1982 18:56:35  VAX-11 Macro V03-00
03-001                DECLARATIONS                              24-APR-1982 18:15:36  _DBB0:[PRTSMB.SRC]SMBMAIN.MA

                         0000             .SBTTL  DECLARATIONS
                         0000   ;
                         0000   ; INCLUDE FILES:
                         0000   ;
                         0000   ;         [PRTSMB.SRC]SMBPRE.MAR
                         0000
                         0000   ;
                         0000   ; MACROS:
                         0000   ;
                         0000
                         0000   ;
                         0000   ; EQUATED SYMBOLS:
                         0000   ;
                         0000             $JBCMSGDEF                          ;JOB CONTROLLER MESSAGES
                         0000             $SHRDEF                             ;SHARED MESSAGES
                         0000
                         0000
                         0000   ;
                         0000   ; OWN STORAGE:
                         0000   ;
                         0000
                         0000             IMPURE_DATA
                         0000
                         0000   SMB$G_DATA::
               00000633  0000             .BLKB   SD_K_SIZE                   ;SYMBIONT DATA BASE
```

Listing 4-5 Main Loop (page 2 of 6)



```
N                    - PRINT SYMBIONT MAIN ROUTINE          3-JUN-1982 18:56:35  VAX-11 Macro V03-00        Page   3
1                    DECLARATIONS                          24-APR-1982 18:15:36  _DBB0:[PRTSMB.SRC]SMBMAIN.MAR;1  (1)

                   0634            PURE_SECTION
                   0000      SMB$L_ADDESC:                            ; DESCRIPTOR FOR PURGE WORKING SET
          00000000 0000            .LONG   0
          7FFFFFFF 0004            .LONG   ^X07FFFFFFF                ; PURGE ENTIRE WORKING SET
                   0008
                   0008            .SBTTL  PRINT SYMBIONT MAIN ROUTINE
                   0008      ;++
                   0008      ; FUNCTIONAL DESCRIPTION:
                   0008      ;
                   0008      ;     THIS IS THE SYMBIONT MAIN LOOP
                   0008      ;
                   0008      ;     SINCE THE PRINTER DRIVER IS DOUBLE BUFFERED, IT IS OPTIMAL
                   0008      ;     TO MAINTAIN TWO OUTSTANDING OUTPUT REQUESTS WHENEVER POSSIBLE.
                   0008      ;     FOR THIS REASON, THIS MAIN LOOP UTILIZES 2 I/O STATUS BLOCKS
                   0008      ;     AND 2 EVENT FLAGS. THE INDEX TO THE CURRENT EVENT FLAG AND
                   0008      ;     STATUS BLOCK IS ALWAYS R2.
                   0008      ;
                   0008      ; CALLING SEQUENCE:
                   0008      ;
                   0008      ;     ENTER DIRECTLY UPON COMPLETION OF THE INIT ROUTINE
                   0008      ;
                   0008      ; INPUT PARAMETERS:
                   0008      ;
                   0008      ;     R11 CONTAINS THE ADDRESS OF THE IMPURE DATA BLOCK
                   0008      ;
                   0008      ; IMPLICIT INPUTS:
                   0008      ;
                   0008      ;     AT WAKE, FILE TO BE PRINTED OPEN, PRINTER ASSIGNED - IN MBAST
                   0008      ;
                   0008      ; OUTPUT PARAMETERS:
                   0008      ;
                   0008      ;     NONE
                   0008      ;
                   0008      ; IMPLICIT OUTPUTS:
                   0008      ;
                   0008      ;     FILE PRINTED
                   0008      ;     DONE MSG SENT TO MGR
                   0008      ;
                   0008      ; COMPLETION CODES:
                   0008      ;
                   0008      ;     NONE
                   0008      ;
                   0008      ; SIDE EFFECTS:
                   0008      ;
                   0008      ;     NONE
                   0008      ;
                   0008      ;--
                   0008
                   0008
                   0008            .ENABL  LSB
                   0008
                   0008      SMB$MAIN::                               ;MAIN LOOP
                   0008            $SETEF_S  #SMB$K_TOFEFN            ;START WITH TOP OF FORM DONE
                   0011      10$:
                   0011            $PURGWS_S       -                  ; PURGE WORKING SET
                   0011                    INADR=SMB$L_ADDESC         ; ADDRESS OF DESCRIPTOR
                   0019
```

## Listing 4-5 Main Loop (page 3 of 6)

```
SMBMAIN                       - PRINT SYMBIONT MAIN ROUTINE         3-JUN-1982 18:56:35  VAX-11 Macro V03-00
V03-001                       PRINT SYMBIONT MAIN ROUTINE          24-APR-1982 18:15:36  _DBB0:[PRTSMB.SRC]SMBMAIN

                         0015              $HIBER_S                              ;WAIT FOR SOMETHING TO DO
   EA 02 AB    01    E5  0022              BBCC    #SD_V_GOOD_WAKE,SD_B_GEN_FLAGS(R11),10$ ;BR IF SHOULDN'T
                         0027
                         0027       :
                         0027       : PRINT THE FLAG PAGE
                         0027       :
                         0027
               01    E1  0027              BBC     #PQR$V_FLAGPAG,-
      10 0035 CB         0029                      SD_T_MSGDATA+SIM$B_FLAGS(R11),30$ ;BR IF NO FLAG PAGE
      04 AB    03    90  002D              MOVB    #STATE$_FLAGPAGE,SD_B_STATE(R11) ;SET FLAG PAGE STATE
           FFCC'     30  0031              BSBW    SMB$FLAGPAGE               ;PRINT THE FLAG PAGE
         03 50       E8  0034              BLBS    R0,20$                     ;BR IF OK
          0051       31  0037              BRW     90$                        ;QUIT
          FFC3'      30  003A       20$:   BSBW    SMB$TOPOFORM               ;PRINT THE FORM FEED
                         003D
                         003D       30$:
   1F 02 AB    02    E1  003D              BBC     #SD_V_OPENERR,SD_B_ERR_FLAGS(R11),35$ ;BR IF OPENED OK
       0104 CB       DD  0042              PUSHL   SD_G_FAB+FAB$L_STV(R11) ;RMS STATUS VALUE
       0100 CB       DD  0046              PUSHL   SD_G_FAB+FAB$L_STS(R11) ;RMS STATUS CODE
       0424 CB       DF  004A              PUSHAL  SD_Q_FILENAME(R11)         ;ADDR OF FILE NAME DESCRIPTOR
               01    DD  004E              PUSHL   #1                         ;1 FAO ARG
     0004109C 8F     DD  0050              PUSHL   #<4216>!SHR$_OPENIN!4      ;CONDITION CODE
    0000'CF    05    FB  0056              CALLS   #5,W^LIB$SIGNAL            ;SIGNAL THE ERROR
          FFA2'      30  005B              BSBW    SMB$TOPOFORM               ;PRINT FORM FEED
          00F0       31  005E              BRW     90$                        ;EXIT
                         0061
                         0061       35$:
    0428 CB    01    B0  0061              MOVW    #1,SD_W_PAGE(R11)          ;INIT PAGE NUMBER
 03 0088 CB    05    E1  0066              BBC     #PQR$V_PAGHDR,SD_T_MSGDATA+SIM$B_FLAGS(R11),40$ ;BR IF N
          FF91'      30  006C              BSBW    SMB$PAGEHDRNFF             ;PRINT A HEADER / NO FORM FEED
                         006F       40$:   SETBIT  SD_V_FLGPAGDONE,SD_B_GEN_FLAGS(R11) ;SET DONE WITH FLAG
                         0073              CLRBIT  SD_V_FFREQ,SD_B_GEN_FLAGS(R11) ;CLEAR FORM FEED REQUIRED
                         0077
                         0077       :
                         0077       : CHECK IF INTERNAL CARRIAGE CONTROL
                         0077       :
                         0077
               07    93  0077              BITB    #FAB$M_CR!FAB$M_FTN!FAB$M_PRN,-
       0116 CB           0079                              SD_G_FAB+FAB$B_RAT(R11) ;IS IT INTERNAL CAR CONT
               08    12  007C              BNEQ    50$                        ;BR IF NO
       00EC CB       D4  007E              CLRL    SD_G_QIOBLK+QIO$_P4(R11);NO CARRIAGE CONTROL
                         0082              SETBIT  SD_V_INTRNLCC,SD_B_GEN_FLAGS(R11) ;SHOW INTERNAL CARRIAG
                         0086
                         0086       :
                         0086       : CHECK IF "CR-LF" CARRIAGE CONTROL
                         0086       :
                         0086
                         0086       50$:
 05 0116 CB    01    E1  0086              BBC     #FAB$V_CR,SD_G_FAB+FAB$B_RAT(R11),60$ ;BR IF NOT "CR-LF"
    00EC CB    20    9A  008C              MOVZBL  #^A\ \,SD_G_QIOBLK+QIO$_P4(R11) ;SET SINGLE SPACE CC
                         0091
                         0091       :
                         0091       : INITIALIZE THE I/O STATUS
                         0091       :
                         0091       60$:
                         0091              $SETEF_S        #SMB$K_LPEFN0   ;INITIALLY SET THE EVENT FLAGS,
                         009A              $SETEF_S        #SMB$K_LPEFN1   ;
```

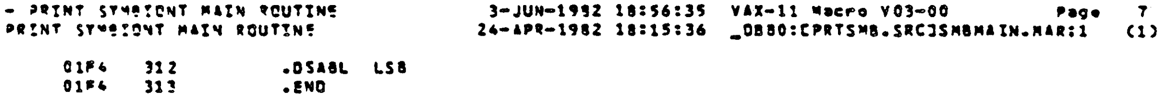
## Listing 4-5 Main Loop (page 4 of 6)

```
IN                      - PRINT SYMBIONT MAIN ROUTINE          3-JUN-1982 18:56:35  VAX-11 Macro V03-00        Page  5
01                      PRINT SYMBIONT MAIN ROUTINE           24-APR-1982 18:15:36  _DBB0:[PRTSMB.SRC]SMBMAIN.MAR;1  (1)

           17 AB    7C  00A3              CLRQ    SD_Q_IOSB0(R11)          ;CLEAR I/O STATUS
           1F AB    7C  00A6              CLRQ    SD_Q_IOSB1(R11)          ;...
              52    D4  00A9              CLRL    R2                       ;AND INITIALIZE THE I/O STATUS INDEX
                        00AB
                        00AB     ;
                        00AB     ; MAIN I/O LOOP
                        00AB     ;
     04 AB    04    90  00AB              MOVB    #STATE$_GET_PRIN,SD_B_STATE(R11) ;SET GET/PRINT STATE
                        00AF
                        00AF     70$:
  50    52    01    C1  00AF              ADDL3   #SMB$K_LPEFNO,R2,R0      ;GET EVENT FLAG NUMBER
                        00B3              $WAITFR_S R0                     ;WAIT FOR THE I/O TO COMPLETE
54 02 AB      00    E0  00BC              BBS     #SD_V_ABORT,SD_B_ERR_FLAGS(R11),80$ ;BR IF ABORTING
 50    17 AB42      7E  00C1              MOVAQ   SD_Q_IOSB0(R11)[R2],R0   ;GET THE IOSB ADDRESS
              60    B5  00C5              TSTW    (R0)                     ;DID IT COMPLETE?
              39    13  00C9              BEQL    77$                      ;BR IF NO (OR 1ST I/O)
           15 60    E8  00CA              BLBS    (R0),75$                 ;BR IF OK
                        00CD              SETBIT  SD_V_PRINTERR,SD_B_ERR_FLAGS(R11) ;SET PRINT ERROR
                        00D1              SIGNAL  JBC$_PRINTOUT,#0,(R0)    ;SIGNAL THE ERROR
              39    11  00E0              BRB     80$                      ;QUIT
                        00E2
  51    04 A0       9A  00E2     75$:     MOVZBL  4(R0),R1                 ;GET #LINES PRINTED
     45 AB    51    C0  00E6              ADDL    R1,SD_L_LINECNT(R11)     ;ADD LINES PRINTED TO TOTAL
14 03 AB      03    E0  00EA              BBS     #SD_V_FFREQ,SD_B_GEN_FLAGS(R11),77$ ;BR IF LAST I/O WAS A FF
              51    D7  00EF              DECL    R1                       ;ALREADY ASSUMED ONE LINE
     43 AB    51    80  00F1              ADDB    R1,SD_B_LTPCNT(R11)      ;ADD TO LINES THIS PAGE COUNT
49 AB      43 AB    91  00F5     76$:     CMPB    SD_B_LTPCNT(R11),SD_B_PAGELEN(R11) ;# LINES GTR THAN PAGE SIZE?
              07    1B  00FA              BLEQU   77$                      ;BR IF NO
43 AB      49 AB    82  00FC              SUBB    SD_B_PAGELEN(R11),SD_B_LTPCNT(R11) ;NORMALIZE
              F2    11  0101              BRB     76$                      ;CHECK AGAIN
              0B    93  0103     77$:     BITB    #SD_M_GETERR!SD_M_PRINTERR!SD_M_ABORT,- ;ANY ERRORS?
           02 AB        0105                      SD_B_ERR_FLAGS(R11)
              12    12  0107              BNEQ    80$                      ;BR IF YES
                        0109
     04 AB    05    91  0109              CMPB    #STATE$_EOF_CLOS,SD_B_STATE(R11) ;ARE WE DONE?
              0C    13  010E              BEQL    80$                      ;BR IF YES
                        0110              CLRBIT  SD_V_FFREQ,SD_B_GEN_FLAGS(R11) ;CLEAR FF REQUIRED
           FEEA'    30  0113              BSBW    SMB$GET                  ;GET AND PRINT THE NEXT LINE
     52       01    8C  0116              XORB    #1,R2                    ;TOGGLE I/O STATUS INDEX
              94    11  0119              BRB     70$                      ;GET AND PRINT NEXT LINE
                        0119
                        0119     ;
                        0119     ; FILE PRINTED, SET PAPER TO TOP OF PAGE, AND SEE IF ANY MORE COPIES
                        0119     ;
                        0119
                        0119     80$:     $SETIMR_S  #SMB$K_TOFEFN,B^85$  ;SET TIMER IN CASE PRINTER IS BROKEN
           FED2'    30  0129              BSBW    SMB$TOPOFORM             ;PUT PAPER AT TOP OF FORM
                        012E              $CANTIM_S                        ;CANCEL TOP OF FORM TIMER
        02 AB       95  0139              TSTB    SD_B_ERR_FLAGS(R11)      ;ANY ERRORS
           20       12  013C              BNEQ    90$                      ;BR IF ERRORS - CAN'T BE OPEN ERROR
      009C CB       97  013E              DECB    SD_T_MSGDATA+SMB$B_FILCOPY(R11) ;SUBT 1 FOR THIS COPY
              1A    1B  0142              BLEQU   90$                      ;BR IF DONE - CAN'T BE ANY ERRORS
                        0144              $REWIND SD_G_RAB(R11)            ;REWIND THE FILE FOR NEXT COPY
     04 AB    02    90  014F              MOVB    #STATE$_OPEN,SD_B_STATE(R11) ;RESET STATE TO OPEN
           FEE7     31  0153              BRW     30$                      ;PRINT NEXT COPY
                        0156
   FFFFFFFF FD050F80    0156     85$:     .LONG   <-1*50000000>,-1         ;5 SECOND DELTA TIME
```

Listing 4-5 Main Loop (page 5 of 6)

```
SMBMAIN                    - PRINT SYMBIONT MAIN ROUTINE          3-JUN-1982 18:56:35  VAX-11 Macro V03-00
V03-001                    PRINT SYMBIONT MAIN ROUTINE           24-APR-1982 18:15:36  _DBB0:[PRTSMB.SRC]SMBMAIN

                           015E   255
                           015E   256 ;
                           015E   257 ; FILE PRINTING DONE
                           015E   258 ;
                           015E   259
                           015E   260 90$:    $SETAST_S  #0                    ;DISABLE ASTS FOR SURE
                           0167   261
            00FA CB    B5  0167   262         TSTW    SD_G_FAB+FAB$W_IFI(R11) ; HAS OPEN BEEN DONE?
                 32    13  016B   263         BEQL    120$                    ; BR IF NO
                           016D   264         CLRBIT  FAB$V_DLT,SD_G_FAB+FAB$L_FOP(R11) ;CLEAR THE DELETE BIT
            0186 CB    B5  0173   265         TSTW    SD_G_NAM+NAM$W_DID(R11) ;FILE HAVE A DIRECTORY?
                 05    13  0177   266         BEQL    112$                    ;IF NO-DELETE IF REQUESTED EVEN
      0F 02 AB   00    E0  0179   267         BBS     #SD_V_ABORT,SD_B_ERR_FLAGS(R11),115$ ;BR IF ABORTING
    09 0088 CB   00    E1  017E   268 112$:   BBC     #PQR$V_DELETE,SD_T_MSGDATA+SIM$B_FLAGS(R11),115$ ;BR IF
                           0184   269         SETBIT  FAB$V_DLT,SD_G_FAB+FAB$L_FOP(R11) ; TELL RMS TO DELETE T
              FE73'    30  018A   270         BSBW    SMB$SETUIC              ;SET UIC TO REQUESTOR'S
                           018D   271 115$:   $CLOSE  SD_G_FAB(R11)           ;CLOSE THE FILE
            014A CB    B4  0198   272         CLRW    SD_G_RAB+RAB$W_ISI(R11) ; EFFECT A FAST DISCONNECT AFTER
              FE61'    30  019C   273         BSBW    SMB$RSTUIC              ;BE SURE WE'RE RUNNING AT RIGHT
                           019F   274
                           019F   275 ;
                           019F   276 ; SET ENDING STATUS
                           019F   277 ;
                           019F   278
         53      0C    9A  019F   279 120$:   MOVZBL  #MOD$_INPERR,R3         ;ASSUME INPUT ERROR
   13 02 AB      01    E0  01A2   280         BBS     #SD_V_GETERR,SD_B_ERR_FLAGS(R11),130$ ; Br if input erro
         53      1C    9A  01A7   281         MOVZBL  #MOD$_OPNERR,R3         ; Assume open error
   13 02 AB      02    E0  01AA   282         BBS     #SD_V_OPENERR,SD_B_ERR_FLAGS(R11),130$  ; Br if open err
         53      14    9A  01AF   283         MOVZBL  #MOD$_PRTERR,R3         ;ASSUME PRINT ERROR
   08 02 AB      03    E0  01B2   284         BBS     #SD_V_PRINTERR,SD_B_ERR_FLAGS(R11),130$ ;BR IF PRINT ERR
         53      04    9A  01B7   285         MOVZBL  #MOD$_ABORT,R3          ;ASSUME ABORT
   03 02 AB      00    E0  01BA   286         BBS     #SD_V_ABORT,SD_B_ERR_FLAGS(R11),130$ ;BR IF ABORT
         53      01    9A  01BF   287         MOVZBL  #MOD$_SUCCESS,R3        ;SET SUCCESS
                           01C2   288
                           01C2   289 ;
                           01C2   290 ; DO FINAL CLEAN UP
                           01C2   291 ;
                           01C2   292
                           01C2   293 130$:
              02 AB    B4  01C2   294         CLRW    SD_B_ERR_FLAGS(R11)     ;RESET BOTH GROUPS OF FLAGS
              0F AB    B5  01C5   295         TSTW    SD_Q_TOFIOSB(R11)       ;DID TOF COMPLETE OR TIMEOUT?
                 0C    12  01C8   296         BNEQ    140$                    ;BR IF IO COMPLETED
                           01CA   297         $CANCEL_S  SD_G_QIOBLK+QIO$_CHAN(R11) ;ABORT THE IO IF NOT DONE
                           01D6   298 140$:   $DASSGN_S  SD_G_QIOBLK+QIO$_CHAN(R11) ;DEASSIGN THE PRINTER
                           01E2   299
                           01E2   300 ;
                           01E2   301 ; SEND DONE MESSAGE TO MGR - R3 HAS STATUS
                           01E2   302 ;
                           01E2   303
                           01E2   304
                           01E2   305         ASSUME  STATE$_IDLE EQ 0
                           01E2   306
              04 AB    94  01E2   307         CLRB    SD_B_STATE(R11)         ;SET IDLE STATE
              FE18'    30  01E5   308         BSBW    SMB$FILE_DONE           ;SEND DONE MSG TO MGR
                           01E8   309         $SETAST_S  #1                   ;ENABLE ASTS
               FE14    31  01F1   310         BRW     SMB$MAIN                ;GO AGAIN
                           01F4   311
```

Listing 4-5 Main Loop (page 6 of 6)



```
- PRINT SYMBIONT MAIN ROUTINE            3-JUN-1982 18:56:35  VAX-11 Macro V03-00          Page   7
PRINT SYMBIONT MAIN ROUTINE             24-APR-1982 18:15:36  _DBB0:[PRTSMB.SRC]SMBMAIN.MAR;1   (1)

     01F4    312          .DSABL  LSB
     01F4    313          .END
```

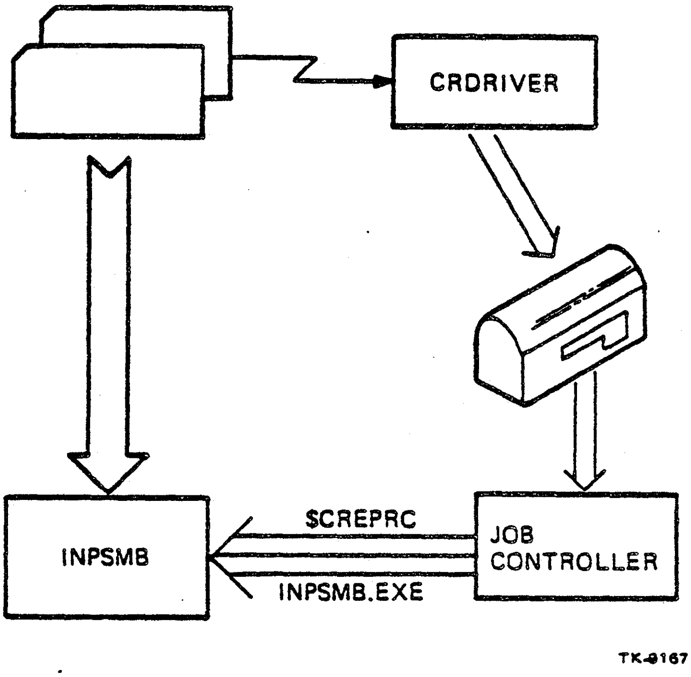
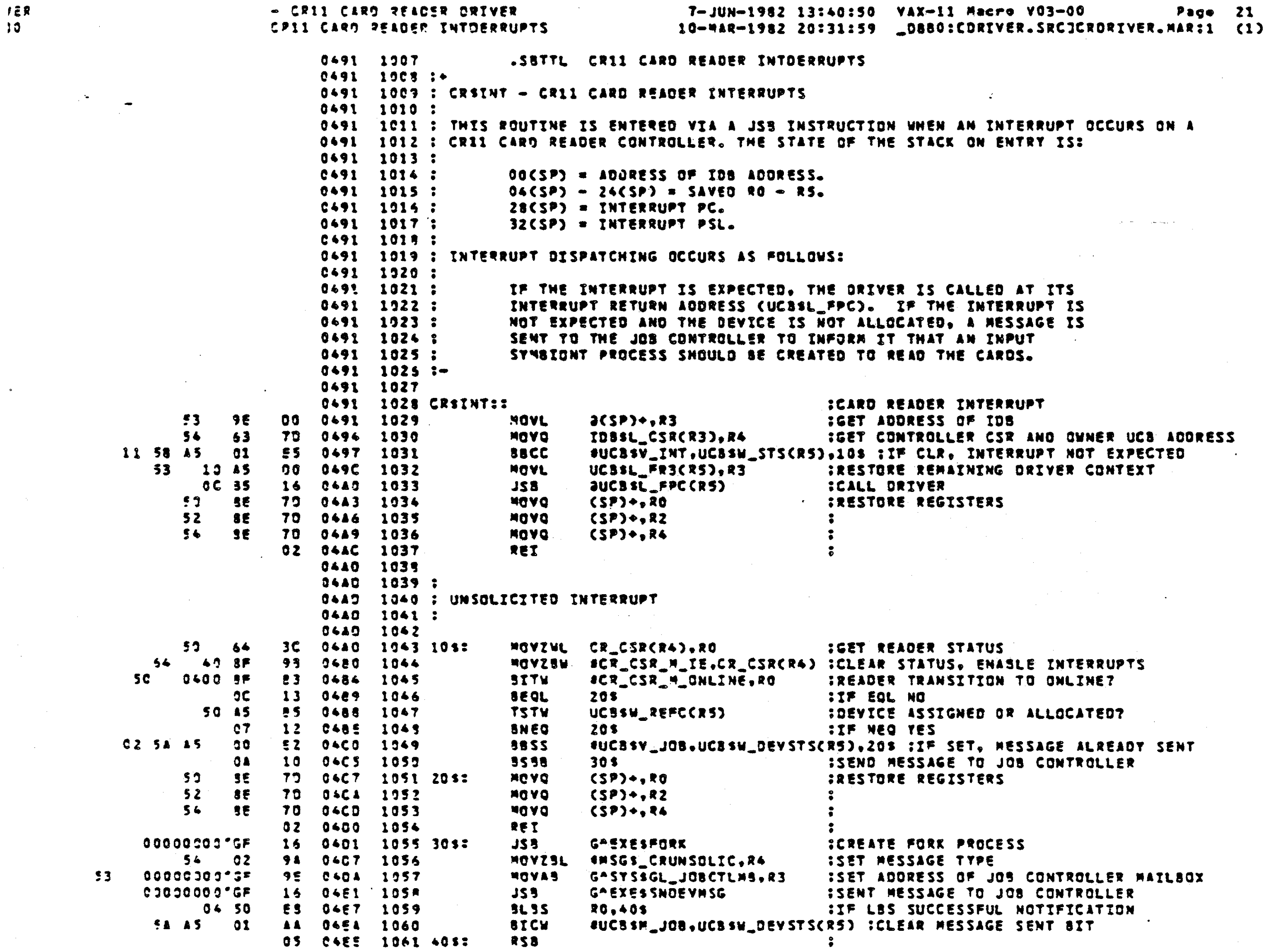


Figure 4-2Ø Input Symbiont

- Card reader activated, generates interrupt
- CRDRIVER sends message to Job Controller's Mailbox
- Job Controller issues a $CREPRC (using INPSMB.EXE)
- INPSMB.EXE issues $QIO's to the card reader

## Listing 4-6 Card Reader Interrupt Routine



```
IER                          - CR11 CARD READER DRIVER                    7-JUN-1982 13:40:50  VAX-11 Macro V03-00          Page  21
10                           CR11 CARD READER INTDERRUPTS                10-MAR-1982 20:31:59  _DBB0:[DRIVER.SRC]CRDRIVER.MAR;1  (1)

                               0491                .SBTTL  CR11 CARD READER INTDERRUPTS
                               0491      ;+
                               0491      ; CR$INT - CR11 CARD READER INTERRUPTS
                               0491      ;
                               0491      ; THIS ROUTINE IS ENTERED VIA A JSB INSTRUCTION WHEN AN INTERRUPT OCCURS ON A
                               0491      ; CR11 CARD READER CONTROLLER. THE STATE OF THE STACK ON ENTRY IS:
                               0491      ;
                               0491      ;         00(SP) = ADDRESS OF IDB ADDRESS.
                               0491      ;         04(SP) - 24(SP) = SAVED R0 - R5.
                               0491      ;         28(SP) = INTERRUPT PC.
                               0491      ;         32(SP) = INTERRUPT PSL.
                               0491      ;
                               0491      ; INTERRUPT DISPATCHING OCCURS AS FOLLOWS:
                               0491      ;
                               0491      ;         IF THE INTERRUPT IS EXPECTED, THE DRIVER IS CALLED AT ITS
                               0491      ;         INTERRUPT RETURN ADDRESS (UCB$L_FPC).  IF THE INTERRUPT IS
                               0491      ;         NOT EXPECTED AND THE DEVICE IS NOT ALLOCATED, A MESSAGE IS
                               0491      ;         SENT TO THE JOB CONTROLLER TO INFORM IT THAT AN INPUT
                               0491      ;         SYMBIONT PROCESS SHOULD BE CREATED TO READ THE CARDS.
                               0491      ;-
                               0491
                               0491      CR$INT::                                            ;CARD READER INTERRUPT
              53    9E    D0   0491                MOVL     @(SP)+,R3                  ;GET ADDRESS OF IDB
              54    63    7D   0494                MOVQ     IDB$L_CSR(R3),R4           ;GET CONTROLLER CSR AND OWNER UCB ADDRESS
        11 58 A5    01    E5   0497                BBCC     #UCB$V_INT,UCB$W_STS(R5),10$ ;IF CLR, INTERRUPT NOT EXPECTED
           53    10 A5    D0   049C                MOVL     UCB$L_FR3(R5),R3           ;RESTORE REMAINING DRIVER CONTEXT
                 0C B5    16   04A0                JSB      @UCB$L_FPC(R5)             ;CALL DRIVER
              50    8E    7D   04A3                MOVQ     (SP)+,R0                   ;RESTORE REGISTERS
              52    8E    7D   04A6                MOVQ     (SP)+,R2                   ;
              54    8E    7D   04A9                MOVQ     (SP)+,R4                   ;
                          02   04AC                REI                                 ;
                               04AD
                               04AD      ;
                               04AD      ; UNSOLICITED INTERRUPT
                               04AD      ;
                               04AD
              50    64    3C   04AD      10$:      MOVZWL   CR_CSR(R4),R0              ;GET READER STATUS
           64    40 8F    9B   04B0                MOVZBW   #CR_CSR_M_IE,CR_CSR(R4) ;CLEAR STATUS, ENABLE INTERRUPTS
        50     0400 8F    B3   04B4                BITW     #CR_CSR_M_ONLINE,R0        ;READER TRANSITION TO ONLINE?
                    0C    13   04B9                BEQL     20$                        ;IF EQL NO
                 50 A5    B5   04BB                TSTW     UCB$W_REFC(R5)             ;DEVICE ASSIGNED OR ALLOCATED?
                    07    12   04BE                BNEQ     20$                        ;IF NEQ YES
        02 5A A5    00    E2   04C0                BBSS     #UCB$V_JOB,UCB$W_DEVSTS(R5),20$ ;IF SET, MESSAGE ALREADY SENT
                    0A    10   04C5                BSBB     30$                        ;SEND MESSAGE TO JOB CONTROLLER
              50    8E    7D   04C7      20$:      MOVQ     (SP)+,R0                   ;RESTORE REGISTERS
              52    8E    7D   04CA                MOVQ     (SP)+,R2                   ;
              54    8E    7D   04CD                MOVQ     (SP)+,R4                   ;
                          02   04D0                REI                                 ;
          00000000'GF     16   04D1      30$:      JSB      G^EXE$FORK                 ;CREATE FORK PROCESS
              54    02    9A   04D7                MOVZBL   #MSG$_CRUNSOLIC,R4         ;SET MESSAGE TYPE
   53     00000000'GF     9E   04DA                MOVAB    G^SYS$GL_JOBCTLMB,R3       ;SET ADDRESS OF JOB CONTROLLER MAILBOX
          00000000'GF     16   04E1                JSB      G^EXE$SNDEVMSG             ;SENT MESSAGE TO JOB CONTROLLER
                 04 50    E8   04E7                BLBS     R0,40$                     ;IF LBS SUCCESSFUL NOTIFICATION
           5A A5    01    AA   04EA                BICW     #UCB$M_JOB,UCB$W_DEVSTS(R5) ;CLEAR MESSAGE SENT BIT
                          05   04EE      40$:      RSB                                 ;
```

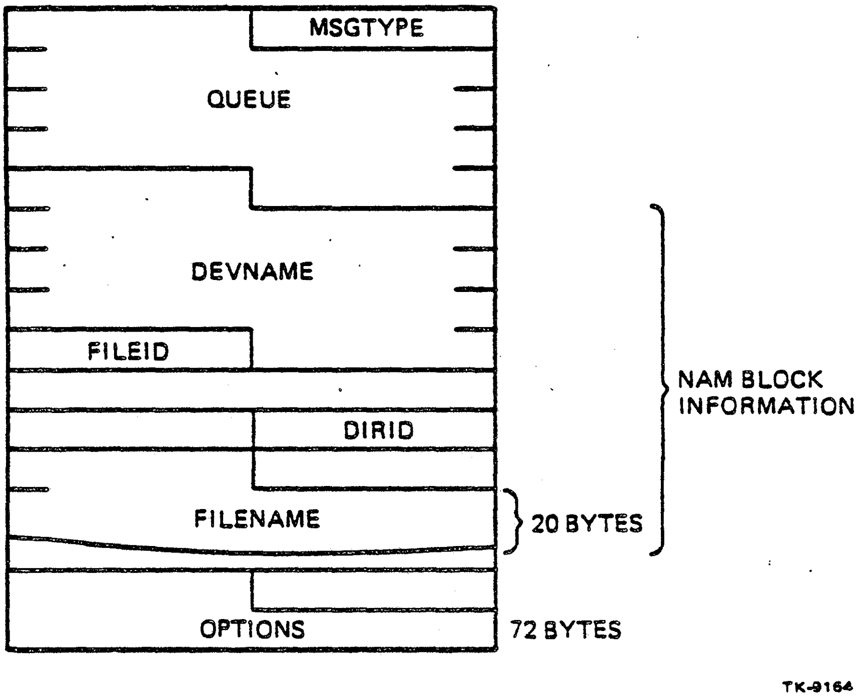


Figure 4-21 Enter BATCH JOB Message

| | |
|---|---|
| MSGTYPE=8 | ENTER BATCH JOB |
| QUEUE | Batch Queue Name |
| FILENAME | Job Name |
| OPTIONS | From $JOB Card |

## Alternate Input Symbionts

1. Create a separate process
2. Allocate device
3. Assign channel to device with associated mailbox
4. Issue QIO to mailbox that activates an AST
5. $HIBER

# WRITING AN APPLICATIONS MIGRATION EXECUTIVE

## INTRODUCTION

The VAX hardware is capable of executing either native mode instructions, or PDP-11 instructions in compatibility mode (CM). A bit in the Processor Status Longword (PSL) determines which instruction set is being used. The main reason for providing a compatibility mode is to allow programs written on PDP-11 systems to run (without modification) on VMS systems.

However, most PDP-11 programs will need assistance to run correctly on a VMS system, since they will typically request operations to be performed by the operating system (e.g., to perform an I/O operation). VMS will not be able to directly recognize the operations being requested, since the requests will be in the form of executive directives understandable by a PDP-11 operating system (like RSX-11M). A translator (emulator) is needed to convert these requests into VMS system service calls. An Applications Migration Executive (AME) serves the role of either emulating a PDP-11 operating system, and performing the requested function, or converting the requested function into an equivalent VMS system service call.

An AME is a native mode image that can issue VMS system services. It is run in the context of the process performing CM operations. It is the responsibility of the AME to load the PDP-11 program into part of its virtual address space. VMS only loads the AME, and the AME must load the CM image. Two images are therefore present in the process's context at the same time, the CM image, containing the program the user wants run, and the native mode AME image, serving as the interface between the user program and VMS. Control is transferred back and forth between these two images whenever intervention by the AME is required.

This module explains the basic details about CM exceptions necessary to write an AME for a PDP-11 operating system. The services provided by VAX/VMS to aid the writer of an AME also will be explained.

This module assumes that you are familiar with at least one PDP-11 operating system so that the functions performed by any AME can be understood. However, the module makes no attempt to explain the workings of any PDP-11 operating system, or how image files are structured by the linkers or task-builders of such systems. Rather, this module assumes you already understand the operating system that will be emulated.

# WRITING AN APPLICATIONS MIGRATION EXECUTIVE

## OBJECTIVES

A specialist with a system level understanding of a PDP-11 operating system will be able to:

- Write a program that is capable of reading into the low 64K of PØ space an image (or task or program) that was created for the target PDP-11 operating system.
- Establish a compatibility mode exception service routine to handle the various exceptions that can occur from compatibility mode.
- Write an exception service routine that can distinguish the various compatibility mode exceptions, and dispatch control accordingly.

## RESOURCES

AME source code listings

VMS source code listings

## Overview of Compatibility Mode

- Environment for execution of non-privileged RSX-11M programs
- Hardware compatibility provided by VAX processor being able to execute subset of the PDP-11 instruction set
- Software compatibility provided by several programs that emulate RSX-11M operating system environment
  - The RSX-11M AME allows non-privileged tasks to execute without change
  - The MCR command language provided as an alternative to DCL.
  - File compatibility is provided at both the record and volume levels:
    - o The record structure of RMS-11 files is identical to the record structure of VAX-11 RMS files
    - o An ACP is provided on VMS to service Files-11 volumes which support the On-Disk Structure Level 1, ODS-1

## PDP-11 Instructions That A Compatibility Mode Program Can Execute

- All PDP-11 non-privileged instructions, except MARK (including Extended Instruction SET, EIS)

| Opcode (octal) | Mnemonic | Opcode (octal) | Mnemonic |
|---|---|---|---|
| ØØØØØ2 | RTI | .Ø63DD | ASL(B) |
| ØØØØØ6 | RTI | ØØ65SS | MFPI* |
| ØØØ1DD | JMP | ØØ66DD | MTPI* |
| ØØØ2ØR | RTS | 1Ø65SS | MFPD* |
| ØØØ24Ø-ØØØ277 | Condition Codes | 1Ø66DD | MTPD* |
| ØØØ3DD | SWAB | ØØ67DD | SXT |
| ØØØ4ØØ-ØØ3777 | Branches | Ø7ØRSS | MUL |
| 1ØØØØØ-1Ø3777 | Branches | Ø71RSS | DIV |
| ØØ4RDD | JSR | Ø72RSS | ASH |
| .Ø5ØDD | CLR(B) | Ø73RSS | ASHC |
| .Ø51DD | COM(B) | Ø74RSS | XOR |
| .Ø52DD | INC(B) | Ø77RNN | SOB |
| .Ø53DD | DEC(B) | .1SSDD | MOV(B) |
| .Ø54DD | NEG(B) | .2SSDD | CMP(B) |
| .Ø55DD | ADC(B) | .3SSDD | BIT(B) |
| .Ø56DD | SBC(B) | .4SSDD | BIC(B) |
| .Ø57DD | TST(B) | .5SSDD | BIS(B) |
| .Ø6ØDD | ROR(B) | Ø6SSDD | ADD |
| .Ø61DD | ROL(B) | 16SSDD | SUB |
| .Ø62DD | ASR(B) | | |

(*) These instructions execute exactly as they would on a PDP-1 in user mode with Instruction and Data space overmapped. Mor specifically, they ignore the previous access level and act lik PUSH and POP instructions referencing the current stack.

PDP-11 Compatibility Mode Trap Instructions

| Opcode (octal) | Mnemonic |
|---|---|
| 000003 | BPT |
| 000004 | IOT |
| 104000-104377 | EMT |
| 104400-104777 | TRAP |

- Execution of any of these instructions results in a compatibility mode exception

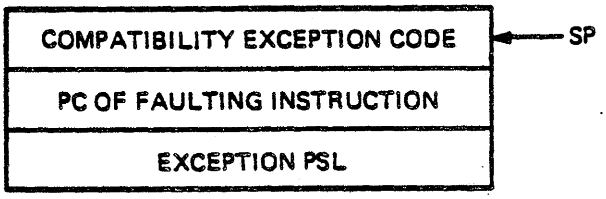
PDP-11 Compatibility Mode Reserved Instructions

| Opcode (octal) | Mnemonic |
|---|---|
| 000000 | HALT |
| 000001 | WAIT |
| 000005 | RESET |
| 00023X | SPL |
| 0064XX | MARK |
| 07500R | FADD (FIS) |
| 07501R | FSUB (FIS) |
| 07502R | FMUL (FIS) |
| 07503R | FDIV (FIS) |
| 17XXXX | FP11 Floating Point Instructions |

- These instructions are unavailable to PDP-11 programs
- Execution of these instructions results in a Reserved Instruction Compatibility Mode Exception.

## Compatibility Mode Exceptions

- Generated when program executes an instruction that would result in a trap on the PDP-11.
- Hardware pushes PSL and PC on kernel stack, along with code that identifies the type of compatibility mode exception



TK-9168

- The types of compatibility mode exceptions are:

| Code | Reason For Compatibility Exception |
|---|---|
| 0 | Reserved Instruction Execution |
| 1 | BPT Instruction Executed |
| 2 | IOT Instruction Executed |
| 3 | EMT Instruction Executed |
| 4 | TRAP Instruction Executed |
| 5 | Illegal Instruction Executed |
| 6 | Odd Address Trap |
| 7 | TBIT Trap |

## The RSX-11M AME

- Integral part of the VAX/VMS system
  - Native mode image SYS$SYSTEM:RSX.EXE
    - Shareable parts of the AME

      SYS$SHARE:RSXSHR.EXE<br>
      SYS$SHARE:RSXUSR.EXE

  - Invoked by image activator
  - Allows non-privileged RSX-11M tasks to execute on VMS systems without change
  - Supplies an environment that simulates the RSX-11M operating system
- Basic functions
  - Initiates RSX-11M task
  - Establishes exception handler
  - Responds to compatibility mode exceptions
    - Identifies type of exception
    - Acts accordingly
  - Responds to native mode exceptions (e.g. access violations)
- Servicing EMT instruction special case
  - User-generated trap, or
  - Executive directive (EMT 377)
    - Can perform service itself
      - For example, Get Task Parameters, GTSK$
    - Or request service from VMS
      - For example, RSX-11 QIO$ executive directive transformed into equivalent VMS $QIO system service

## Invoking an AME

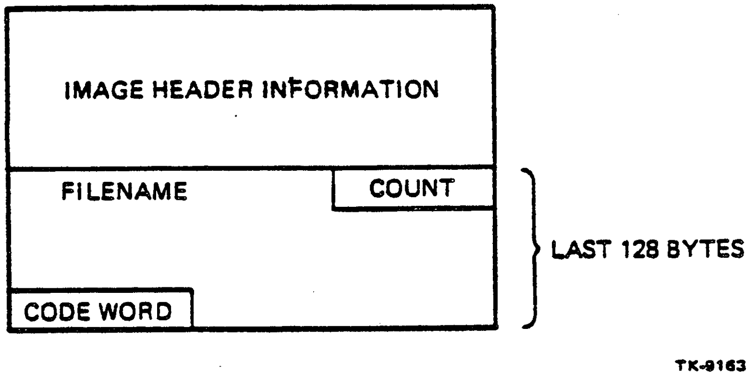
- The RSX-11M AME activated automatically
  - By image activator
  - In response to $RUN program
  - Using information in image file
  - Information placed there by task builder
  - Distinguishes native mode and compatibility mode images

- User-Written AME's can be activated in the following ways:
  - Could $RUN user-written-AME
    - AME would prompt for compatibility mode image name
  - Could use foreign command to invoke AME
    - $MYAME :== "$SYS$SYSTEM:MYAME.EXE"

      (in system-wide LOGIN.COM)
    - User types $MYAME MYPROG
      - MYAME invoked, it could use LIB$GET_FOREIGN to pick up image name (MYPROG)
  - Could add command to DCL tables to invoke AME
    - Would issue CLI callback to obtain image name (CLI$GET_VALUE)
  - Could place information in image file identifying AME

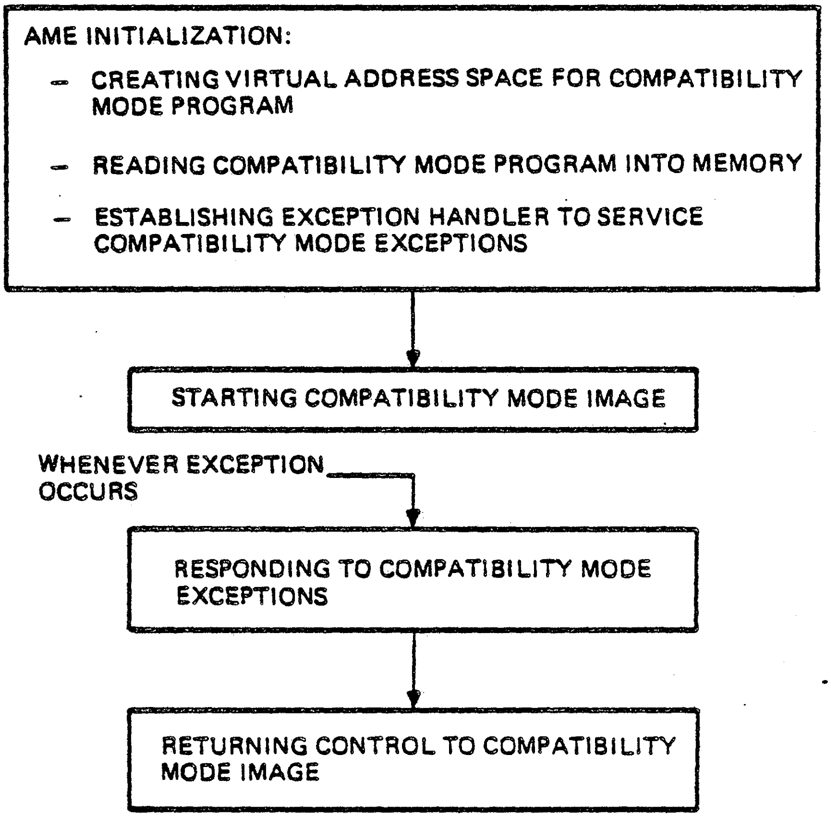
## Identifying AME in Image File

First Block in Image File



| Code Word Value | Image to Activate |
|---|---|
| <Ø | Native mode image in image file (VAX-11 Linker uses -1) |
| Ø | SYS$SYSTEM:RSX.EXE (RSX-11M AME) |
| 1 | SYS$SYSTEM:BPA.EXE (currently not used) |
| 2 | Filename specified by last 128 bytes (less last word used to hold code)<br>Filename specified as counted ASCII string<br>File could be an AME (or any other native mode image)<br>Name of image file user specified on RUN command stored as counted ASCII string at CTL$AG_CMEDATA |

## General AME Flow

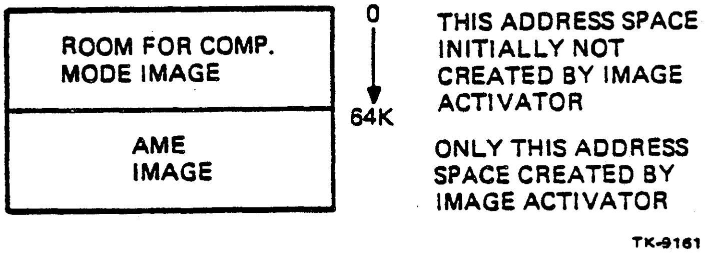

AME INITIALIZATION:
– CREATING VIRTUAL ADDRESS SPACE FOR COMPATIBILITY MODE PROGRAM
– READING COMPATIBILITY MODE PROGRAM INTO MEMORY
– ESTABLISHING EXCEPTION HANDLER TO SERVICE COMPATIBILITY MODE EXCEPTIONS
STARTING COMPATIBILITY MODE IMAGE
WHENEVER EXCEPTION OCCURS
RESPONDING TO COMPATIBILITY MODE EXCEPTIONS
RETURNING CONTROL TO COMPATIBILITY MODE IMAGE

TK-9160

## Creating Virtual Address Space For The CM Program

- AME image typically linked with base address of 64K.

```
$LINK AME, BASE.OPT/OPTIONS   where

BASE.OPT contains BASE=%X1ØØØØ ! hex 64K
```


ROOM FOR COMP. MODE IMAGE
AME IMAGE
0
64K
THIS ADDRESS SPACE INITIALLY NOT CREATED BY IMAGE ACTIVATOR
ONLY THIS ADDRESS SPACE CREATED BY IMAGE ACTIVATOR
TK-9161

- Leave 64K for compatibility mode image since that is maximum address space for any PDP-11 image.
- Must leave low addresses (starting at Ø) free, since compatibility mode image will need to use those addresses.
- AME typically creates virtual address space using $CRETVA system service:
  - Amount created depends on environment being simulated
  - Must create enough space to hold entire image
  - Size of image is system dependent, and usually put into image file by that system's linker (or task builder)

## Reading Compatibility Mode Program Into Memory

- Once address space created, can use $QIO calls or RMS block reads.
  - Requires understanding of how image file structured by system's linker or task builder.
- If compatibility mode program contains overlays:
  - AME need only read root segment into memory
  - Overlay code that is part of compatibility mode image contains the calls necessary to read in the overlay segments as they are referenced
- Alternative Method to using $CRETVA to create virtual address space, and reading in image:
  - Use $CRMPSC system service
    - Creates required address space
    - Sets up the page tables in such a way that all the input from the image file is performed by the pager.
  - Cannot be used if overlays are involved

## Establishing The Exception Handler

- Two Options

  - Normal Condition Handler

    ```
    CALL LIB$ESTABLISH (handler_address)

    or  MOVAL handler, (FP)     ; MACRO only
    ```

    - Must be established for handling non-compatibility mode exceptions

      - For example, access violations

    - May or may not want to service compatibility mode exceptions in this handler as well

  - Special Handler for Compatibility Mode Exceptions

    ```
    $DCLCMH system service     (specify TYPE=#1)
    ```

    - Normal exception dispatching bypassed
    - Control immediately passed to handler when compatibility mode exception occurs
    - Handler address stored in Pl space

      ```
      CTL$GL_CMHANDLR
      ```

## Starting The Compatibility Mode Image

- Once the image has been read into memory, final step of initialization is to pass control to the compatibility mode image.
- Push special PSL onto stack
- Then push PC at which compatibility mode image is to begin executing on stack.
  - Typically found in image file, but is system dependent
- Issue REI instruction

PSL Used to Enter Compatibility Mode = 83C000xx

| Bits in PSL | Meaning and Required Values |
|---|---|
| 0-3 | Condition codes (no required values) |
| 4 | T-Bit (no required value) |
| 5-7 | Arithmetic Trap Enables (must be zero) |
| 8-15 | All set to zero |
| 16-20 | IPL must be zero |
| 21 | Must be zero |
| 22-23 | Previous mode=User (both set) |
| 24-25 | Current mode=User (both set) |
| 26-30 | Must be zero (Bit 26=IS,30=Trace Pending) |
| 31 | Compatibility Mode (must be set) |

## Responding to Compatibility Mode Exceptions

### System Response

- Push PSL, PC, and exception code on kernel stack
- Vector through SCB to routine EXE$COMPAT
  - New PSL formed that indicates now running in native mode again
- EXE$COMPAT performs the following operations:
  - Saves RØ-R6 in area in Pl space
  - Pops exception code, PC and PSL
  - Saves exception code, PC, and PSL in Pl space
  - Transfers control to exception dispatcher
- Information stored in Compatibility Mode Context Area (in Pl space) before control is passed to Compatibility Mode Exception Handler.

```
+-----------------------+
| Saved RØ              | <--  CTL$AL_CMCNTX:: <----+
| Saved R1              |            or             |
| Saved R2              |      SYS$GL_CMCNTX:: -----+
| Saved R3              |
| Saved R4              |
| Saved R5              |
| Saved R6              |
| Exception Code        |
| Exception PC          |  <-- RØ
| Exception PSL         |
+-----------------------+
```

## Dispatch to Customized Compatibility Mode Exception Handler

- Established by $DCLCMH
- Control is passed to handler in user mode
- Handler accesses information in P1 space

```
MOVL G^SYS$GL_CMCNTX,R11      ; Put address of context
                              ; area into R11 for
                              ; subsequent displacement
                              ; mode addressing
MOVW (RØ),R1Ø                 ; Pick up PC of
                              ; faulting instruction,
MOVW 4(RØ),R9                 ; and also faulting
                              ; PSL
```

  - Use SYS$ symbol not CTL$ symbol to avoid linking to system symbol table
  - If use CTL$ symbol, access information in P1 space using:

```
MOVAL G^CTL$AL_CMCNTX,R11   ; Use MOVAL, not MOVL
```

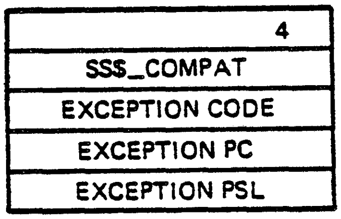
- RSX-11M AME uses symbolic offsets to reference contents of the compatibility mode context area

```
  I_RØ =  Ø
  I_R1 =  4
  I_R2 =  8
  I-R3 = 12
  I_R4 = 16
  I_R5 = 2Ø
  I_R6 = 24
I_TYPE = 28
  I_PC = 32
  I_PS = 36
```

- Exception handler must perform whatever action is necessary to service the exception.
  - May handle exception internally
  - May involve system service calls
  - May involve RMS calls

## Dispatch to Normal Condition Handler

- No handler established via $DCLCMH
- Exception dispatcher pushes exception PSL, PC, code, and symbol SS$_COMPAT on kernel stack
- Signal and mechanism arrays, as well as argument list, built on user stack
- Signal array contains:



4

SS$_COMPAT

EXCEPTION CODE

EXCEPTION PC

EXCEPTION PSL

TK-9162

- Exception handler must perform whatever action is necessary to service the exception.
  - May handle exception internally
  - May involve system service calls
  - May involve RMS calls

## Returning Control To Compatibility Mode Image

- Normal Condition Handlers
  - Must update PC in signal array to point to next instruction
    - Required since compatibility mode exception is a fault
    - If not updated, will loop indefinitely, since instruction will be re-executed
  - May update PSL in signal array to alter condition code bits
    - To return information to compatibility mode image
  - Place SS$_CONTINUE in RØ to indicate exception has bee successfully serviced
  - Exit with RET instruction
- Customized Compatibility Mode Handler
  - Must restore registers RØ-R6
  - PC in context save area must be updated as in norma handler case
  - PSL in context save area may be updated as in norma handler case
  - The new PSL and new PC are pushed onto the stack
  - An REI instruction is issued

## Summary Of Alternative Approaches To Declaring CM Exception Handler

| Operation | Normal Handler | Customized CM Handler |
|---|---|---|
| Method of Declaring Handler | - LIB$ESTABLISH | - $DCLCMH |
| | - Capable of servicing all possible exceptions | - Can only service compatibility mode exceptions |
| | | - Regular condition handler must also be established to handle other exceptions like access violation |
| Dispatching to the Handler | - Normal exception dispatching used to locate handler | - Normal exception dispatching bypassed |
| | | - No time spent looking for handler |
| | | - Dispatching much faster |
| | - Need entry mask | - No entry mask |
| | - Entered via CALLG | |
| | - Second longword of signal array must be tested for CM exception | - No special code required to determine which type of VMS exception occurred |
| | | - Only services CM exceptions |
| | - Exception code must be examined to distinguish type of CM exception | - Exception code still must be examined to distinguish type of CM exception |

## Summary Of Alternative Approaches To Declaring CM Exception Handler

| Operation | Normal Handler | Customized CM Handler |
|---|---|---|
| Arguments Passed to Handler | - Standard signal and mechanism arrays | - The PC, PSL, CM exception code, and RØ-R6 saved in P1 space |
| | - Because condition handlers are a standardized part of VMS, their properties remain constant from release to release | - The format of P1 space, and the use of RØ to locate the faulting PC, are not governed by any standard, and could change in a future release of VMS |
| | - No special link with executive | - Handler may need to be linked with system symbol table |
| | | - If so, handler must be relinked with each new version |
| | | - Using SYS$ symbol and not CTL$ symbol avoids need to link with system symbol table |
| Returning to the CM Program | - Put SS$_CONTINUE in RØ | - Must first restore RØ-R6 |
| | | - PSL and PC from P1 space must be pushed on stack |
| | - Issue RET | - Issue REI to dismiss the exception, and pass control back to CM program |
| | - Exception dispatcher dismisses the exception, passing control back to CM program | |