digital

VAX/VMS
Release Notes

Order No. AA-D015B-TE

VAX11

February 1979

This document contains information not included elsewhere in the documentation set. Typically, this information covers software and/or documentation errors that were discovered or changes that were made late in the development cycle, plus hints concerning system installation and operation. This document should be read before the system is installed or used.

VAX/VMS
Release Notes

Order No. AA-D015B-TE

SUPERSESSION/UPDATE INFORMATION:	This revised document supersedes VAX/VMS Release Notes (Order No. AA-D015A-TE) and VAX/VMS Version 1.01 Release Notes (Order No. AA-H371A-TE).
OPERATING SYSTEM AND VERSION:	VAX/VMS V1.5
SOFTWARE VERSION:	VAX/VMS V1.5

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation · maynard, massachusetts

First Printing, August 1978
Revised, February 1979

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10
VAX	VMS	SBI
DECnet	IAS	PDT
DATATRIEVE	TRAX	

CONTENTS

CONTENTS

CONTENTS

1.0 DESCRIPTION OF VERSION 1.5 OF VAX/VMS

Version 1.5 of VAX/VMS is being distributed in two ways: as a set of updates and as a totally reconfigured system.

To customers who are already users of version 1.01 of VAX/VMS, version 1.5 is being distributed as updates to the VAX/VMS operating system (V1.01) and to VAX-11 utilities (V1.01). These updates and brief descriptions of them are contained on seven floppy diskettes.

The updates contained in the updating kit are guaranteed to work, providing two conditions are met:

1. Only if you have applied the mandatory version 1.01 updates to your original version 1.0 VAX/VMS system

2. Only if you have applied no other patches to the version 1.0 system or to the version 1.01 system

NOTE

If you have not already applied the version 1.01 updates, you should do so now, before installing version 1.5 of VAX/VMS. Follow the instructions for updating the system that are contained in Chapter 6 of the VAX-11 Software Installation Guide.

If you have already applied the version 1.01 updates, you need not do so again.

All updates in the updating kit are mandatory. Rejecting an update now may make it difficult for DIGITAL to respond to problem reports later. You may also find it difficult to apply a future update that might be supplied by DIGITAL.

To new customers, version 1.5 of VAX/VMS is being distributed either on a 1600 bit-per-inch (bpi), 9-track magnetic tape or on an RK07 disk cartridge. This version of VAX/VMS is up to date and, thus, need not be updated in any way.

2.0 FIELD CHANGE ORDERS AND MICROCODE FILES NEEDED FOR VERSION 1.5

The followng hardware field change orders (FCOs) are needed for the correct and reliable operation of version 1.5 of VAX/VMS.

- The FCOs required for version 1.0 of VAX/VMS

- VAX-R-001; this FCO is required for users of version 4.0 of VAX-11 COBOL-74

In addition, the following microcode file is required for the correct and reliable operation of programs produced by the VAX-11 COBOL-74 compiler.

WCS118.PAT or later.

Refer to Section 4.3 for further explanation of the FCO VAX-R-001 and the microcode file needed for the correct and reliable operation of programs produced by the VAX-11 COBOL-74 compiler.

3.0 INSTALLING VERSION 1.5 OF VAX/VMS

To update the VAX/VMS system using floppy diskettes, follow the instructions for updating the system given in Chapter 6 of the VAX-11 Software Installation Guide. Use the diskettes in the following order:

1. VMS150A
2. VMS150B
3. VMS150C
4. VMS150D
5. VMS150E
6. VMS150F
7. VMS150G

When updating the VAX/VMS system, ignore the following informational messages produced by the Native Image File Patch Utility (PATCH).

```
%PATCH-I-NOLCL, image does not contain local symbols
%PATCH-I-NOGBL, some or all global symbols not accessible
```

To install a VAX/VMS system from a magnetic tape or from an RK07 disk cartridge, follow the instructions for bootstrapping and installing a VAX/VMS system. The VAX-11 Software Installation Guide contains these instructions.

Note that throughout Chapter 2 of the VAX-11 Software Installation Guide, the part descriptions of the two floppy diskettes should be changed as follows:

- The floppy diskette (formerly ESZCCnn 11780 S/A DSC2 FLP) that contains the stand-alone Disk Save and Compress (DSC) utility program should now be referred to as RX 9/ 11780 S/A DSC2 FLP.
- The console floppy diskette (formerly ESZABnn 11780 LOCAL CNSL PKG) should now be referred to as RX 1/ 11780 LOCAL CNSL PKG.

4.0 OPTIONAL SOFTWARE PRODUCTS RELATED TO VAX/VMS

The following sections cover optional software products that are related to this version of VAX/VMS.

4.1 DECnet-VAX

A new version (1.1) of DECnet-VAX is now being distributed to eligible customers.

If you attempt to install any previous version of DECnet-VAX, you must install it before updating VAX/VMS. You can, however, install version 1.1 of DECnet-VAX either before or after updating VAX/VMS.

4.2 VAX-11 FORTRAN IV-PLUS

A new version (1.2) of the VAX-11 FORTRAN IV-PLUS optional software product is now being distributed to eligible customers.

For present VAX-11 FORTRAN IV-PLUS customers, the changes to this product are contained on a new floppy diskette labeled VAXFORTV1 BIN RX01. This floppy diskette replaces the present floppy diskette that is labeled VAXFORTV1 BIN RX01.

To update VAX-11 FORTRAN IV-PLUS, you must reinstall it according to the instructions for installing VAX-11 FORTRAN IV-PLUS that are contained in Chapter 6 of the VAX-11 Software Installation Guide. Use the floppy diskettes in the following order:

1. The new VAXFORTV1 BIN RX01
2. VAXFORTBB BIN RX01 from the original kit

No changes in documentation result from this updating of VAX-11 FORTRAN IV-PLUS.

New VAX-11 FORTRAN IV-PLUS customers, on the other hand, receive a VAX-11 FORTRAN IV-PLUS kit consisting of two floppy diskettes that already contain the version 1.2 changes. For this reason, new customers need install VAX-11 FORTRAN IV-PLUS only once, according to the instructions for installing VAX-11 FORTRAN IV-PLUS that are contained in Chapter 6 of the VAX-11 Software Installation Guide.

4.3 VAX-11 COBOL-74

VAX-11 COBOL-74, version 4.0, a new optional software product, is now available to eligible customers. This product replaces the previously available PDP-11 COBOL-74/VAX, version 3.

4.3.1 **Required FCO and Microcode File** - The following hardware field change order (FCO) and microcode file are needed for the correct and reliable operation of programs produced by the VAX-11 COBOL-74, version 4.0, compiler.

- VAX-R-001; this FCO includes KA780-processor ECO M8235-TW005 (also referred to as the CVTTP ECO)
- WCS118.PAT or later

The compiler runs in compatibility mode and hence is not affected by the FCO. However, executable COBOL-74 programs may give unexpected or undesired results if the FCO is not present. In particular, overpunched data items with a length of 1 will always appear to have a value of 0. The first sample test program ([SYSTEST]IO001.COB), which is contained in the VAX-11 COBOL-74 distribution kit, makes use of this fact to test for the absence of the FCO.

4.3.2 **Restriction in Use of CALL Statement** - The following restriction is placed on the use of the CALL statement.

An argument passed BY DESCRIPTOR cannot be USAGE COMPUTATIONAL, JUSTIFIED RIGHT, or have a SIGN...SEPARATE clause. Change General Rule 7c on page 5-33 of the VAX-11 COBOL-74 Language Reference Manual to read as follows:

> "c. DESCRIPTOR - The address of (pointer to) the descriptor of the data item is passed to the called program. The usage of the data item cannot be COMPUTATIONAL; nor can its description specify the JUSTIFIED or SIGN...SEPARATE clause."

This restriction is documented in Appendix D of the VAX-11 COBOL-74 User's Guide. In this appendix, the description of diagnostic 1114 documents this restriction.

4.4 VAX-11 BLISS-32

VAX-11 BLISS-32, version 1.0, a new optional software product, is now available to eligible customers.

Following is a list of known problems with BLISS-32, version 1.0:

- The following construct may cause access violations during compilation:

```
BIND A = PLIT( REP 0 OF (0) );
```

  To achieve the desired data allocation, use:

```
BIND A = UPLIT(0) + %UPVAL;
```

  This will be fixed in a future release.

- Use of %EXPAND before a name that has not been declared may cause access violations during compilation. The compiler should ignore the %EXPAND. A solution is under investigation.

- Page 1-7 of the VAX-11 BLISS-32 User's Guide states that a switches-declaration counterpart of the /DEBUG command qualifier and the DEBUG module-head switch exists. This is not true; no such declaration exists. The BLISS-32 Language Guide correctly lists the valid set of SWITCHES switch items.

4.5 FORTRAN IV/VAX To RSX Cross Compiler

The following section entirely replaces Section 6.4.2 in the VAX-11 Software Installation Guide.

6.4.2 Building the Compiler

After the code generation phase is completed, you receive the message:

```
Please put the second floppy disk (VMS11FORA) in the drive.

Are you ready to continue?
```

Place diskette VMS11FORA in the drive. When you are ready to proceed, type Y. The following message then appears:

> The compiler object library must be copied and the compiler task-built. This will take approximately 20 minutes.

Completion of these procedures is indicated by the message:

> Please put the third floppy disk (VMS11FORB) in the drive.
>
> Are you ready to continue?

Place diskette VMS11FORB in the drive. When you are ready to continue, type Y. You will now be queried about OTS options.

4.6 Other Products

At this time, there are no changes to the following optional software products.

- PDP-11 BASIC-PLUS-2/VAX
- PDP-11 DATATRIEVE/VAX

NOTE

> Refer to Appendix A for instructions on using the PDP-11 BASIC-PLUS-2/VAX compiler on the VAX/VMS operating system.

5.0 COMMAND LANGUAGE AND FILE SYSTEM

This section notes restrictions, documentation omissions, and documentation errors on commands and file-system-related issues.

5.1 COPY Command

The following notes apply to the COPY command.

5.1.1 **Copying Subdirectories** - Although subdirectory files can be copied from one directory to another, either explicitly or through the use of wildcards, the copied directories are unusable. For example:

```
$ COPY *.* [MALCOLM]
```

If any subdirectory files (that is, files with the file type DIR created with the CREATE/DIRECTORY command) exist in the current default directory, they are copied to the directory [MALCOLM], but cannot be used. Files listed in the subdirectory are never copied by this command.

5.1.2 **Wildcards in Input File Lists** - The COPY command handles wildcards in input file lists inconsistently, as shown in the following examples.

```
$ COPY *.LIS,*.LST OUTPUT
```

When this command executes, if no files currently exist with file types of LIS, the COPY command continues execution and copies files with file types of LST. However, the following case does not work correctly:

```
$ COPY A.FIL,*.LIS,*.LST   OUTPUT
```

In this case, if no files with file types of LIS exist, the COPY command terminates without copying files with file types of LST.

This problem will be fixed in a future release of the system.

5.2 DIFFERENCES Command

The items listed below will be corrected in future releases of the system.

5.2.1 **Matching Records Reported as Unmatched** - The default match size for the DIFFERENCES command, which can be changed with the MATCH qualifier, is 3. This means that DIFFERENCES requires match-size records to be identical after it reports an unmatched record.

When "match-size minus 1" records at the end of a file follow one or more unmatched records, DIFFERENCES always lists these records as unmatched, even if they do match. For example, if the last three lines in the files A.B and D.B are as shown below, DIFFERENCES always lists all three records as unmatched:

File: A.B	File: D.B
AAAAAA	DDDDDD
BBBBBB	BBBBBB
CCCCCC	CCCCCC

Note that this condition also occurs when DIFFERENCES compares files that contain fewer than "match-size" records.

This problem will be corrected in a future release of the system.

5.2.2 **Logical Names** - The DIFFERENCES command performs logical name translation correctly only when both of the following conditions are met:

- The logical name is equated to only a device and/or a directory and does not contain a file name or a file name and file type
- The logical name is terminated with a colon

For example, the following command is valid only if the logical name OUTFILE was assigned to only a device and/or directory:

```
$ DIFFERENCES/OUTPUT=OUTFILE:  A.B
```

If the equivalence name for the logical name OUTFILE contains a file name or a file name and type, the DIFFERENCES command issues an error message. If the logical name is not terminated with a colon, the DIFFERENCES command assumes that OUTFILE is a file name and writes the output file OUTFILE.DIF.

This problem will be corrected in a future release of the system.

5.3 DIRECTORY Command

The following notes apply to the DIRECTORY command.

5.3.1 **Use of Colons in Logical Names** - When a file specification for the DIRECTORY command contains no punctuation, the DIRECTORY command assumes that the file specification contains a single file name, and searches for one specific file, rather than listing all files (listing all files is the default when only a device and/or directory is specified).

When a logical name has been equated to a device and/or directory name string, the DIRECTORY command searches the specified device/directory for a file with a null file name and file type if the logical name is not terminated with a colon. For example:

```
$ ASSIGN [TESTFILES] ABC
$ DIRECTORY ABC:
```

The colon terminating the logical name ABC in this example is required.

5.3.2 **Anticipated Change in DIRECTORY Command** - In a future major release of VAX/VMS, the format of the output produced by use of the DIRECTORY command may differ from that now produced by use of this command.

Users who now use or who plan to use this command in command procedures, for example, should keep this fact in mind.

DIGITAL recommends that you not process this output by a command procedure or by a program unless you are prepared to change the command procedure or the program after a future system release in which the format of this output changes.

5.4 EXAMINE Command

The EXAMINE command does not enforce no-read access to an image file for which execute access is allowed. For example, if users in the group and world categories execute an image file that is protected by the code RWED,RWED,E,E, the image can be interrupted and examined during execution.

This problem will be corrected in a future release of the system.

5.5 MACRO Command

The following notes apply to the MACRO command.

5.5.1 Removal of a Restriction

- Version 1.5 of VAX/VMS removes a restriction that formerly applied to the VAX-11 MACRO command line.

Previously, in the MACRO command line, macro libraries had to be specified before any input source files were specified. This restriction was made necessary by the way in which the VAX-11 MACRO assembler handled its free memory allocation (see Section 5.5.2).

This restriction no longer applies. In fact, because of the way in which DCL handles implicit outputs, it is better and safer to specify the input source files first.

The following example illustrates why you should specify input files first. Assume that EXECML$ is a logical name with the translation SYS$LIBRARY:LIB.MLB and that SRC$ is a logical name with the translation [.SRC].

Although the following two commands appear equivalent, their effects are quite different.

- $MACRO EXECML$/LIB+SRC$:MYFILE
- $MACRO SRC$:MYFILE+EXECML$/LIB

The first command shown above creates a new version of EXECML$, the system macro library. The second command produces the desired result: the file MYFILE.OBJ in the current directory.

The reason these commands give different results is explained in Section 6.3.3 of the VAX/VMS Command Language User's Guide.

5.5.2 Incompatibility Between Previous and Present Versions of VAX-11 MACRO

- The following command line illustrates an incompatibility between the present version of VAX-11 MACRO and the previous version. Assume that SRC1 refers to the macro $XYZDEF and that $XYZDEF is defined in LIB1.MLB.

```
$MACRO SRC1+LIB1/LIB+SRC2
```

Previously, this command would have caused an error. The new version of the VAX-11 MACRO assembler, however, gathers up all the libraries before processing the command. Hence, no error is generated.

5.6 REPLY Command

This release note supplements Section 3.10 of the VAX/VMS Operator's Guide.

You can specify the keyword NET with both the /ENABLE and /DISABLE command qualifiers of the REPLY command. This keyword is described below:

/ENABLE=NET	Designates a terminal to receive messages pertaining to networks

/DISABLE=NET Inhibits a terminal from receiving messages pertaining to networks. If, however, a terminal has been enabled as central, you must use the command REPLY/DISABLE to prevent that terminal from receiving network messages.

5.7 SET PROTECTION Command

The SET PROTECTION command does not apply temporary defaults to file specifications in a list. For example:

```
$ SET PROTECTION=SYSTEM:D [TEMP]A.EXE,B
```

This command changes the protection code for the files:

```
[TEMP]A.EXE
[]B.
```

The directory name TEMP and the file type EXE are not applied to the specification of the second input file.

This restriction will be lifted in a future release of the system.

5.8 SET TERMINAL/PERMANENT Command

This release note corrects Section 3.13.6 and Table 3-7 of the VAX/VMS Operator's Guide.

The following new command qualifiers are also available for the SET TERMINAL/PERMANENT command:

/[NO]PARITY=[option]
Defines the parity for the terminal. You can specify one of the following options:

EVEN
ODD

/NOPARITY is the default. If you specify /PARITY and you do not specify an option, the command assumes /PARITY=EVEN. Any value other than EVEN or ODD will produce unpredictable results.

/[NO]READSYNC
Controls whether the system solicits read data from a terminal using CTRL/S and terminates the read using CTRL/Q.

/NOREADSYNC is the default. The system does not use CTRL/S and CTRL/Q to control reads to the terminal. The /READSYNC qualifier is useful for certain classes of terminals that demand synchronization or on special-purpose terminal lines where data synchronization is appropriate.

Table 3-7 should be modified to reflect the following changes:

1. Under the first column labeled "Terminal Characteristics," HEIGHT should be changed to PAGE, and the values in the third, fourth, and fifth columns that correspond to this characteristic should be changed to 8, 20, and 24, respectively.

2. Under the first column labeled "Terminal Characteristics," EIGHTBIT should be changed to EIGHT_BIT, HOLDSCREEN should be changed to HOLD_SCREEN, and TYPEAHEAD should be changed to TYPE_AHEAD.

3. The values in the third, fourth, and fifth columns that correspond to the terminal characteristic CRFILL should be changed to 0, 0, and 0.

4. The values in the third, fourth, and fifth columns that correspond to the terminal characteristic LFFILL should be changed to 0, 3, and 0, respectively.

5. The following terminal characteristics and values should be added to Table 3-7:

PARITY	*	no	no	no
READSYNC	*	no	no	no

5.9 STOP/ENTRY Command

This release note corrects Section 3.17 of the VAX/VMS Operator's Guide.

The purpose of the STOP/ENTRY command is to terminate the execution of a batch job while it is running. If you want to delete an entry from a printer or batch job queue while the entry is waiting to be executed, use the DELETE/ENTRY command. This command is described in the VAX/VMS Command Language User's Guide.

5.10 Compatibility Mode Commands

Commands that execute in compatibility mode do not display directory name strings exceeding 17 characters if the file specification in the command does not contain an explicit directory name (for example, if the directory is defaulted or specified as a logical name). Directory name strings greater than 17 characters are truncated, when formatted for output, by the DIRECTORY, DUMP, EDIT, and UNLOCK commands.

None of the compatibility mode commands accepts logical names that have underscores in them. This applies to the BASIC, COBOL/C74, COBOL/RSX11, CREATE, DIFFERENCES, DIRECTORY, DUMP, EDIT, FORTRAN, LIBRARY, LINK/RSX11, RENAME, SET PROTECTION, SORT/RSX11, and UNLOCK commands.

Both of these problems will be corrected in a future release of the system.

5.11 Truncating Keywords

The following release note supplements Section 6.1.3 of the VAX/VMS Command Language User's Guide. The documentation does not make it clear that when truncating command qualifier keywords to the minimum four characters for guaranteed uniqueness you must not count the prefix NO as two of the four characters. Keywords in the negative form require six characters for guaranteed uniqueness. A future update to the documentation will include this clarification.

Users who have been abbreviating the /NOUNLOAD qualifier of the DISMOUNT command to /NOUN were actually violating the four-character rule as explained above, and will discover they are now required to enter at least /NOUNL, due to the introduction of the /UNIT qualifier with this release. (Note that using the six-character negative form, /NOUNLO in this case, is recommended in command procedures to ensure compatibility with all future releases.)

6.0 PASSING PARAMETERS TO IMAGES EXECUTED WITH THE COMMAND INTERPRETER

The command interpreter allows you to define foreign commands. A foreign command is a command that is not known to the command interpreter but that can be executed by entering a command string.

The command interpreter provides the following mechanisms so you can execute your programs as foreign commands and pass variable data to them at execution time:

- To specify parameters when an image is run, you must use an assignment statement to define a command name to be used instead of a RUN command to execute an image.
- To obtain the parameters, the image must request the parameter string from the command interpreter and must perform all string parsing and analysis itself.

Each of these mechanisms is described in detail below.

6.1 Defining a Foreign Command

Use the following syntax of an assignment statement to define a foreign command:

```
$ symbol-name :=[=] $image-file-spec
```

symbol-name
The name by which you want to invoke the image.

$image-file-spec
The file specification of the image to be executed. The image-file-spec must contain a device name and a file name; optionally, you can specify a device name, a directory name, a file type, or a file version. In general, the image-file-specification should contain a directory name. The default device and directory name is SYS$SYSTEM, the default file type is .EXE, and the default file version is the latest version.

The dollar sign ($) preceding the image-file-spec is required.

After you have defined a foreign command as shown above, the request to execute the image is implicit in the symbol definition. When this symbol-name is specified as the first token, or item, in a command, the command interpreter executes the specified image. For example:

```
$ PROCESS := $DB1:[MALCOLM.PROG]CREPROCES
$ PROCESS
```

In this example, the symbol-name PROCESS is defined as a foreign command. When PROCESS is specified as the first token in a command, you can specify any data following it. For example:

```
$ PROCESS ORION
```

This command string passes the string ORION to the executing image. The image must obtain the parameter string. The image must also perform any parsing or evaluation of the command string; the command interpreter does not parse the line.

Note that during command input the command interpreter performs all symbol substitution requested by apostrophes (') in the command string. Thus, if you use symbols preceded by apostrophes to specify parameters in a command string, substitution of these symbols occurs before the resulting command string is passed to the program.

6.2 Obtaining a Parameter String from the Command Interpreter

To obtain a parameter string passed with a foreign command, the image must call the command interpreter with the address of a 28-byte request block. A request block to the command interpreter is a structure that defines the type of request and provides storage for return information.

To request a command line, the request block must contain the constant 1 in its first byte (this constant has a symbolic name of CLI$K_GETCMD) to indicate the type of request. On return from this particular type of request, the third longword in the block contains the address of a character string descriptor that describes the parameter string present in the command (this offset within the structure is defined by the constant CLI$Q_RQDESC).

The following examples show a VAX-11 MACRO program and then a VAX-11 FORTRAN IV-PLUS program requesting the parameter string from the command interpreter and displaying the line on SYS$OUTPUT.

Sample MACRO Program

```
        .PSECT  RWDATA  WRT,RD,BYTE

;
; Build request descriptor for call back to get command line
; using the macro $CLIREQDESC (in STARLET.MLB).
; The first byte in the descriptor indicates the type of request.
; CLI$K_GETCMD is a constant, equated to 1, that requests the
; command line.
;

GETCMD:  $CLIREQDESC     -
                RQTYPE=CLI$K_GETCMD


        .PSECT  PURE    RD,NOWRT,BYTE

        .SBTTL  TEST
;++
; Functional description:
;
;       Result parse test program
;
; Outputs:
```

```
;
;       The command line is printed on output
;--

        .ENTRY  GETCMDLIN,0             ;
        PUSHAB  W^GETCMD                ; Address of the request
                                        ; descriptor
        CALLS   #1,@#SYS$CLI            ; Request CLI for command line
        PUSHAQ  W^GETCMD+CLI$Q_RQDESC   ; Command buffer descriptor
        CALLS   #1,LIB$PUT_OUTPUT       ; Call LIB$PUT_OUTPUT to
                                        ; display it
        RET
        .END    GETCMDLIN
```

Sample FORTRAN Program

```
        PROGRAM GETCMD
!++
!
! Declare the routines SYS$CLI and LIB$PUT_OUTPUT as external
!
!--
        EXTERNAL SYS$CLI,LIB$PUT_OUTPUT


!++
!
! Define the request block as an array of 7 longwords
! and put a 1 in the first longword for the request type
!
!--
        INTEGER*4 GETLINE(7)
        GETLINE(1)=1

!++
!
! Call the command interpreter;  on return, call LIB$PUT_OUTPUT with
! the third longword in the array as an argument.
!
!--

        CALL SYS$CLI(%REF(GETLINE))
        CALL LIB$PUT_OUTPUT (GETLINE(3))
        END
```

7.0 SYSTEM SERVICES

This section provides changes and clarifications for system services.

7.1 Time Values for $SETIMR and $SCHDWK

The Set Timer ($SETIMR) system service does not return the status code SS$_IVTIME. If an absolute time value specified for $SETIMR has already passed, the timer expires at the next clock cycle, that is, within 10 milliseconds.

The Schedule Wakeup ($SCHDWK) system service returns the status value SS$_IVTIME only under two conditions:

- If a delta repeat time argument specifies a positive (that is, an absolute time) value
- If a specified absolute time plus a specified delta repeat time results in an absolute time that is less than the current time

If an absolute time that has already passed is specified for the $SCHDWK system service and no repeat time is specified, the timer expires at the next clock cycle, that is, within 10 milliseconds.

A repeat time value for $SCHDWK cannot be less than 10 milliseconds. If smaller times are specified, they are increased automatically to 10 milliseconds.

These corrections for the $SETIMR and $SCHDWK system services will be incorporated in a future update to the documentation.

7.2 Disposition of Messages Output by $PUTMSG

The Put Message ($PUTMSG) system service writes one or more formatted messages to a process's current output and/or error devices. A message is written after an action routine specified in the call to $PUTMSG, if any, returns control with a successful status value. If no action routine is specified, the message is always written.

The actual disposition of each message depends on the severity level of the status value associated with the message. The following table indicates:

- Whether the message is written to the current output device (SYS$OUTPUT)
- Whether the message is written to the current error device (SYS$ERROR)
- Whether the message cancels the effect of CTRL/O, that is, if the message is displayed when the CTRL/O function has suppressed all output to the terminal

Severity Level	Written to SYS$OUTPUT	Written to SYS$ERROR	Cancels CTRL/O
Warning	yes	yes	yes
Success	yes	no	no
Error	yes	yes	yes
Informational	yes	yes	no
Severe error	yes	yes	yes

This information will be included in a future update to the VAX/VMS System Services Reference Manual.

7.3 Correction to $PUTMSG Documentation

In the VAX/VMS System Services Reference Manual, the note at the bottom of page 4-118 should be deleted, because the $PUTMSG service does not disable AST delivery while it is executing.

7.4 Corrected Explanation of JPI$_LOGINTIM for $GETJPI

In the VAX/VMS System Services Reference Manual, in Table 4-4 on page 4-101, the explanation of the JPI$_LOGINTIM item identifier for the $GETJPI system service is incorrect. The "Information Returned" column for this identifier should read: "The absolute time of process creation; returned as a 64-bit value expressing the login (process creation) time."

This correction will be included in a future update of the manual.

7.5 Character String Descriptors for System Services

The availability of a new assembler directive (.ASCID) eliminates the need for VAX-11 MACRO programmers to write a special macro to create a character string descriptor, which several system services require as an argument. This special macro is illustrated and given the name "DESCRIPTOR" on page 2-10 of the VAX/VMS System Services Reference Manual, and this macro is used in examples throughout Chapter 3.

To use the example on page 2-10, you can achieve the same result by eliminating the macro definition and replacing the line

```
CYGNUSDESC:  DESCRIPTOR <CYGNUS>
```

with the line

```
CYGNUSDESC:  .ASCID /CYGNUS/
```

8.0 RECORD MANAGEMENT SERVICES

This section provides changes and clarifications for VAX-11 Record Management Services (VAX-11 RMS).

8.1 Default Date and Time Values for XAB Macros

If you specify any parameters defining date and time fields as 0, either explicitly or by default, the fields are set to November 17, 1858. This applies to the following fields specified for the $XABDAT (Date and Time XAB) and $XABRDT (Revision Date and Time XAB):

- Creation date and time
- Expiration date and time
- Revision date and time

Note that when either the $XABDAT or $XABRDT macro is supplied for a file and any of the fields listed above are not specified in the macro instruction, the macro parameter defaults to a value of 0 and the date and time fields are given the erroneous 1858 date and time.

This problem will be corrected in a future release of the system.

8.2 Device Characteristics Returned on Parse

A successful call to the SYS$PARSE service returns with the device characteristics for the target device in the DEV field of the FAB. If the target device for the SYS$PARSE routine is a process permanent file, however, no device characteristics are returned and the DEV field of the FAB remains unchanged.

This problem will be corrected in a future release of the system.

8.3 RMSSHARE Utility

Before the file sharing capability of RMS can be used, the system manager must run the utility program RMSSHARE. This utility should be run each time the system is booted. For this reason, it is a good idea to include a command for running RMSSHARE in a system start-up command procedure.

All file sharing structures maintained by VAX-11 RMS reside in the system S0 space and are allocated from the system paged pool. This utility program allocates the initial page from the system in which are maintained the current and maximum page counts of the file sharing data base and various list headers.

RMSSHARE informs the system manager as to whether file sharing has or has not been enabled, and if it has been, RMSSHARE displays current and maximum page counts. The system manager then can increase or decrease the maximum page count to which the data base can grow. However, because these pages are not returned to the system pool until the system is booted again, the maximum cannot be set lower than the current count.

8.3.1 **Running RMSSHARE** - Any user who has the CMEXEC privilege can invoke and run the RMSSHARE utility program by use of the following command.

```
RUN SYS$SYSTEM:RMSSHARE
```

Usually, however, it is a duty of the system manager to run this program.

When the program prompts for the maximum page count, the number that is estimated by use of the guidelines given in Section 8.3.2 can be entered.

To terminate the execution of this utility, the command EXIT (in uppercase letters) should be entered in response to the program prompt.

8.3.2 **Guidelines for Estimating Maximum Page Count** - The following guidelines can help the system manager to determine the maximum number of pages that may be needed for the file sharing structures that are maintained by VAX-11 RMS. The number of pages needed are estimated as follows:

- Two pages per system
- For each relative file being shared, one page for the first three sharers plus one page for every four additional sharers
- For each indexed file being shared, one page for the first two sharers plus one page for every two additional sharers

The preceding guidelines assume a maximum of one locked record per sharer and a default multibuffer count of 1 per relative file and 2 per indexed file. If these assumptions are not true for your system, you can apply the following additional guidelines to both relative and indexed files on a per-file basis, as follows:

- 16 bytes per additional locked record
- 36 bytes per additional buffer

Note that space is allocated to a file in whole pages.

Note too that, when file sharing is being used, the size of the paged dynamic pool, PAGEDYN, should be increased to accommodate the maximum number of pages needed for file sharing, in addition to the other requirements.

8.4 ISAM Files and System Working Set Size

When VAX-11 RMS ISAM files are operated upon, an increase in system page faulting can occur. Increasing the system working set size, SYSMWCNT, by 50 pages should be enough to maintain the current system page fault rate.

8.5 VAX-11 RMS Bugchecks

Setting BUGCHECKFATAL to 0 will cause RMS bugchecks to delete the process. Setting BUGCHECKFATAL to 1 will cause RMS bugchecks to produce a system crash. In the unlikely event of an RMS bugcheck, deleting the process will minimize the impact of the error but will result in the gathering of only minimum debugging information.

8.6 Access Mode in Which VAX-11 RMS Runs

VAX-11 RMS runs in executive mode and assumes that at the time of the user call executive mode ASTs are enabled. If the user process calls RMS in executive mode with ASTs disabled, the process will hang if RMS is required to perform a SYS$QIO to complete the operation. The reason for this is that RMS is AST driven on I/O completion. Note also that RMS must not be called from kernel mode.

8.7 Restriction on Opening Relative and Indexed Files

VAX-11 RMS relative and indexed files can not be opened as process-permanent files.

8.8 Restriction on Copying Relative and Indexed Files

By use of the COPY command, VAX-11 RMS relative and indexed files can be copied only to disk.

To copy relative and indexed files to magnetic tape or to record-oriented devices, the RMS-11 CONVERT utility (invoked by use of the command MCR CNV) should be used.

9.0 KNOWN IMAGES AND GLOBAL SECTIONS ON PRIVATE VOLUMES

The INSTALL utility program cannot display the names of or list information about known executable images or permanent global sections that are installed from private volumes. (A private volume is a volume that is mounted with neither the SHARE nor the SYSTEM qualifier.)

Therefore, it is recommended that all images to be installed as either known executable images or as permanent global sections be installed from shareable volumes.

This information will be included in a future update to the documentation.

10.0 INDEX FILE HEADER ERRORS ON DISK VOLUMES

When a disk volume (except for system disk) is mounted and the primary index file header is bad, a warning message appears and MOUNT attempts to use the back-up index file header. When this occurs, or when an I/O error occurs on other portions of the file structure (for example, the bitmap), the volume is software write-locked to prevent further corruption.

Messages associated with these error conditions do not always indicate that the volume is software write-locked. The messages are as follows:

- IDXHDRBAD, index file header is bad; backup used
- MAPHDRBAD, storage map header is bad; volume locked
- IDXMAPERR, I/O error on index file bitmap; volume locked
- BITMAPERR, I/O error on storage bitmap; volume locked

Explanations of these messages in the VAX/VMS System Messages and Recovery Procedures Manual will be clarified in a future update to that document.

11.0 DISK VOLUME SETS

The following information on defining and using disk volume sets will be incorporated into future updates to VAX/VMS documentation.

The update to the VAX/VMS Command Language User's Guide for this release incorporates changes to the MOUNT and DISMOUNT commands reflecting this new support.

11.1 Introduction

Using VAX/VMS, you can bind two or more disk volumes into a volume set. A volume set has a single directory structure; the MFD (master file directory) for the entire volume set exists on the first volume in the set, called the root volume. Each volume in the set is identified by a relative volume number in the set, where the root volume is always relative volume 1.

To create a volume set, you use the MOUNT command with the BIND qualifier. The BIND qualifier identifies a set by giving it a volume set name, which applies to all volumes in the set, and it identifies the root volume and creates the directory structure for the volume.

Once a volume set has been created:

- All users who have directories and files on the set access their files either by referring to the physical device name of the device on which the root volume is mounted or by referring to a logical name established for the volume set.
- When users create files on a volume set, the file system allocates space for the files anywhere on the set, wherever there is the most room.
- When existing files on any volume are extended, extension occurs on the same volume, unless the volume is physically full.
- New volumes can be added to a volume set whenever additional space is required.

For example, all disk volumes that are mounted on a daily basis can be bound into a volume set. Since this set contains all user file directories, users do not need to specify device names in file specifications to access files that would be on other volumes. In fact, the physical location of a file is transparent to all users of the system.

The next sections describe the procedures for creating and mounting volume sets, and contain additional notes on volume sets.

11.2 Creating a Volume Set

You can create a volume set from new, freshly initialized volumes or you can create a volume set by extending an existing volume that already contains a directory structure and files.

No special privileges are required to create or use volume sets; however, you must have write access to index files on all volumes that you are attempting to bind into a volume set. In general, this means

that you must have a system UIC, have the user privilege LOG_IO, or be the owner of the volumes.

11.2.1 **Creating a Volume Set from New Volumes** - This procedure assumes that none of the volumes to be bound into a volume set contains files or data.

1. Allocate the necessary devices and physically mount the volumes.

2. Initialize each volume in the set:

```
$ INITIALIZE DB1:  PAYVOL1
$ INITIALIZE DB2:  PAYVOL2
$ INITIALIZE DB3:  PAYVOL3
```

   When you initialize volumes for a volume set, you can also use other qualifiers on the INITIALIZE command to define the volume ownership and protection. Although not required, it is recommended that all volumes in a set have the same protection and the same owner.

3.
```
$ MOUNT/BIND=MASTER_SET -
$_DB1:, DB2:, DB3:  PAYVOL1, PAYVOL2, PAYVOL3
```

The MOUNT/BIND command creates the volume set. This command defines the volume set name, MASTER_SET, and defines the relative volume numbers of the volumes PAYVOL1, PAYVOL2, and PAYVOL3.

A volume set name can have from 1 to 12 alphanumeric characters; the volume set name must be different from all volume labels within the set and all labels in the set must be unique.

The order of the device names corresponds to the volume labels specified: PAYVOL1 must be physically mounted on DB1, PAYVOL2 on DB2, and PAYVOL3 on DB3.

PAYVOL1, because it is listed first in the list of labels, becomes the root volume of the set. Its master file directory (MFD) contains the directory structure for the entire set.

Note that the MOUNT/BIND command creates the volume set and mounts the volumes. When this command completes successfully, all volumes in the set are ready for use: user file directories can now be created.

11.2.2 **Creating a Volume Set from an Existing Volume** - The following example assumes that the volume USERFILES already contains a directory structure and files and that the volume is currently mounted on the device DB1.

```
$ MOUNT/SYSTEM/BIND=USERS -
$_DB1:, DB2:    USERFILES, USERFILES2
```

The initial volume USERFILES must be specified first: it becomes the root volume of the set. When you create a volume set from an existing volume, you must specify that volume first because the file system must build on the existing directory structure.

Note that if you attempt to create a volume set from two or more volumes that already contain files and data, the file system does not

issue an error message when you issue the MOUNT/BIND command. However, the volumes are unusable as a volume set because the directory structures are not properly bound.

11.3 Mounting a Volume Set

When you mount an existing volume set, you must specify the names of the devices on which the volumes are mounted and the volume labels in a corresponding order. The MOUNT command verifies the label on each device/volume pair specified in the list. For example:

```
$ MOUNT/SHARE DB1:, DB2:, DB3:, -
$_PAYVOL1, PAYVOL2, PAYVOL3
```

You can also issue separate MOUNT commands for each device and volume in the set. For example:

```
$ MOUNT/SHARE DB1:  PAYVOL1
$ MOUNT/SHARE DB2:  PAYVOL2
$ MOUNT/SHARE DB3:  PAYVOL3
```

The effect of these commands is the same as that of the previous MOUNT command example: the three volumes in the set are mounted.

Note that the file system does not require that all volumes in a volume set be mounted. If a user attempts to access a file in a volume set and the volume is not currently mounted or the root volume is not mounted, the fatal status DEVNOTMOUNT is returned.

11.3.1 Volume Status

Volume Status - When you mount a volume set by mounting volumes individually, you must ensure that the MOUNT commands define the volume in the same way. For example, if one or more volumes are mounted /SHARE initially, all subsequent volumes must also be mounted /SHARE.

The file system maintains the names of volume sets in two lists corresponding to the possible statuses. In each list, volume set names must be unique. There is one list consisting of the names of all volume sets that are mounted privately and another list consisting of the names of all volume sets that are mounted as shareable. A volume's status is defined on the MOUNT command. The possible statuses and the corresponding qualifiers that define them are:

Status	Qualifier
Private	/NOSHARE
Shared	/SHARE
Group shared	/GROUP
System	/SYSTEM

Thus, if a volume set is mounted /SYSTEM and subsequently a request is made to bind another volume to the set and the SYSTEM qualifier is omitted, the volume set name is placed in the list corresponding to volume sets mounted privately. Assuming that no volume set of the specified name is mounted /NOSHARE, the new volume will become relative volume 1 of a new volume set, because the volume set was not found. To correctly bind it into the existing volume set, you must dismount the volume, reinitialize it, and then remount it.

11.3.2 **Logical Names** - When you mount a volume set, you can specify a logical name for the set or you can allow the MOUNT command to make default logical name assignments. A logical name for a volume set can be used to refer to all volumes in the set. For example:

```
$ MOUNT/SHARE DB1:, DB2:, DB3:  -
$_PAYVOL1, PAYVOL2, PAYVOL3   PAY
```

This MOUNT command mounts three volumes in a volume set and assigns the logical name PAY to the set. Users who are sharing this volume set can use the logical name PAY in place of the device name in file specifications to refer to the set, as follows:

```
$ PRINT PAY:[WEEKLY.JAN0878]EMPLOY.LIS
```

If you do not specify a logical name, the MOUNT command assigns the default logical name DISK$volume-set-name to the root volume, that is, to the device on which the root volume is mounted. If the root volume is not mounted, no logical name assignment is made. Each volume in the set is also assigned a default logical name of the format DISK$volume-label. However, since there is normally no need to refer to individual volumes in a volume set, except for maintenance purposes, these names are rarely used.

The MOUNT command places the logical name for the volume set and for individual volumes in different logical name tables based on the status of the set:

Status	Logical Name Table
Group	Group
Private	Process
Shared	Process
System	System

The user privileges GRPNAM and SYSNAM are required to place names in the group and system logical name tables, respectively. Hence, these privileges are required to mount a volume set in either group or system status.

11.4 Making the System Disk Part of a Volume Set

Although it is possible to make the system disk part of a volume set, it is strongly recommended that you not do so. Making the system disk part of a volume set can result in the following severe maintenance problems:

- Certain files critical to the system, such as EXEC, RMS, and SYSINIT, must physically reside on the sytem disk to be available for the boot process. When such a file is patched, the new copy could well be located on a disk other than the system disk, if the system disk is part of a volume set.

- At some point, the system volume may need to be replaced, (for example, to reallocate the paging or swap file). A single volume belonging to a volume set cannot readily be replaced.

11.5 Adding Volumes to a Volume Set

You can add volumes to an existing volume set at any time. The maximum number of volumes in a set is 255.

The following procedure assumes that the volume set named MASTER_PAY is online and mounted and has volumes named PAYVOL1, PAYVOL2, and PAYVOL3:

1. $ INITIALIZE DB4: PAYVOL4

2. $ MOUNT/BIND=MASTER_PAY DB4: PAYVOL4

This MOUNT command binds the volume PAYVOL4 with the existing volume set and makes the volume ready and available for use. Note that if the volume set MASTER_PAY was mounted in a system, group, or shared status, the MOUNT/BIND command that adds a volume to the set must also specify the appropriate qualifier.

When you add a volume to an existing set, the only volume in the set that must be mounted is the root volume, relative volume 1. None of the other volumes need be mounted.

You can also add a volume to a set at the same time that you are mounting the set. The following procedure assumes an existing volume set named MASTER_SET with volumes named PAYVOL1, PAYVOL2, and PAYVOL3:

1. $ INITIALIZE DB4: PAYVOL4

2. $ MOUNT/BIND=MASTER_SET DB1:, DB2:, DB3:, DB4: -
   $_PAYVOL1, PAYVOL2, PAYVOL3, PAYVOL4/SYSTEM

Note, in the above example of the MOUNT command, that the first device/volume pair listed in the command is the root volume of the set. When you add a volume to a set while mounting the set, you must list the root volume first.

11.6 Dismounting Volume Sets

To dismount an entire volume set, use the DISMOUNT command specifying the name of any device on which a volume of the set is mounted. For example:

$ DISMOUNT DB1:

By default, the DISMOUNT command dismounts all volumes in the set that are currently mounted.

You can use the /UNIT qualifier on the DISMOUNT command to request that only the volume on the specified device be dismounted, if necessary.

11.7 Messages

The messages listed below have been added to the VAX/VMS operating system. The documentation for these messages will be incorporated into the VAX/VMS System Messages and Recovery Procedures Manual in a future update to that document.

DEVCOUNT, number of devices must match number of volumes

Explanation: The number of devices specified in a MOUNT command to mount a disk volume set did not match the number of volume labels specified in the command.

User Action: Check the command line. Verify the names of the devices and volume labels for the command and then reissue the command with a matching number of devices and volume labels.

DUPRVN, duplicate volume number already mounted

Explanation: A volume of the same relative volume number and the same volume set name is already mounted. This message indicates that you are trying to mount a volume that belongs to a volume set with the same name as a volume set that is already mounted.

User Action: Make sure that you have mounted the correct volume. Verify that you are defining the volume status (/SHARE, /NOSHARE, /GROUP, or /SYSTEM) consistently with volumes already mounted.

DUPVOLNAM, volume label already present in set

Explanation: During an attempt to add a volume to an existing volume set, the specified volume set was found to contain a volume with the label specified.

User Action: Verify the labels on the existing volumes in the set. Choose a new label for the volume you are adding to the set, reinitialize the volume, and reenter the MOUNT/BIND command.

HOMBLKCHK, home block software consistency error

Explanation: During an attempt to add a volume to a disk volume set, the data in the home block of either the new volume or the root volume was found to be inconsistent.

User Action: Because a volume has at least two home blocks, the first occurrence of this error is a warning; the second message indicates a fatal error. Reinitialize the volume being added and retry the operation.

INCONSDEV, inconsistent device types

Explanation: A list of devices in a MOUNT command specified two or more devices that are not the same device type (for example, a disk and a tape). All devices in a volume set must be of the same type.

User Action: Verify that you entered the command string correctly and reissue the command.

RVN1NOTMT, root volume is not mounted

Explanation: The root volume for the volume set specified in the MOUNT/BIND command is not mounted. The root volume must be mounted when you add a new volume to an existing volume set.

User Action: Verify that the root volume is mounted and online; if not, mount it and reissue the command. Verify the device names and volume labels specified in the command and retry the operation.

SETLIMIT, too many volumes in volume set

Explanation: The specified volume set has the maximum number of volumes. You cannot add any more volumes to the set. Because the maximum number of volumes is 255, this error is not likely to occur.

User Action: Check for possible corruption of the home block on the root volume.

VOLINSET, volume is already part of another volume set

Explanation: A volume specified in a request to create or add to a disk volume set is already bound in another volume set. A volume can be part of only one volume set.

User Action: Verify that you have mounted the correct volume. If necessary, initialize the volume to delete all existing data and reissue the command to add the volume to the set.

11.8 Special Conditions

If any volume in a mounted volume set is write-locked, all volumes in the set are also locked. This is true if the volume set or any volume in the volume set is mounted /NOWRITE or if any volume is write-locked as a result of an I/O error.

11.9 Removing Volumes

You cannot remove a volume from a volume set. Once the file system has bound the volumes, the structure cannot be undone. The only way to remove a volume is to back up the entire set, reinitialize each volume in the set, re-create the volume set, and restore the backed-up files.

12.0 DISK SAVE AND COMPRESS

The following notes pertain to the VAX-11 Disk Save and Compress (DSC) utility program DSC2.

12.1 DSC2 Verification Errors

When the system disk of a VAX/VMS system is copied with DSC2, using verify, some verification errors will inevitably appear, even though the system is "quiescent." This occurs because it is not possible to completely stop activity on such files as the error log, paging file, and so on. All DSC verification errors should be checked with a command of the form:

```
$ MCR DMP TI:=DEV:/FI:n:m/HD/BL:0
```

In this command, DEV is the device name and n and m are the components of the file ID reported by DSC2. This command will produce a dump giving the internal file name and owner UIC of the file, allowing the user to tell whether the verification error can be ignored.

Note that the problem described above can be avoided by using stand-alone DSC2. In fact, it is in general a better practice to use stand-alone DSC2 to back up the system disk.

12.2 Allocation Errors Backing Up the System Disk

If an allocation error occurs during the back-up of the system disk with DSC2, it generally indicates that the target disk contains too many bad blocks and does not have enough contiguous space to contain large system files.

If an allocation error occurs, retry the procedure on another disk volume.

A future update to the documentation will include this explanation of allocation errors during the back-up procedure.

12.3 Handling of Multireel Tape

You must have the logical I/O (LOG_IO) privilege when using multireel tapes that are not all mounted before you invoke DSC. If you do not have the LOG_IO privilege and the reels are not all mounted before you invoke DSC, DSC continues to prompt for a new reel after one has been mounted.

13.0 ADJUSTING THE DEFAULT PAGE LENGTH FOR LISTINGS

The default page length for printed listings for VAX/VMS components is 66 lines. (The page length is defined as the number of lines printed between perforations.) This page length is the default for listings created by the following commands (the programs these commands invoke are given in parentheses):

```
EDIT/SOS (SOS)
LINK (VAX-11 Linker)
LINK/RSX11 (RSX-11M Task Builder)
MACRO (Compatibility Mode VAX-11 MACRO Assembler)
MACRO (Native Mode VAX-11 MACRO Assembler)
MACRO/RSX11 (MACRO-11 Assembler)
```

The command procedure LINEPAGE.COM applies updates to all these utilities to modify the default page length. Any user who has read/write access to system files can execute this command procedure. Normally, this means users with system UICs.

The procedure for changing the default page length is:

1. Change the current default disk to the system disk by issuing the command:

```
$ SET DEFAULT SYS$SYSTEM
```

2. Execute the procedure, located in the directory [SYSUPD], specifying the count of lines per page as a parameter:

```
$ @[SYSUPD]LINEPAGE line-count
```

For example, to set the default line count per page to 54, execute the procedure as shown below:

```
$ @[SYSUPD]LINEPAGE 54
```

The LINEPAGE.COM procedure assumes default values for the number of blank lines on each page (these blank lines provide top and botton margins on the listings). These default values can be modified for one or more utilities before execution of the procedure by editing the procedure and changing the values equated to the following symbols:

Symbol Name	Default	Utility
LINK_BLANK	8	VAX-11 Linker
MAC_BLANK	6	MACRO-11 Assembler
MACRO_BLANK	9	Native Mode VAX-11 MACRO Assembler
MAR_BLANK	6	Compatibility Mode VAX-11 MACRO Assembler
SOS_BLANK	11	SOS
TKB_BLANK	6	RSX-11M Task Builder

14.0 REFERRING TO SYSTEM VALUES IN FORTRAN PROGRAMS

VAX-11 FORTRAN-IV PLUS users need not create INCLUDE files to provide values for commonly used system symbols that are globally defined in system libraries. To refer to a system-defined constant equated to a global symbol name (for example, to test return status codes for system services), the FORTRAN programmer can:

- Define each symbol to be used as an EXTERNAL reference
- Refer to symbols thus defined with the %LOC intrinsic function

For example:

```
EXTERNAL SS$_WASCLR
  .
  .
  .
IF (%LOC(SS$_WASCLR) .EQ.  SYS$CLREF(%VAL(5))) THEN ...
```

In the above example, the IF statement compares the value of the symbol SS$_WASCLR with the value returned from the function reference to the Clear Event Flag ($CLREF) system service.

More information on this capability will be included in future updates to the documentation.

15.0 COMMON RUN-TIME LIBRARY

The following notes apply to the VAX-11 Common Run-Time Procedure Library.

15.1 Network Sequential I/O in FORTRAN

VAX-11 FORTRAN IV-PLUS sequential I/O now sets the VAX-11 RMS sequential-only bit upon opening a file. This allows VAX-11 RMS to optimize certain operations, especially over the network. However, a read operation followed by a write operation over the network is no longer permitted, unless these operations are separated by a rewind operation. This change does not affect nonnetwork operations.

For the unusual situations where this restriction can be a problem, the user can clear FAB$V_SQO in a USEROPEN routine.

15.2 Network Node Names in FORTRAN File Specifications

To access a file (other than a file in the guest account) on a remote node, the file name string must contain an account and password separated by a space. However, in VAX-11 FORTRAN IV-PLUS, all space characters are removed from file name strings and all letters are converted to uppercase. The specification of the account and password therefore requires the use of logical names. For example:

```
$ ASSIGN NODE7"""JONES 123""":::DB1:[OPSYS.SRC] TARGET
$ ASSIGN TARGET:FOR001.DAT FOR001
```

In the first of the above ASSIGN commands, the logical name TARGET is equated to the directory [OPSYS.SRC] on the device DB1: on the node identified as NODE7. The file is owned by the account JONES with a password of 123.

The second of the above ASSIGN commands, the logical name TARGET precedes the specification of FOR001.DAT; this assignment defines a file to be opened on logical unit 1 in a FORTRAN program.

Note that the use of logical names for this purpose also provides security in the use of passwords for file specifications.

15.3 Error Message for Inconsistent Records

The Run-Time Library issues the following error message when the record length specified in a FORTRAN OPEN statement is not consistent with the record length of an existing file:

```
INCONSISTENT RECORD LENGTH
```

This message is also issued when the record types are inconsistent, for example, if an OPEN statement specifies a record type of FIXED and the file is VARIABLE.

A future update to the Run-Time Library will add an error message to distinguish between these errors.

15.4 Error in VAX-11 Procedure Calling Standard

The copy of the VAX-11 Procedure Calling Standard printed as Appendix C of the VAX-11 Common Run-Time Procedure Library Reference Manual reversed the reservations of future data type codes. They should be:

```
 24 to 191 reserved to DIGITAL
192 to 255 reserved to DIGITAL's Special Systems Group
           and to customers for their own use
```

15.5 Local Event Flag Resource Allocation Routines

Three routines are now provided to allow users to allocate and deallocate local event flags. A process-wide pool is maintained of available event flags; thus, a routine can use a local event flag without needing to know beforehand which ones are already in use or will be used later.

It is strongly recommended that any user-written program using local event flags use these procedures. The use of global resource allocation procedures greatly enhances a program's modularity and independence from other programs.

15.5.1 LIB$GET_EF (Allocate One Event Flag)

LIB$GET_EF allocates one event flag from a process-wide pool. If no event flags are available for use, an error is returned. Otherwise the number of the event flag is placed in the output parameter.

Format

```
ret-status = LIB$GET_EF (event-flag-number)
```

event-flag-number
: The address of a longword to contain the number of the event flag allocated. If no event flags are available, event-flag-number is set to -1.

Return Status

SS$_NORMAL

: Routine successfully completed.

LIB$_INSEF

: Insufficient event flags. There are no more event flags available for allocation.

Notes

1. Event flag numbers are allocated downward from number 63 to 0.

2. Event flags 0 and 24 through 31 are reserved by the system and are not available to users.

3. Event flag numbers 1 through 23 are preallocated. This is to help avoid conflicts with routines not using the event flag allocation procedures. DIGITAL recommends that you not use event flag numbers 1 through 23.

15.5.2 LIB$FREE_EF (Deallocate an Event Flag)

LIB$FREE_EF is used to deallocate an event flag when it is no longer needed by a routine. An error is returned if the event flag was not allocated or if it was one of the system reserved flags (0 and 24 through 31).

Format

```
return-status = LIB$FREE_EF (event-flag-number)
```

event-flag-number
The address of a longword containing the number of the event flag to be deallocated. This is an input parameter.

Return Status

SS$_NORMAL

Routine successfully completed.

LIB$_EF_ALRFRE

Event flag already free.

LIB$_EF_RESSYS

Event flag reserved to system. This occurs if the event flag number is outside the range of 1 through 23 or 32 through 63.

15.5.3 **LIB$RESERVE_EF (Reserve Event Flag for Future Use)** - LIB$RESERVE_EF is used to reserve a particular event flag. This is different from LIB$GET_EF, which allocates an arbitrary event flag from the pool. LIB$RESERVE_EF returns an error if the event flag is already reserved.

Format

```
return-status = LIB$RESERVE_EF (event-flag-number)
```

event-flag-number
The address of a longword containing the number of the event flag to reserve. This is an input parameter.

Return Status

SS$_NORMAL

Routine successfully completed.

LIB$_EF_ALRRES

Event flag already reserved.

LIB$_EF_RESSYS

Event flag reserved to system. This occurs if the event-flag-number is outside the range of 1 through 23 or 32 through 63.

16.0 SYE

The following notes apply to the SYE error log formatting program. A future update to the documentation will include this information.

16.1 Default Input File

The default input file for SYE is [SYSERR]ERRLOG.OLD, located on the current default disk.

16.2 Errors

SYE internal errors that are not file open errors are reported by means of VAX-11 FORTRAN IV-PLUS error messages. If an error message occurs, you should rerun SYE to be sure that the error was not an operator error.

16.3 Device Errors

You can request that all device errors be reported, or only those that occur on one or more devices that you specify by a device name. SYE prompts for the device name by typing:

```
_device name:   [<all>]  ?
```

By default, errors on all devices are reported (that is, if only a carriage return is typed, all error types are inspected).

If a device name is specified, then device errors and mount/dismount messages whose device names match are selected for further inspection.

SYE accepts generic device names, allowing you to specify that errors be reported for all devices of a particular type (for example DB:), for devices attached to a particular controller (for example DBA:), or for a particular device (for example DBA1:).

Instead of specifying a device name to SYE, you can also request a report on one of three special classes of errors. To request a report on one of these classes of errors, enter one of the following symbols (instead of a device name) in response to the prompt for device name.

- CP--Hardware errors other than device errors. These errors include machine checks, corrected read data, read data substitutes, SBI alerts, SBI faults and asynchronous write errors.

- CO--Configuration changes. These include mount and dismount messages.

- SY--System information. This includes system start up, power recovery, crash/restart, system service and network messages, and system and user bug checks.

  Although time stamp messages are included in the summary totals under system information, they are not included in this option.

You can also use a device name to deselect one device class or special class by prefixing the name with a minus sign (-). For example:

```
_device name:   [<all>] ?  -DMA1:
```

This command string means output all errors other than DMA1: errors. If you specify -SY, all errors except system information entries will be reported.

This format can be used only to exclude one device or one special class of device.

16.4 Listings

Listings will include both device errors and hardware errors (those covered by the CP class).

16.5 Magnetic Tape Errors

TU45 magnetic tape errors are sometimes reported as TE16 errors.

16.6 LPA11-K Errors

Device error reporting is extended by version 1.5 of VAX/VMS to include error reporting for the LPA11-K. The device name is LAx, where x represents unit A, B, C, or D.

17.0 I/O DEVICE DRIVERS

The following notes apply to I/O device drivers.

17.1 Terminal Driver

The following corrections apply to the VAX/VMS I/O User's Guide.

17.1.1 **IO$_SETMODE and IO$_SETCHAR** - The following correction applies to Section 2.4.3.

The P2 argument of the Set Mode function and of the Set Characteristic function specifies the size of the characteristics buffer.

17.1.2 **Page Width Limit** - The following note applies to Section 2.4.3.

The value of the page width limit contained in the Set Mode characteristics buffer can be between 1 and 255.

17.1.3 **IO$_SENSEMODE and IO$_SENSECHAR** - The following two I/O functions should be added to Section 2.4.3: Sense Mode (IO$_SENSEMODE) and Sense Characteristics (IO$_SENSECHAR).

IO$_SENSEMODE returns the process-associated characteristics of the terminal; IO$_SENSECHAR returns the permanent characteristics of the terminal.

These functions take two device/function-dependent arguments, as follows:

- P1 = address of characteristics buffer
- P2 = size of characteristics buffer

17.1.4 **Title of Figure 2-10** - Change the title of Figure 2-10 to read as follows: IOSB Contents -- Set Mode, Set Characteristics, Sense Mode, and Sense Characteristics Functions.

17.2 Magnetic Tape Driver

The following correction applies to Section 4.1 and to Table 4-1 of the VAX/VMS I/O User's Guide.

The VAX/VMS magnetic tape driver also supports the TU45 magnetic tape drive.

17.3 Disk Drivers

The following paragraphs apply to the VAX/VMS I/O User's Guide.

17.3.1 **Set Mode Characteristics Buffer and Set Characteristics Buffer** - The following correction applies to Figures 3-3 and 3-4.

The second longword of the Set Mode characteristics buffer and the second longword of the Set Characteristics buffer contain information on the cylinder, track, and sector configuration of the particular device; that is, number of cylinders per mass storage media volume (bits 31 through 16), number of tracks per cylinder (bits 15 through 8), and number of sectors per track (bits 7 through 0).

17.3.2 **RX01 Console Disk Driver** - The following addition supplements Chapter 3.

The RX01 console disk driver supports Files-11 Structure Level 1 and Level 2 file structures. Access to these file structures is through the standard DCL commands MOUNT and INIT, followed by the appropriate RMS-32 calls. Files in RT-11 format can be read or written with the file exchange facility FLX.

17.3.2.1 **Logical to Physical Translation** - Logical block to physical sector translation adheres to the standard VAX/VMS format. For each 512-byte block selected, the driver reads or writes four 128-byte physical sectors. To minimize rotational latency, the physical sectors are interleaved. This allows the processor time to complete a sector transfer before the next sector in the block reaches the read/write heads. To allow for track-to-track switch time, the next logical sector that falls on a new track is skewed by six sectors. Logical blocks are allocated starting at track 1; track 0 is not used.

Specifically, the translation procedure is as follows:

1. Compute an uncorrected media address using the following dimensions:

   ```
   Number of sectors per track = 26
   Number of tracks per cylinder = 1
   Number of cylinders per disk = 76
   ```

2. Correct the computed address for interleaving and track-to-track skew (in that order) as shown in the following VAX-11 FORTRAN IV-PLUS statements (ISECT is the sector address and ICYL is the cylinder address computed in step 1):

   a. Interleaving:

      ```
      ITEMP = ISECT*2
      IF (ISECT .GT.  12) ITEMP = ITEMP+1
      ISECT = ITEMP
      ```

   b. Skew:

      ```
      ISECT = ISECT + (6*ICYL)
      ISECT = MOD (ISECT, 26)
      ```

3. Set the sector number in the range of 1 to 26 as required by the hardware:

   ```
   ISECT = ISECT + 1
   ```

4. Adjust the cylinder number past the unused cylinder (cylinder 0):

   ```
   ICYL = ICYL + 1
   ```

17.3.2.2 **Supported I/O Calls** - The RX01 console disk driver supports all the functions listed in Table 3-3 of the VAX/VMS I/O User's Guide.

For read or write physical block, the track sector and cylinder parameters described in Figure 3-2 describe a physical 128-bit RX01 sector. Note that the driver does not apply track-to-track skew, cylinder offset, or sector interleaving to this physical media address.

17.3.2.3 **Bootstrap Block Content** - The contents of the RX01 bootstrap block are CPU and operating system dependent. For the LSI-11 Console on the VAX-11/780, the standard bootstrap for the RT-11 operating system is used. Your software support specialist can supply you with more information on the RT-11 bootstrap.

17.4 LPA11 Driver

The directory [SYSMGR] contains two command procedures that pertain to LPA11 users: LPA11STRT.COM and LPA11MREG.COM. LPA11STRT.COM performs the following:

- Runs the LPA11 microcode loader process as a detached process named LALOADER

- Assigns the device LAA0: to the logical name LPA11$0. This name is used by the LPA11 procedure library.

If an installation has an LPA11, it is suggested that this command file be called from SYSTARTUP.COM, so that it gets executed whenever the system is booted.

NOTE

Note that the LALOADER process does not automatically load LPA11 microcode. Instead, this process hibernates until it receives a load request over a mailbox. Microcode can be loaded by running the image [SYSEXE]LALOAD. This process sends load requests to the LALOADER process. These processes are documented in the VAX/VMS I/O User's Guide.

The second command procedure, LPA11MREG.COM, is used in patching the LPA11 driver to preallocate UNIBUS map registers when the driver is loaded. The rationale for this command procedure is as follows. The LPA11 driver can use (especially in multirequest mode) large numbers of UBA map registers for an indefinite amount of time. In fact, if several users started transfers from large buffers, the driver could use every UBA map register. If the driver is patched to preallocate map registers, it will allocate a specified number when the driver is loaded and use only those map registers. This preallocation of map registers prevents the LPA11 from using all of the available map registers; it also ensures that map registers will be available when LPA11 transfers are requested.

As distributed, the LPA11 driver does not preallocate map registers. Running LPA11MREG.COM patches the driver (and creates a new copy) to preallocate a specified number. However, the system must be rebooted to use the new copy of the driver. The number of registers to be preallocated can be changed by editing the command procedure. Examination of the command procedure will show how to do this. A preallocation of 0 means that map registers are not to be preallocated at all.

17.5 Restriction In Device Names

A maximum of eight units can be connected to one I/O device controller. Thus, device names are restricted to the form XXC0 through XXC7.

If a control block for a unit with a unit number greater than 7 is added to the I/O data base by use of the SYSGEN command CONNECT, the resulting system will probably crash.

This restriction on device names will be removed in a future major release of the system.

17.6 Source Files for I/O Device Drivers

Version 1.5 of VAX/VMS includes source files for three unsupported I/O device drivers. These files are as follows:

- DADRIVER.MAR. The source file of the DA11-B driver. The DA11-B is an interprocessor link.
- ADDRIVER.MAR. The source file of the AD11-K driver. The AD11-K is an A/D converter.
- TDRIVER.MAR. The source file of a template I/O driver. You may modify this file to suit your specific needs.

If you want to make use of these files in program development, you will find them in directory [SYSEXE].

18.0 USER ENVIRONMENT TEST PACKAGE (UETP)

To run the User Environment Test Package (UETP), refer to the VAX/VMS UETP User's Guide. However, for a successful run to take place, the following changes and additions to the user's guide must be noted:

Page 1-2, Section 1.1.2

Add this sentence:

> The UETP V1.5 also tests VAX-11 SORT, RSX-11M Executive Directives, and (optional) VAX-11 COBOL-74.

Page 1-4, Figure 1-1, The UETP Master Command Procedure

Replace Figure 1-1 with the following.

UETP.COM

```
$ RUN UETINIT00
$ RUN UETINIT01
$ RUN UETPDEV01
$ @UETNRMS00
$ RUN UETNATV01
$ @SORTUETP
$ @UETFORT00
$ @UETPC74
$ RUN UETLOAD01
$ @UETCOMP00
$ @UETCOMP03
```

Page 1-5, Section 1.2.4

Replace the name UETNATV02.COM with SORTUETP.COM and UETPC74.COM.

Delete all references to "native mode utilities."

Page 1-6, Section 1.2.6

Add this sentence to the end of the section:

This phase also tests Executive Directives with command procedure UETCOMP03.COM.

Page 1-6, Section 1.2.7

Delete UETTERM01.EXE.

Page 2-2, Section 2.1.2

The second paragraph lists privileges required to run the test. The following additional privileges are also required:

VOLPRO
PHY_IO

Page 2-3, Section 2.1.3.1

Both $CREATE/DIRECTORY commands show spaces following the colons (:). This is incorrect syntax. There should be no spaces following the colons.

Pages 2-3 and 2-4, Section 2.1.3.2

Ignore the sentences dealing with $INITIALIZE and $MOUNT commands; that is, after the tape is online, preparation is complete.

Page 2-7, Section 2.2.4

The revised prompt should read:

Now you will be asked to type the device name of a scratch magtape for the FLX and RMS tests (for example MTA0:)
ENTER SCRATCH MAGTAPE (MTCU:) OR <RET>: <device-name> <RET>

Page 2-7, Section 2.2.5

Add Section 2.2.5, as follows.

2.2.5 **Selecting Optional Tests**

The UETP displays the following prompt; your answer to the prompt determines whether optional tests of unbundled software are to be selected.

Now you will be asked if you wish to enter a longer dialogue to select optional tests. The default is NO optional tests.
Enter ALL if you want all the optional tests.
Enter YES if you want some optional tests. (In this case, further prompts will allow you to select specific tests.)
DO YOU WISH TO SELECT OPTIONAL TESTS [Y/N/ALL]: y<RET>

If you answer YES, additional questions will be asked. The number of questions asked depends on the number of unbundled

products the UETP is able to test (two in Version 1.5). If you press <RET> as a response, the UETP assumes NO optional tests are desired. If you answer NO or ALL, no more questions appear.

NOTE

Do not answer YES to any of the optional questions unless the corresponding software product is installed on your system disk. The tests will not execute and the UETP may not complete properly.

The additional questions are:

```
DO YOU HAVE FORTRAN-IV PLUS INSTALLED [Y/N]:  n<RET>

DO YOU HAVE COBOL INSTALLED [Y/N]:  y<RET>
```

These tests are brief (approximately two minutes each) and will type out error messages if they fail.

Page 2-7, Section 2.3

Delete this entire section, and replace it with:

2.3 RUNNING THE ENTIRE UETP

The following dialogue shows how to initiate one or more complete UETP runs.

```
$ @UETP [/OUTPUT=filespec]<RET>
   *** Welcome to UETP V1.5 built 24-JAN-79 ***
UETP STARTING AT dd-mmm-1979 hh:mm:ss.ss

ENTER NUMBER OF LOAD TEST USERS [D]:  n<RET>

ENTER NUMBER OF COMPLETE UETP RUNS [D]:  n<RET>

  Now you will be asked to type the device name of a scratch
  magtape for the FLX and RMS tests (for example MTA0:)
ENTER SCRATCH MAGTAPE ( MTCU:  ) OR <RET>:  <device-name> <RET>

  Now you will be asked if you wish to enter a longer dialogue
  to select optional tests.  The default is NO optional tests.
       Enter ALL if you want all the optional tests.
       Enter YES if you want some optional tests.  (In this case,
       further prompts will allow you to select specific tests.)
DO YOU WISH TO SELECT OPTIONAL TESTS [Y/N/ALL]:  y<RET>

DO YOU HAVE FORTRAN-IV PLUS INSTALLED [Y/N]:  n<RET>
DO YOU HAVE COBOL INSTALLED [Y/N]:  y<RET>
```

When you have entered the first line, optionally specifying a short console log, the UETP responds by asking the questions shown. (See Sections 2.2.2, 2.2.3, 2.2.4, and 2.2.5 for explanations of these questions.) After you have answered all four questions (plus the optional questions, if desired), the UETP proceeds through its entire sequence of tests without further input from you.

Page 2-9, Section 2.3.2

Under the subheading UETPLOG.LOG, delete the line "The native mode utility tests."

Delete the entire item headed UNATIVE.LOG.

The last line of the first full paragraph on this page should read:

> The UETP then deletes the individual load test and device test logs.

Page 2-10, Table 2-3

In the column headed "Commands to Invoke the Test," the following command string is invalid:

```
$ RUN UETTAPE00
```

It should be:

```
$ @UETTAPE00
```

Page 2-11, Section 2.4.2

Delete the line containing UETNATV02.COM and replace it with: UETPC74.COM (COBOL 74) and SORTUETP.COM (VAX-11 SORT).

Page 2-12, Section 2.4.2.2

Delete this entire section, and replace it with:

> 2.4.2.2 **VAX-11 Sort Test** - The VAX-11 Sort test exercises the functions of the VAX-11 Sort utility. To activate the test, you run the command procedure SORTUETP.COM as follows:
>
> ```
> $ @SORTUETP [/OUTPUT=filespec] <RET>
> ```
>
> The sorted data are written into sequential, relative or indexed sequential files. After valid Sort commands are tried, several invalid ones are issued to see if they give the expected error message. There should be eight fatal errors issued of the form:
>
> ```
> %SORT-F-text
> ```
>
> If the SORT test is run as part of the UETP, the invalid commands are not tried.
>
> After the SORT command has been tested, SORT is tested as a called subroutine from three high-level languages (BLISS, COBOL and FORTRAN). The tests are executed by linking and running precompiled object modules (the unbundled compilers are not necessary for the SORT test). Furthermore, the COBOL program is not automatically run unless COBOL is installed and you indicate this to UETP by defining the symbol UET$COB and then running command procedure SORTUETP:
>
> ```
> $ UET$COB:==Y
> $ @SORTUETP [/OUTPUT=filespec] <RET>
> ```
>
> The SORT tests take about five minutes.

Page 2-13, Section 2.4.2.3

Change all references to the name MAGTAP to UET$MAGTAP.

Page 2-14, Section 2.4.2.4

Add the following to the end of Section 2.4.2.4.

When the FORTRAN test has run successfully, the console log should look like:

```
*** VAX-11 FORTRAN-IV PLUS TEST BEGINNING ***
*** VAX-11 FORTRAN-IV PLUS TEST ENDING ***
```

Page 2-14, Section 2.4.2.5

Add Section 2.4.2.5, as follows.

2.4.2.5 **VAX-11 COBOL-74 Test** - To run the VAX-11 COBOL-74 compiler test, enter the following command:

```
$ @UETPC74
```

Commands within the procedure UETPC74.COM compile, link, and run three COBOL programs. The first test creates a sequential file of 100 fixed-length records. The second test creates a relative file with 100 records. The third test reads the first file sequentially and writes its output to logical device LP: which is assigned to a disk file. Upon completion of these tests all created files are deleted.

If the test is successful, the console log should show:

```
*** VAX-11 COBOL-74 TEST BEGINNING ***

*** VAX-11 COBOL-74 TEST ENDING ***
```

Because COBOL is unbundled, it must be completely installed before these tests are run. The tests themselves are distributed with the COBOL kit. No error messages should be received from these tests. They run about two minutes.

Pages 2-16 and 2-17, Sections 2.4.4 and 2.4.4.1

Change all occurrences of the name MAGTAP to UET$MAGTAP.

Page 2-17, Section 2.4.4.3

Add Section 2.4.4.3, as follows.

2.4.4.3 **RSX-11M Executive Directives Test** - To run the Executive Directives part of the compatibility mode phase, use the following command:

```
$ @UETCOMP03 [/OUTPUT=filespec] <RET>
```

The command procedure UETCOMP03.COM calls on various task images to test the RSX-11M Executive Directives. This phase should run two or three minutes and should not produce any errors. While this test is running, several lines of characters should be printed at your terminal following a line which says:

```
*** PRINT TEST ***
```

When the test is run as part of the UETP, all of the output is directed to a file named UCOMP.LOG, which is later copied into the file UETPLOG.LOG.

Page A-1, Section A.2.1

On the last line, delete the space after the colon (:).

Page A-2, Section A.2.2

Delete the $INITIALIZE and $MOUNT commands from the magnetic tape setup.

Page A-3, Section A.3

Replace Section A.3, as follows.

A.3 **RUNNING THE ENTIRE UETP**

To initiate the UETP package, enter a call to the UETP master command procedure and respond to the prompts shown below:

```
$ @UETP [/OUTPUT=filespec]<RET>
   *** Welcome to UETP V1.5 built 24-JAN-79 ***
UETP STARTING AT dd-mmm-1979 hh:mm:ss.ss

ENTER NUMBER OF LOAD TEST USERS [D]:  n<RET>

ENTER NUMBER OF COMPLETE UETP RUNS [D]:  n<RET>

  Now you will be asked to type the device name of a scratch
  magtape for the FLX and RMS tests (for example MTA0:)
ENTER SCRATCH MAGTAPE ( MTCU:  ) OR <RET>:  <device-name> <RET>

  Now you will be asked if you wish to enter a longer dialogue
  to select optional tests.  The default is NO optional tests.
       Enter ALL if you want all the optional tests.
       Enter YES if you want some optional tests.  (In this case,
       further prompts will allow you to select specific tests.)

DO YOU WISH TO SELECT OPTIONAL TESTS [Y/N/ALL]:  y<RET>

DO YOU HAVE FORTRAN-IV PLUS INSTALLED [Y/N]:  n<RET>
DO YOU HAVE COBOL INSTALLED [Y/N]:  y<RET>
```

Sections 2.2.2, 2.2.3, 2.2.4, and 2.2.5 explain the prompts in detail. Table 2-2 provides a guideline for choosing the maximum number of load test users according to the amount of memory in the VAX/VMS system being tested.

Use CTRL/Y or CTRL/C to interrupt the tests (see Section 2.3.1).

Page A-3, Section A.4.1

On the fourth line from the bottom, the following command string is invalid:

```
$ RUN UETTAPE00
```

It should be:

```
$ @UETTAPE00
```

Page A-4, Section A.4.2

Replace Section A.4.2, as follows.

A.4.2 **The Native Mode Tests**

The native mode test phase includes these separate tests:

- The system services test
- The VAX-11 Record Management Services (RMS) test
- The VAX-11 SORT test
- The VAX-11 FORTRAN-IV PLUS compiler test (optional)
- The VAX-11 COBOL-74 compiler test (optional)

To run the system services test, issue the following command:

```
$ RUN UETNATV01
```

To run the VAX-11 RMS test, issue the following commands:

```
[$ UET$MAGTAP:==device-name:]
$ @UETNRMS00
```

Note that the RMS test cannot include magnetic tape tests unless you explicitly define the symbol UET$MAGTAP as shown above.

To run the VAX-11 SORT test, issue the following commands:

```
[$ UET$COB:==Y]
$ @SORTUETP [/OUTPUT=filespec]
```

To run the optional VAX-11 FORTRAN-IV PLUS compiler test, issue the following command:

```
$ @UETFORT00
```

To run the optional VAX-11 COBOL-74 compiler test, issue the following command:

```
$ @UETPC74
```

Page A-5, Section A.4.4

Change the name MAGTAP to UET$MAGTAP.

Page A-5, Section A.4.4

Add the following to the end of this section.

To run the RSX-11M Executive Directives test, issue:

```
$ @UETCOMP03 [/OUTPUT=filespec]
```

19.0 RECOMMENDED CHANGE TO FIELD SERVICE ACCOUNT

If you have not already done so, you should alter the UAF record for the field service account (with user's name FIELD) as follows:

- The PSWAPM privilege should be granted to the account.
- The limits WSQUOTA and WSDEFAULT should be raised to 150 pages.

20.0 SPECIFYING PRINTER FORMS

The system manager can specify the various types of line printer forms that are available at an installation by making entries in the file [SYSMGR]FORMSTYPE.DAT, which is located on the system disk.

Besides form entries, this file contains information about the format and content of entries in the file and information about the use of the file.

21.0 SYSTEM DUMP ANALYZER

Refer to Appendix E for instructions for installing and using the System Dump Analyzer (SDA), an unsupported utility used in determining the causes of system crashes.

22.0 VAX-11 SYMBOLIC DEBUGGER

Version 1.5 of the VAX-11 Symbolic Debugger includes all the functionality of version 1.0 and supports two additional languages: VAX-11 BLISS-32 and VAX-11 COBOL-74.

VAX-11 BLISS-32 functions that are supported include BLISS radix control operators, the "dot" indirect operator, BLISS expression syntax (excluding function and routine calls), and the REF attribute. Also supported are BLISS predefined structures.

VAX-11 BLISS-32 functions not supported include BLISS general structures, %REF, and CH$ functions. For more detail, refer to the VAX-11 BLISS-32 User's Guide.

VAX-11 COBOL-74 support allows the setting of watchpoints by line or virtual address, setting of breakpoints and tracepoints by line only, 31 character names, and support for additional data types, such as packed decimal.

Not included is support for the quadword data type (that is, signed and unsigned data items of 10 through 18 digits, for example, computational items defined by the following picture clauses: PIC S9(10) through PIC S9(18)). There is also a limitation on the qualification of names: the last occurrence of a qualified data name is the one referred to when an unqualified name is specified. Further information can be found in the VAX-11 COBOL-74 User's Guide.

23.0 PATCHES AND UPDATES APPLIED TO VAX/VMS

Appendix B summarizes the patches that were applied to version 1.0 of VAX/VMS after the microfiche of the source was made. These patches were included in version 1.0 of VAX/VMS.

Appendix C summarizes the updates that were released with version 1.01 of VAX/VMS.

Appendix D summarizes the updates that are being released with this version, 1.5, of VAX/VMS.

APPENDIX A

USING PDP-11 BASIC-PLUS-2/VAX

This appendix provides a special supplement to the BASIC-PLUS-2 RSX-11M/IAS User's Guide. It describes how to use the PDP-11 BASIC-PLUS-2/VAX* compiler on the VAX/VMS operating system. This supplement describes only:

- Aspects of BASIC program development that are different from the procedures described in the BASIC-PLUS-2 RSX-11M/IAS User's Guide
- Features of the VAX/VMS operating system and command language that are pertinent to the discussion of BASIC program development

Procedures that are common to all users of VAX/VMS are introduced in the VAX/VMS Primer. The VAX/VMS Command Language User's Guide provides detailed reference information on DCL commands.

In this supplement, user-entered commands and data are underlined to distinguish them from messages and responses displayed by the system.

A.1 COMPATIBILITY MODE

When executed on the VAX/VMS operating system, the BASIC compiler runs in RSX-11M compatibility mode and creates code that executes in compatibility mode.

This supplement describes the commands available in the default VAX/VMS command interpreter for BASIC program

---

*The PDP-11 BASIC-PLUS-2/VAX compiler is referred to simply as BASIC throughout this supplement.

development. Related and functionally equivalent commands are also available in the MCR command interpreter. For information on the MCR command interpreter, refer to the VAX-11/RSX-11M User's Guide.

A.2 PROGRAM DEVELOPMENT ON VAX/VMS

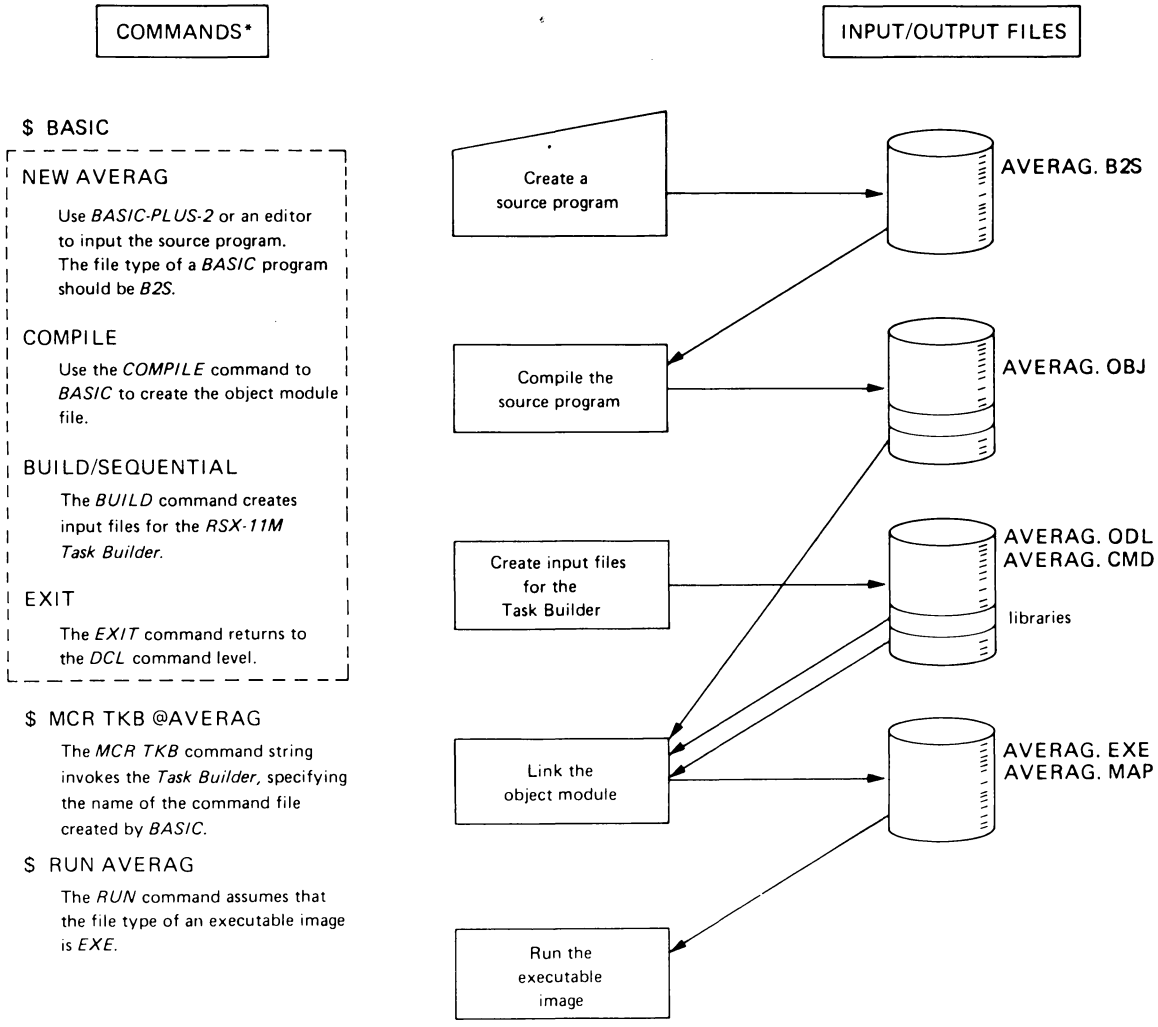
The DCL command BASIC invokes the BASIC compiler, as follows:

```
$ BASIC<RET>

Basic Plus 2 V01-50


Basic2
```

The Basic2 prompt indicates that the compiler is ready to accept commands.

Use BASIC commands to create, modify, and compile source programs, as described in the BASIC-PLUS-2 RSX-11M/IAS User's Guide.

Figure A-1 shows the steps required to prepare and run a BASIC program on VAX/VMS. This figure lists the commands to use at each step and indicates the input and output files and the default file types used by each command.



*Commands within the area marked by broken lines are commands processed by the *BASIC* compiler.

Figure A-1 Steps in BASIC Program Development

A.3 CREATING AND COMPILING A SOURCE PROGRAM

The user's guide for BASIC describes the BASIC commands you can use to create and make changes to your files, and to compile the programs to create object modules.

When you create a file with the SAVE command, BASIC gives the file a file type of B2S, by default. The COMPILE command uses the file name of the file currently in memory and creates an object module with that file name and a file type of OBJ.

You can also use an interactive editor to create and modify your BASIC programs. The default editor on VAX/VMS, called SOS, is invoked with the EDIT command. For example, to create a BASIC program named AVERAG you would issue the command:

```
$ EDIT AVERAG.B2S<RET>
```

The SOS editor is introduced in the VAX/VMS Primer. For complete details on how to use this editor, see the VAX-11 Text Editing Reference Manual.

After editing a file, you can invoke BASIC and specify the name of the file to bring it in memory. For example:

```
$ BASIC<RET>

Basic Plus 2 V01-50

OLD AVERAG<RET>
```

Then, use BASIC commands to create input files for the RSX-11M Task Builder.

A.4 TASK BUILDER INPUT FILES

The BUILD command creates two files required for input to the RSX-11M Task Builder:

- A command file that has the same name as the source file currently in memory and a file type of CMD
- An overlay description file with the same file name and a file type of ODL

The command file specifies input and output files for the Task Builder. It also contains specifications for Task Builder options required for BASIC program execution. The

overlay description file specifies library routines required for the program's execution.

For example, after creating a source file named AVERAG, you can issue the following commands to BASIC:

```
BUILD/SEQUENTIAL<RET>

Basic2

EXIT<RET>
```

The BUILD command creates the files AVERAG.CMD and AVERAG.ODL. The EXIT command restores the DCL command environment, where you can link and execute the program.

A.5 USING THE TASK BUILDER

The Task Builder links one or more object modules (as specified in the command file) into an image file that can be executed. The command file created by BASIC's BUILD command contains all the necessary input for the Task Builder. To invoke the Task Builder to link the program AVERAG, issue the command:

```
$ MCR TKB @AVERAG<RET>
```

The DCL command MCR passes the TKB command to the system to invoke the Task Builder. The at sign (@) character precedes the specification of the command file AVERAG to indicate that the Task Builder should obtain its input from the file AVERAG.CMD.

In VAX/VMS, the default output file type for an executable image file is EXE. Thus, when the Task Builder completes in the above example, it creates a file named AVERAG.EXE. The default command file also requests the Task Builder to create a memory allocation file; this file is named AVERAG.MAP.

If you want to request any Task Builder options, edit the command file and add or change the appropriate switches.

A.6 EXECUTING THE PROGRAM

The DCL command RUN executes an image created with the TKB command. To execute the image AVERAG.EXE created in the example above, issue the command:

```
$ RUN AVERAG<RET>
$
```

The RUN command uses the default file type of EXE. When the image completes execution, the DCL command interpreter displays a prompt for the next command.

A.7 LOGICAL NAMES

The VAX/VMS operating system provides a logical naming capability that lets you equate a physical device name or file name to a temporary name. The temporary name is called a logical name; the physical device or file specification is called the equivalence name. When you write a BASIC program, you can refer to an input or output file by its logical name. Then, before you issue the RUN command to execute the program, you can assign an equivalence name to the logical name.

For example, suppose the BASIC program named FILES contains the statements:

```
10    OPEN "INFILE" AS FILE #1% SEQUENTIAL
20    OPEN "OUTFILE" AS FILE #2% SEQUENTIAL
      .
      .
      .
```

The names INFILE and OUTFILE can be logical names. Before you execute the image FILES, you can equate file names to these logical names with the DCL command ASSIGN. For example:

```
$ ASSIGN TEST1.OUT OUTFILE<RET>
$ ASSIGN PAYROLL.DAT INFILE<RET>
$ RUN FILES<RET>
```

The first ASSIGN command equates the file TEST1.OUT with the logical name OUTFILE, and the second ASSIGN command equates the file PAYROLL.DAT with the logical name INFILE. When the program FILES executes, it reads from the file PAYROLL.DAT and writes to the file TEST1.OUT.

Subsequently, you can reassign these logical names to different files for a different run of the program, so that the program reads from and writes to different files.

If you do not assign equivalence names for the logical names when you run the program, the system assumes that INFILE and OUTFILE are physical file specifications. The program reads from a file named INFILE. and writes to a file named

OUTFILE. (both files have null file types).

A.8 DEFAULT LOGICAL NAMES

The VAX/VMS operating system provides default logical names for each user process. These logical names are:

SYS$INPUT	input stream
SYS$OUTPUT	output stream
SYS$ERROR	error stream

For interactive users these logical names are all equated to the terminal.

When a BASIC program uses INPUT and PRINT statements that do not specify file numbers, input for the INPUT statement and output written by the PRINT statement are directed to the current devices for SYS$INPUT and SYS$OUTPUT, respectively.

APPENDIX B

SUMMARY OF PATCHES INCLUDED IN VERSION 1.0 OF VAX/VMS

This appendix summarizes the patches that were applied to version 1.0 of VAX/VMS after the microfiche of the source was made.

```
COVMS.JNL;1                    28-AUG-1978 09:18:20.48          Page 1

$ SET DEFAULT EXED$:[COVMS.OBJ]
$ IF P1 .NES "" THEN SET DEFAU 'P1'
$ COPY COVMS.EXE DBA2:[COVMS.UPD]*/NEW_VERSION
$ MCR HEXZAP
;
; IDENTIFICATION:
;
;       CAM001 - 25-AUG-1978
;
; AUTHOR:
;
;       C. A. MONIA
;
; CHANGE DESCRIPTION:
;
;       ECC ERROR CORRECTIONS NOT APPLIED PROPERLY IF BUFFER IS NOT PAGE-
;       ALIGNED.
;
DBA2:[COVMS.UPD]COVMS.EXE
1:8008;0R
0,56\
F0V
C1
0,57\
52V
A5
0,58\
00V
6C
0,59\
09V
52
0,5A\
51V
53
0,5B\
3CV
F0
0,5C\
A5V
53
0,5D\
6CV
00
0,5E\
53V
09
0,5F\
8CV
51
0,60\
50V
8C
0,61\
43V
50
0,62\
61V
61
```

```
-1*
-1*
0,56\
0,57\
0,58\
0,59\
0,5A\
0,5B\
0,5C\
0,5D\
0,5E\
0,5F\
0,60\
0,61\
0,62\
X
$ EXIT
```

```
  PATCH BASE LEVEL X5.04  15 DEC 77

IMAGE FILE BEING PATCHED:       "DBA2:[RL1PATCH]DMDRIVER.EXE;2"
JOURNAL FILE:                   "DBA2:[RL1PATCH]DMDRIVER.JNL;1"
DATE/TIME OF PATCH:             23-AUG-1978 15:32:12.98

%PATCH-I-NOLCL, image does not contain local symbols
%PATCH-I-NOGBL, some or all global symbols not accessible
PATCH>DEFINE BASE=80000068
symbol "BASE" defined as 80000068
PATCH>EXAMINE/I BASE+3C9
80000431:  BITW    #0080,B^0A(R4)
PATCH>E
80000437:  06DD0C12
PATCH>E/I .
80000437:  BNEQ    80000445
PATCH>REPLACE/I BASE+3C9
OLD>  'BITW #0080,B^0A(R4)'
OLD>  'BNEQ 80000445'
OLD>  EXIT
NEW>  'BITW #2000,B^0A(R4)'
NEW>  'BEQL 80000445'
NEW>  EXIT
old:    80000431:  BITW    #0080,B^0A(R4)
old:    80000437:  BNEQ    80000445
new:    80000431:  BITW    #2000,B^0A(R4)
new:    80000437:  BEQL    80000445
PATCH>UPDATE
UPDATING IMAGE FILE "DBA2:[RL1PATCH]DMDRIVER.EXE;3"
PATCH>EXIT
```

```
  PATCH BASE LEVEL X5.04  15 DEC 77

IMAGE FILE BEING PATCHED:       "DBA2:[RL1PATCH]ERRFMT.EXE;1"
JOURNAL FILE:                   "DBA2:[RL1PATCH]ERRFMT.JNL;1"
DATE/TIME OF PATCH:             24-AUG-1978 15:44:57.73

%PATCH-I-NOGBL, some or all global symbols not accessible
PATCH>REPLACE/I 7A4
OLD>  'MOVW B^06(R9),W^049B'
OLD>  EXIT
NEW>  'MOVW B^0E(R2),W^0498'
NEW>  EXIT
old:    000007A4:  MOVW    B^06(R9),W^00000498
new:    000007A4:  MOVW    B^0E(R2),W^00000498
PATCH>UPDATE
UPDATING IMAGE FILE "DBA2:[RL1PATCH]ERRFMT.EXE;2"
PATCH>EXIT
```

```
PATCH.JNL;1                    28-AUG-1978 09:18:30.21          Page 1


  PATCH BASE LEVEL X5.04  15 DEC 77

IMAGE FILE BEING PATCHED:        "DBA2:[PATCH.OBJ]KATHY.EXE;1"
JOURNAL FILE:                    "DBA2:[PATCH.OBJ]KATHY.JNL;1"
DATE/TIME OF PATCH:              25-AUG-1978 15:00:54.49

%PATCH-I-NOGBL, some or all global symbols not accessible
PATCH>SET ECO 1
PATCH>RE /I
LOC>  ^X00017874
OLD>  'MOVL (R10),R0'
OLD>  EXIT
NEW>  'MOVL R10,R0'
NEW>  EXI
old:    00017874:  MOVL    (R10),R0
new:    00017874:  MOVL    R10,R0
PATCH>SE MODU
NAM>  PATWRT
NAM>  EXI
PATCH>RE /I
LOC>  ^X00019503
OLD>  'MOVL B^^X0000000C(R7),L^^X00002914'
OLD>  EXIT
NEW>  'BLBC B^^X00000008(R7),KAT'
NEW>  'TSTL B^^X0000000C(R7)'
NEW>  'BEQL    PAT$WRTIMG+3C5'
NEW>  'KAT :MOVL B^^X0000000C(R7),L^^X00002914'
NEW>  EXI
old:    PATWRT\PAT$WRTIMG+3CF:  MOVL    B^0C(R7),L^00002914
new:    PATWRT\PAT$WRTIMG+3CF:  JMP     L^PAA
new:    PATWRT\PAT$WRTIMG+3D5:  NOP
new:    PATWRT\PAT$WRTIMG+3D6:  NOP
new:    PAA:  BLBC    B^08(R7),KAT
new:    00022C04:  TSTL    B^0C(R7)
new:    00022C07:  BNEQ    KAT
new:    00022C09:  JMP     L^PATWRT\PAT$WRTIMG+3C5
new:    KAT:  MOVL    B^0C(R7),L^00002914
new:    00022C17:  JMP     L^PATWRT\PAT$WRTIMG+3D7
symbol "KAT" defined as 0001950C
symbol "KAT" redefined from 0001950C to 00022C0F
PATCH>SE MODU
NAM>  PATERR
NAM>  EXI
PATCH>RE /I
LOC>  ^X00013C14
OLD>  'CALLS #^X00000001,@#80000140'
OLD>  EXIT
NEW>  'BBSS #^X0000001C,(SP),LBL2'
NEW>  'LBL2 :CALLS #^X00000001,@#80000140'
NEW>  EXI
old:    PATERR\PAT$ERROR_EXIT+5:  CALLS   #01,@#80000140
new:    PATERR\PAT$ERROR_EXIT+5:  JMP     L^00022C1D
new:    PATERR\PAT$ERROR_EXIT+0B:  NOP
new:    00022C1D:  BBSS    #1C,(SP),LBL2
new:    LBL2:  CALLS   #01,@#80000140
new:    00022C28:  JMP     L^PATERR\PAT$ERROR_EXIT+0C
symbol "LBL2" defined as 00013C18
symbol "LBL2" redefined from 00013C18 to 00022C21
```

```
PATCH>U
UPDATING IMAGE FILE "DBA2:[PATCH.OBJ]KATHY.EXE;2"
PATCH>EXI
```

```
  PATCH BASE LEVEL X5.04  15 DEC 77

IMAGE FILE BEING PATCHED:       "DB2:[117,30]RMS.EXE;2"
JOURNAL FILE:                   "DB2:[117,30]RMS.JNL;1"
DATE/TIME OF PATCH:             22-AUG-1978 17:32:08.36

%PATCH-I-NOLCL, image does not contain local symbols
%PATCH-I-NOGBL, some or all global symbols not accessible
PATCH>E 80007200+4
80007204:  00007208
PATCH>D .=\+80000000
old:    80007204:  00007208
new:    80007204:  80007208
PATCH>SET PATCH 80007200
PATCH>E/I 80006E55
80006E55:  BITL    #4C0F0000,B^04(R8)
PATCH>REPLACE/INST .
OLD>  'BITL #4C0F0000,B^4(R8)'
OLD>  EXIT
NEW>  'BICL2 #02000020,B^4(R8)'
NEW>  'BITL #4C0F0000,B^4(R8)'
NEW>  EXIT
old:    80006E55:  BITL    #4C0F0000,B^04(R8)
new:    80006E55:  BRW     PAB
new:    80006E58:  NOP
new:    80006E59:  NOP
new:    80006E5A:  NOP
new:    80006E5B:  NOP
new:    80006E5C:  NOP
new:    PAB:  BICL2   #02000020,B^04(R8)
new:    80007210:  BITL    #4C0F0000,B^04(R8)
new:    80007218:  BRW     80006E5D
PATCH>E/I 8000620D+11
8000621E:  BSBW    800039ED
PATCH>REPLACE/INST .
OLD>  'BSBW 800039ED'
OLD>  'MOVL R6,W^150(R7)'
OLD>  EXIT
NEW>  'CLRL R6'
NEW>  'BSBW 800039ED'
NEW>  'BEQL 8000620D+3B'
NEW>  'MOVL R6,W^150(R7)'
NEW>  EXIT
old:    8000621E:  BSBW    800039ED
old:    80006221:  MOVL    R6,W^0150(R7)
new:    8000621E:  BRW     8000721B
new:    80006221:  NOP
new:    80006222:  NOP
new:    80006223:  NOP
new:    80006224:  NOP
new:    80006225:  NOP
new:    8000721B:  CLRL    R6
new:    8000721D:  BSBW    800039ED
new:    80007220:  BNEQ    80007225
new:    80007222:  BRW     80006248
new:    80007225:  MOVL    R6,W^0150(R7)
new:    8000722A:  BRW     80006226
```

```
RMS.JNL;3                          28-AUG-1978 09:18:34.03          Page 2

PATCH>E/I 8000620D+108
80006315:  MOVL    W^0150(R7),R6
PATCH>INSERT/INST
LOC>  .
OLD>  'MOVL W^150(R7),R6'
NEW>  'BEQL 8000620D+112'
NEW>  'MOVL B^44(R7),B^10(R8)'
NEW>  EXIT
old:    80006315:  MOVL    W^0150(R7),R6
new:    80006315:  BRW     8000722D
new:    80006318:  NOP
new:    80006319:  NOP
new:    8000722D:  MOVL    W^0150(R7),R6
new:    80007232:  BNEQ    80007237
new:    80007234:  BRW     8000631F
new:    80007237:  MOVL    B^44(R7),B^10(R8)
new:    8000723C:  BRW     8000631A
PATCH>UPDATE
UPDATING IMAGE FILE "DB2:[117,30]RMS.EXE;3"
PATCH>EXIT


  PATCH BASE LEVEL X5.04  15 DEC 77

IMAGE FILE BEING PATCHED:       "DBA2:[RL1PATCH]RMS.EXE;3"
JOURNAL FILE:                   "DBA2:[RL1PATCH]RMS.JNL;2"
DATE/TIME OF PATCH:             24-AUG-1978 09:33:02.74

%PATCH-I-NOLCL, image does not contain local symbols
%PATCH-I-NOGBL, some or all global symbols not accessible
PATCH>SET PATCH 80007200
PATCH>DEF MAGTA=80005186
symbol "MAGTA" defined as 80005186
PATCH>INSERT/I MAGTA+4A
OLD>  'MOVZWL #28,R0'
NEW>  'BBC #14,(R9),NOTMBX'
NEW>  'BISB2 #40,R0'
NEW>  'NOTMBX: NOP'
NEW>  EXIT
old:    80005200:  MOVZWL  #28,R0
new:    80005200:  BRW     PAB
new:    PAB:  MOVZWL  #28,R0
new:    80007242:  BBC     #14,(R9),NOTMBX
new:    80007246:  BISB2   #40,R0
new:    NOTMBX:  NOP
new:    8000724B:  BRW     80005203
symbol "NOTMBX" defined as 8000520B
symbol "NOTMBX" redefined from 8000520B to 8000724A
PATCH>UPDATE
UPDATING IMAGE FILE "DBA2:[RL1PATCH]RMS.EXE;4"
PATCH>EXIT


  PATCH BASE LEVEL X5.04  15 DEC 77

IMAGE FILE BEING PATCHED:       "DBA2:[RL1PATCH]RMS.EXE;4"
JOURNAL FILE:                   "DBA2:[RL1PATCH]RMS.JNL;3"
DATE/TIME OF PATCH:             24-AUG-1978 09:54:36.28
```

```
RMS.JNL;3                          28-AUG-1978 09:18:34.03          Page 3

%PATCH-I-NOLCL, image does not contain local symbols
%PATCH-I-NOGBL, some or all global symbols not accessible
PATCH>SET PATCH 80007200
PATCH>DEF OPEN=80003FF7
symbol "OPEN" defined as 80003FF7
PATCH>INSERT/I OPEN+6E
OLD>  'MOVW B^4E(R9),B^3A(R9)'
NEW>  'MOVW B^4E(R9),B^14(R8)'
NEW>  EXIT
old:    80004065:  MOVW    B^4E(R9),B^3A(R9)
new:    80004065:  BRW     PAB
new:    80004068:  NOP
new:    80004069:  NOP
new:    PAB:  MOVW    B^4E(R9),B^3A(R9)
new:    80007253:  MOVW    B^4E(R9),B^14(R8)
new:    80007258:  BRW     8000406A
PATCH>UPDATE
UPDATING IMAGE FILE "DBA2:[RL1PATCH]RMS.EXE;5"
PATCH>EXIT
```

```
  PATCH BASE LEVEL X5.04  15 DEC 77

IMAGE FILE BEING PATCHED:       "DBA2:[RL1PATCH]SHOW.EXE;1"
JOURNAL FILE:                   "DBA2:[RL1PATCH]SHOW.JNL;1"
DATE/TIME OF PATCH:             24-AUG-1978 14:43:54.58

%PATCH-I-NOLCL, image does not contain local symbols
%PATCH-I-NOGBL, some or all global symbols not accessible
PATCH>DEF
NAM>  X1
NEW>  ^X00004495
NAM>  EXI
symbol "X1" defined as 00004495
PATCH>E /I -
_>    ^X0000461E
0000461E:  CALLS   #00,W^00004967
PATCH>RE /I
LOC>  ^X0000461E
OLD>  'CALLS #^X00000000,W^^X00004967'
OLD>  EXIT
NEW>  'PUSHL #^X00000000'
NEW>  'PUSHAL W^^X00004967'
NEW>  'CALLS #^X00000002,@#^X80000090'
NEW>  EXI
old:    0000461E:  CALLS   #00,W^00004967
new:    0000461E:  BRW     PAA
new:    00004621:  NOP
new:    00004622:  NOP
new:    PAA:  PUSHL   #00
new:    00005202:  PUSHAL  W^00004967
new:    00005206:  CALLS   #02,@#80000090
new:    0000520D:  BRW     00004623
PATCH>E /I -
_>    ^X00004623
00004623:  RET
PATCH>U
UPDATING IMAGE FILE "DBA2:[RL1PATCH]SHOW.EXE;2"
```

```
SYS.JNL;6                       28-AUG-1978 09:18:44.25         Page 1


  PATCH BASE LEVEL X5.04  15 DEC 77

IMAGE FILE BEING PATCHED:       "DB4:[SYSEXE]SYS.EXE;3"
JOURNAL FILE:                   "DB4:[SYSEXE]SYS.JNL;1"
DATE/TIME OF PATCH:             22-AUG-1978 09:33:38.09

%PATCH-I-NOLCL, image does not contain local symbols
%PATCH-I-NOGBL, some or all global symbols not accessible
PATCH>DEPOSIT 80014600+78C+0C=7FFE1FBC+8
old:    80014DC8:  7FFE1FC0
new:    80014DC8:  7FFE1FC4
PATCH>UPDATE
UPDATING IMAGE FILE "DB4:[SYSEXE]SYS.EXE;4"
PATCH>EXIT


  PATCH BASE LEVEL X5.04  15 DEC 77

IMAGE FILE BEING PATCHED:       "DBA2:[RL1PATCH]SYS.EXE;4"
JOURNAL FILE:                   "DBA2:[RL1PATCH]SYS.JNL;1"
DATE/TIME OF PATCH:             23-AUG-1978 16:02:58.42

%PATCH-I-NOLCL, image does not contain local symbols
%PATCH-I-NOGBL, some or all global symbols not accessible
PATCH>DEF MBDRV=8000127C
symbol "MBDRV" defined as 8000127C
PATCH>SET PATCH 80004600
PATCH>REPLACE/I MBDRV+0EF
OLD>  'MOVQ R6,-(SP)'
OLD>  EXIT
NEW>  'PUSHR #00F0'
NEW>  EXIT
old:    8000136B:  MOVQ    R6,-(SP)
new:    8000136B:  BRW     PAB
new:    PAB:  PUSHR   #00F0
new:    8000460C:  BRW     8000136E
PATCH>REPLACE/I MBDRV+169
OLD>  'MOVQ (SP)+,R6'
OLD>  EXIT
NEW>  'POPR #00F0'
NEW>  EXIT
old:    800013E5:  MOVQ    (SP)+,R6
new:    800013E5:  BRW     8000460F
new:    8000460F:  POPR    #00F0
new:    80004613:  BRW     800013E8
PATCH>UPDATE
UPDATING IMAGE FILE "DBA2:[RL1PATCH]SYS.EXE;5"
PATCH>EXIT


  PATCH BASE LEVEL X5.04  15 DEC 77

IMAGE FILE BEING PATCHED:       "DBA2:[RL1PATCH]SYS.EXE;5"
JOURNAL FILE:                   "DBA2:[RL1PATCH]SYS.JNL;3"
DATE/TIME OF PATCH:             24-AUG-1978 00:07:47.37

%PATCH-I-NOLCL, image does not contain local symbols
```

```
SYS.JNL;6                          28-AUG-1978 09:18:44.25          Page 2

%PATCH-I-NOGBL, some or all global symbols not accessible
PATCH>SET PATCH 80010E00
PATCH>DEF ASCEFC=8000BF21
symbol "ASCEFC" defined as 8000BF21
PATCH>REPLACE/I ASCEFC+53
OLD>  'INCW B^4A(R4)'
OLD>  EXIT
NEW>  'DECW B^4A(R4)'
NEW>  EXIT
old:    8000BF74:  INCW    B^4A(R4)
new:    8000BF74:  DECW    B^4A(R4)
PATCH>INSERT/I ASCEFC+59
OLD>  'BLBS R0,8000BF80'
NEW>  'INCW B^4A(R4)'
NEW>  EXIT
old:    8000BF7A:  BLBS    R0,8000BF80
new:    8000BF7A:  BRW     PAB
new:    PAB:  BLBC    R0,80010E0E
new:    80010E0B:  BRW     8000BF80
new:    80010E0E:  INCW    B^4A(R4)
new:    80010E11:  BRW     8000BF7D
PATCH>UPDATE
UPDATING IMAGE FILE "DBA2:[RL1PATCH]SYS.EXE;6"
PATCH>EXIT


  PATCH BASE LEVEL X5.04  15 DEC 77

IMAGE FILE BEING PATCHED:       "DBA2:[RL1PATCH]SYS.EXE;6"
JOURNAL FILE:                   "DBA2:[RL1PATCH]SYS.JNL;6"
DATE/TIME OF PATCH:             24-AUG-1978 09:27:59.50

%PATCH-I-NOLCL, image does not contain local symbols
%PATCH-I-NOGBL, some or all global symbols not accessible
PATCH>DEF TTYSUB=800022EC
symbol "TTYSUB" defined as 800022EC
PATCH>SET PATCH 80004600
PATCH>REPLACE/I TTYSUB+64
OLD>  'MOVQ R6,-(SP)'
OLD>  EXIT
NEW>  'PUSHR #00F0'
NEW>  EXIT
old:    80002350:  MOVQ    R6,-(SP)
new:    80002350:  BRW     PAB
new:    PAB:  PUSHR   #00F0
new:    8000461A:  BRW     80002353
PATCH>REPLACE/I TTYSUB+0EB
OLD>  'MOVQ (SP)+,R6'
OLD>  EXIT
NEW>  'POPR #00F0'
NEW>  EXIT
old:    800023D7:  MOVQ    (SP)+,R6
new:    800023D7:  BRW     8000461D
new:    8000461D:  POPR    #00F0
new:    80004621:  BRW     800023DA
PATCH>UPDATE
UPDATING IMAGE FILE "DBA2:[RL1PATCH]SYS.EXE;7"
PATCH>EXIT


  PATCH BASE LEVEL X5.04  15 DEC 77
```

```
SYS.JNL;6                          28-AUG-1978 09:18:44.25          Page 3

IMAGE FILE BEING PATCHED:          "DBA2:[RL1PATCH]SYS.EXE;7"
JOURNAL FILE:                      "DBA2:[RL1PATCH]SYS.JNL;6"
DATE/TIME OF PATCH:                24-AUG-1978 18:57:41.44

%PATCH-I-NOLCL, image does not contain local symbols
%PATCH-I-NOGBL, some or all global symbols not accessible
PATCH>SET PATCH 80004600
PATCH>DEF SETPRI=8000937D
symbol "SETPRI" defined as 8000937D
PATCH>INSERT/I SETPRI+36
OLD>  'MOVB R3,B^33(R4)'
NEW>  'MOVB R3,B^0B(R4)'
NEW>  EXIT
old:    80009383:  MOVB    R3,B^33(R4)
new:    80009383:  BRW     PAB
new:    80009386:  NOP
new:    PAB:  MOVB    R3,B^33(R4)
new:    80004628:  MOVB    R3,B^0B(R4)
new:    8000462C:  BRW     80009387
PATCH>UPDATE
UPDATING IMAGE FILE "DBA2:[RL1PATCH]SYS.EXE;8"
PATCH>EXIT


  PATCH BASE LEVEL X5.04  15 DEC 77

IMAGE FILE BEING PATCHED:          "DBA2:[RL1PATCH]SYS.EXE;8"
JOURNAL FILE:                      "DBA2:[RL1PATCH]SYS.JNL;6"
DATE/TIME OF PATCH:                25-AUG-1978 10:42:40.49

%PATCH-I-NOLCL, image does not contain local symbols
%PATCH-I-NOGBL, some or all global symbols not accessible
PATCH>SET PATCH 80004600
PATCH>DEF MBDRV=8000127C
symbol "MBDRV" defined as 8000127C
PATCH>EXAMINE/I MBDRV+0EF
8000136B:  BRW     80004608
PATCH>EXAMINE/I MBDRV+169
800013E5:  BRW     8000460F
PATCH>EXAMINE/I MBDRV+155
800013D1:  MOVAB   B^0C(R5),R7
PATCH>INSERT/I MBDRV+155
OLD>  'MOVAB B^0C(R5),R7'
NEW>  'MOVL (SP),R4'
NEW>  EXIT
old:    800013D1:  MOVAB   B^0C(R5),R7
new:    800013D1:  BRW     PAB
new:    800013D4:  NOP
new:    PAB:  MOVAB   B^0C(R5),R7
new:    80004633:  MOVL    (SP),R4
new:    80004636:  BRW     800013D5
PATCH>EXAMINE 80003B4C
80003B4C:  32392E30
PATCH>DEPOSIT 80003B4C=30302E31
old:    80003B4C:  32392E30
new:    80003B4C:  30302E31
```

```
SYS.JNL;6                          28-AUG-1978 09:18:44.25          Page 4

PATCH>EXAMINE/AS .
80003B4C:  '1.00'
PATCH>UPDATE
UPDATING IMAGE FILE "DBA2:[RL1PATCH]SYS.EXE;9"
PATCH>EXIT


  PATCH BASE LEVEL X5.04  15 DEC 77

IMAGE FILE BEING PATCHED:          "DBA2:[RL1PATCH]SYS.EXE;9"
JOURNAL FILE:                      "DBA2:[RL1PATCH]SYS.JNL;6"
DATE/TIME OF PATCH:                25-AUG-1978 17:56:20.02

%PATCH-I-NOLCL, image does not contain local symbols
%PATCH-I-NOGBL, some or all global symbols not accessible
PATCH>EXAMINE 800126CC+15
800126E1:  32392E30
PATCH>DEPOSIT 800126CC+15=30302E31
old:    800126E1:  32392E30
new:    800126E1:  30302E31
PATCH>EXAMINE/AS .
800126E1:  '1.00'
PATCH>UPDATE
UPDATING IMAGE FILE "DBA2:[RL1PATCH]SYS.EXE;10"
PATCH>EXIT
```

```
  PATCH BASE LEVEL X5.04  15 DEC 77

IMAGE FILE BEING PATCHED:       "DBA2:[RL1PATCH]TMDRIVER.EXE;2"
JOURNAL FILE:                   "DBA2:[RL1PATCH]TMDRIVER.JNL;1"
DATE/TIME OF PATCH:             24-AUG-1978 15:31:27.25

%PATCH-I-NOLCL, image does not contain local symbols
%PATCH-I-NOGBL, some or all global symbols not accessible
PATCH>REPLACE/I 80000060+0EB9
OLD>  'BITL #00000080,B^04(R4)'
OLD>  EXIT
NEW>  'BRB 80000060+0EC5'
NEW>  EXIT
old:    80000F19:  BITL    #00000080,B^04(R4)
new:    80000F19:  BRB     80000F25
new:    80000F1B:  NOP
new:    80000F1C:  NOP
new:    80000F1D:  NOP
new:    80000F1E:  NOP
new:    80000F1F:  NOP
new:    80000F20:  NOP
PATCH>UPDATE
UPDATING IMAGE FILE "DBA2:[RL1PATCH]TMDRIVER.EXE;3"
PATCH>EXIT
```

```
  PATCH BASE LEVEL X5.04  15 DEC 77

IMAGE FILE BEING PATCHED:       "DBA2:[RL1PATCH]UETDISK00.EXE;2"
JOURNAL FILE:                   "DBA2:[RL1PATCH]UETDISK00.JNL;1"
DATE/TIME OF PATCH:             25-AUG-1978 15:14:44.44

%PATCH-I-NOLCL, image does not contain local symbols
%PATCH-I-NOGBL, some or all global symbols not accessible
PATCH>SET ECO 1
PATCH>SET MODE BYTE
PATCH>REPL 5999
OLD>  03
OLD>  EXIT
NEW>  04
NEW>  EXIT
old:    00005999:  03
new:    00005999:  04
PATCH>UPDATE
UPDATING IMAGE FILE "DBA2:[RL1PATCH]UETDISK00.EXE;3"
PATCH>EXIT
```

```
  PATCH BASE LEVEL X5.04  15 DEC 77

IMAGE FILE BEING PATCHED:       "DBA2:[RL1PATCH]XMDRIVER.EXE;2"
JOURNAL FILE:                   "DBA2:[RL1PATCH]XMDRIVER.JNL;1"
DATE/TIME OF PATCH:             25-AUG-1978 14:16:08.60

%PATCH-I-NOLCL, image does not contain local symbols
%PATCH-I-NOGBL, some or all global symbols not accessible
PATCH>EXAMINE 80000000
80000000:  00000000
PATCH>E
80000004:  00000000
PATCH>E
80000008:  001E0B5F
PATCH>DEPOSIT/WORD 80000008=0BF0
old:    80000008:  0B5F
new:    80000008:  0BF0
PATCH>EXAMINE 80000B60
80000B60:  00000000
PATCH>DEPOSIT 80000B60=90-8
old:    80000B60:  00000000
new:    80000B60:  00000088
PATCH>DEPOSIT 80000B64=80000B68
old:    80000B64:  00000000
new:    80000B64:  80000B68
PATCH>SET PATCH 80000B60
PATCH>DEFINE XMDRV=8000005C
symbol "XMDRV" defined as 8000005C
PATCH>EXAMINE/I XMDRV+640
8000069C:  BISW2   #2,(R4)
PATCH>INSERT/I XMDRV+640
OLD>  'BISW2 R2,(R4)'
NEW>  'BISW2 R2,(R4)'
NEW>  EXIT
old:    8000069C:  BISW2   R2,(R4)
new:    8000069C:  BRW     PAB
new:    PAB:  BISW2   R2,(R4)
new:    80000B6B:  BISW2   R2,(R4)
new:    80000B6E:  BRW     8000069F
PATCH>E/I .
80000B6E:  BRW     8000069F
PATCH>E/I \
8000069F:  BBSC    #02,8^5A(R5),800006A8
PATCH>UPDATE
UPDATING IMAGE FILE "DBA2:[RL1PATCH]XMDRIVER.EXE;3"
PATCH>EXIT
```

APPENDIX C

SUMMARY OF UPDATES RELEASED WITH VERSION 1.01 OF VAX/VMS

PATCH

ECO02 KDM0008 16-OCT-1978
Print updating file message to SYS$ERROR.

ECO03 KDM0006 16-OCT-1978
Allow execution of next ECO level if current ECO level was already performed.

ECO04 KDM0009 30-OCT-1978
Add argument count to correct error message for duplicate CREATE commands.

8USER.PAR

ECO01 RIH0004 17-OCT-1978
Set correct number of global sections for 8 user system.

BCK - RMS BACKUP

ECO01 SPRXXXXX 02-NOV-1978
Fixes problem with restoration of files that were copied from magtape.

COPY utility

ECO01 CHP19547 17-Oct-1978
Removes version number stickiness in APPEND.

ECO02 JAK0001 01-Sep-1978
To support copy of relative files over the network, set BRO bit in output FAB if NET bit is set.

ECO03 JAK0002 01-Sep-1978
To support copy of files over the network, get ALQ and DEQ values from input XABALL if NET bit is set.

ECO04 JAK0003 01-Sep-1978
To support copy of files in VFC format over the network, put RHB address in both input and output FABs if NET bit is set.

ECO05 JAK0004 06-Sep-1978
To support file append over the network, omit 'incompatible attributes' check if NET bit is set.

ECO06 CHP20339 25-Oct-1978
Corrects error message displayed when user specifies an invalid file specification.

SUMMARY OF UPDATES RELEASED WITH VERSION 1.01 OF VAX/VMS

DEBUG

ECØØ1 DAR19460 17-OCT-1978
Fixes reserved operand fault if user types ^Y DEBUG while the debugger is handling an access violation because the user is watching a location on that page of memory.

DELETE utility

ECØØ1 CHP20437 25-Oct-1978
Defines YESTERDAY to be 24-hours before Ø-time today.

DMDRIVER

ECØØ1 LMKØØØ1 3-OCT-1978
Fixes occasional Operation Incomplete errors.
ECØØ2 LMKØØØ2 3-OCT-1978
Fixes IPL problem during timeouts and power failures.
ECØØ3 LMKØØØ3 7-NOV-1978
Adds UBA XMIT error clearing.

F11AACP

ECØØ1 ACGØØØ2 13-OCT-1978
Do FCB cleanup only if a file header has been read. This prevents a file system crash if a disk write error occurrs under obscure circumstances.

ECØØ2 ACGØØØ5 3-NOV-1978
Return the AQB to the system pool when shutting down the ACP. The space was being lost.

F11BACP

ECØØ1 ACGØØØ2 13-OCT-1978
Do FCB cleanup only if a file header has been read. This prevents a file system crash if a disk write error occurrs under obscure circumstances.

ECØØ2 ACGØØØ5 3-NOV-1978
Return the AQB to the system pool when shutting down the ACP. The space was being lost.

SUMMARY OF UPDATES RELEASED WITH VERSION 1.01 OF VAX/VMS

INSTALL

ECO01 KDM0001 03-NOV-1978
Print error message if cannot install all global sections.

JOBCTL - VMS Job controller

ECO01 WHB0001 1-OCT-1978
Improves system batch and print queue update strategy to provide for better recovery after shut-down or crash.

LPDRIVER

ECO01 LMK0001 24-OCT-1978
Fixes number of lines QIO returns which also fixes the page accounting for the Print Symbiont.

MINIMUM.PAR

ECO01 RIH0006 31-OCT-1978
Set correct number of required SPT entries (SPTREQ) for minimum system. Original value is not sufficient for large memory systems (>1.25Mb).

RMS

ECO01 JAK0001 28-AUG-1978
Miscellaneous clean-up prior to DECnet V1.0 code freeze.
ECO02 JAK0002 31-AUG-1978
Fix failure to return DEQ value to FAB on CREATE.
ECO03 RAN0001 31-AUG-1978
Eliminate wait for I/O on mailbox.

RSXSHR

ECO01 TMH0001 5-OCT-1978
Fix $RUN directive for names >4 characters.

SDA

Install System Dump Analyzer (SDA)
(Print SDA.HLP for partial documentation)

SUMMARY OF UPDATES RELEASED WITH VERSION 1.01 OF VAX/VMS

SYS

ECO#1 RIH0001 20-SEP-1978
Fixes multiple UNIBUS error.

ECO#2 TMH0001 5-OCT-1978
Fixes SWAPPER map allocation which fixes negative reference count bugcheck.

ECO#3 LMK0001 5-OCT-1978
Fixes SETPRI problem when setting priority of other compute state process.

ECO#4 RIH0002 17-OCT-1978
Fix failure to decrement quota when releasing common event flag blocks.

ECO#5 RIH0003 17-OCT-1978
Correct failure to re-awaken from mailbox resource wait.

ECO#6 SGD0001 25-OCT-1978
Fix bug in NETDRIVER when user exceeds quota.

ECO#7 TGD0001 26-OCT-1978
Fixes TTYDRV, no longer echos a null after reading terminator during read no-echo.

ECO#8 KDM0001 31-OCT-1978
Fix SYSCRMPSC bug, prohibiting RUN MT: or run TT: .

ECO#9 TGD0002 30-OCT-1978
FIXES TTYDRV SO THAT /HOSTSYNC WORKS

ECO10 LMK0001 01-NOV-1978
Fixes generic allocation of spooled devices.

ECO11 RIH0004 23-OCT-1978
Change system version number to 1.01

UETP (VMS test package)

UETDISK00

ECO#1 RAB0001 22-AUG-78
Fix for RM03 disk included in Release V1.0

ECO#2 RAB0002 06-NOV-78
Fix RK07 system disk full error.

VMSRTL

ECO#1 SRL00001 30-OCT-1978
Correct default digits_in_fraction parameter in FOR$CNV_IN_DEFG to agree with RTL reference manual, as noted in V01 Release Notes.

ECO#2 SRL20364 30-OCT-1978
Correct OTS$POWCJ so that negative powers of imaginary values work correctly.

XMDRIVER

ECO01 ADE001 1-NOV-1978
Fixes a race condition which cause a receive request to get lost at 1 Mbaud.

ECO02 ADE001 1-NOV-1978
Fixes a failure to check the return status of a request for a buffer allocation. This could cause the system to crash if memory is scarce and the user does not have resource wait mode enabled.

ECO03 ADE003 1-NOV-1978
Causes the maximum xmit buffer size ot be 4095 bytes which is the DMC limit. It had been tied to the size of the preallocated receive buffers established at device initialization.

APPENDIX D

SUMMARY OF UPDATES RELEASED WITH VERSION 1.5 OF VAX/VMS

COPY

ECO7 TMH 02-Jan-1979
Fix copying ISAM file after any non-ISAM file.
(Error message: No primary key defined)

CRDRIVER

ECO01 LMK0001 28-DEC-1978
Enhance the hardware error handling.

ECO02 LMK0002 02-MAR-1979
Fix unsolicited interrupt handler so that Input Symbiont always gets started properly.

DBDRIVER

ECO01 LMK0001 15-JAN-1978
ADD DR DEVICE TYPES TO DRIVE TYPE DESCRIPTOR TABLE
TO FIX POWERFAIL PROBLEMS.

DELETE

ECO2 TMH 04-Jan-1979
Fix PURGE defaulting when device/directory is specified.
e.g. PURGE [USER] defaults to PURGE [USER]*.*

DRDRIVER

ECO01 LMK0001 28-DEC-1978
Add IO$M_DIAGNOSTIC and IO$M_MOVETRACKD subfunctions.

ECO02 LMK0002 22-JAN-1979
ADD RP DEVICE TYPES TO DEVICE DESCRIPTOR TABLE
TO FIX POWERFAIL PROBLEMS.

F11ACP

ECO003 ACG0017 19-Jan-1979
Limit window size value to 80 to avoid overflowing window buffer on the stack, which causes a system crash.

ECO004 ACG0023 7-Feb-1979
Fix maximum attribute code parameter to enable use of DIRSEQ attribute.

INIT

EC0001 ACG0016 18-Jan-1979
Correct code to build format 3 map pointers. This bug caused a corrupt volume to result from /HEADERS values greater than 16000.

EC0002 ACG21786 19-Jan-1979

Limit home block search to 10 tries to prevent, for example, initing a floppy from taking forever.

LPA11

Install LPA-11 driver and run-time package

LPDRIVER - LINE PRINTER DRIVER

EC002 LMK0001 19-JAN-1978
Fix repeating characters when printer is turned off-line.

RSXSHR

EC02 TMH 22-DEC-1978
Add placement control list processing to IO.EXT

EC03 TMH 14-FEB-1979
Fix AST delivery which was causing BCK/RST to hang.

EC04 TMH 1-Mar-1979
Fix problem with returning directory descriptor in FCS-11 when the length to be returned is zero.

EC05 TMH 12-Mar-1979
Fix floating point exception AST delivery.

SET

EC01 TMH 06-NOV-1978
Fix /VT52,/VT05,etc. to set page size.
Restrict WIDTH to 255 maximum.
EC02 TMH 15-DEC-1978
Fix SET ACCOUNTING/ENABLE=keyword to work.

SOS

EC01 TMH 15-Jan-1979
Fixes improper initialization of page/line before entering the first line of a new file.
EC02 TMH 15-Jan-1979
Fixes problems with multiple % in match strings.

STARTUP

Insert system-wide logical names for COBOL

SYS

EC012 KDM0002 21-NOV-1978
Remove SCANDEADPT check of SHRCNT.

ECO13 LMK0001 18-DEC-1978
Clear DEVSM_RCK, DEVSM_WCK, and DEVSM_SWL when dismounting a foreign mounted device.

ECO14 TGD001 26-DEC-1978
Remove faulty escape sequence from terminal driver escape sequence table.

ECO15 TGD002 26-DEC-1978
Set 2 stop bits for terminals at 150 baud or less.

ECO16 ACG0003 29-Dec-1978
Add handling of volume set mounted volume list entries to the system DISMOUNT routine.

ECO17 RIH0005 04-Jan-1979
Correct initialization of paged pool size by SWAPPER.

ECO18 RIH21586 10-Jan-1979
Fix incorrect probing of 0 length buffers by IOCIOPOST.

ECO20 TMH 06-Feb-1979
Fix occasional BUGCHECK failure to write dump file.

ECO21 SGD 13-Feb-1979
Permit transmission of zero-length DECnet messages.

ECO22 RIH22252 15-FEB-1979
Prevent MFYNULPGFL bugcheck caused by intense swapping/paging

ECO23 KDM22235 19-FEB-1979
Fix purge working set scan to allow deletion of global writable pages.

ECO24 KDM0004 19-FEB-1979
Fix bug that had global slave PTE converted to valid form instead of transition form in FREWSLE.

ECO25 RIH0007 23-FEB-1979
Enable outswapper to handle PFNLOCK. Remove bugcheck.

ECO26 TGD 28-FEB-1979
Fix terminal driver so terminal width must be positive.

ECO27 TGD 28-FEB-1979
Fix ASSIGN so when second attempt is made to associate mailbox with device an error is returned.

ECO28 TGD 21-MAR-1979
DONT LOSE LF DURING MULTIPLE CR'S AT COLUMN 0

VMSLIB

Replace macro definitions in STARLET.MLB for:

LPA-11 support	$SSDEF,$LADEF,$IODEF,$DCDEF.
Multi-volume support	$FIBDEF,$FIDDEF.
ISAM	$RMSDEF,$XABKEY,$XABSUMDEF,$XABKEYDEF.
RTL	$LIBDEF.
System	$CNTREG_G

Replace macro definitions in LIB.MLB for:

LPA-11 support	$SSDEF,$LADEF,$IODEF,$DCDEF.
Multi-volume support	$FIBDEF,$FIDDEF,$VCBDEF,$RVTDEF,$RVXDEF, $FCBDEF,$MTLDEF,$FM2DEF.
ISAM	$RMSDEF,$XABKEY,$XABSUMDEF,$XABKEYDEF.
DCL	$CLIDEFQUAL<DISM,MACR,MOUN,SORT,COBO>.

Replace object modules in STARLET.OLB for:

DCL	Modules CLIGBL0,CLIGBL1 holding global symbols for $CLIDEFQUAL<DISM,MACR,MOUN, SORT,COBO>.
LPA-11	Module SYSVECTOR holding global symbols for $IODEF, $SSDEF. New modules LASWEEP, LABUFFER,LASNDLDRQ.
SORT32 run-time	New modules RMSIO,SORTSR,SCRIO,KEYSUB.
VMSRTL	ECO's to modules FCLOSE,FORCNVIR, FUDFRL,LIBMSGDEF,FORCNVOI, FORCNVII,FUDVRL,LVM,LIBEF,FOPEN.

VMSRTL

ECO03 SBL20340 07-DEC-1978
Fix so that FORTRAN users can set FAB$V_SCF with USEROPEN.

ECO04 SBL20988 07-DEC-1978
Fix overflow bug in FOR$CNV_IN_DEFG.

ECO05 SBL0005 07-DEC-1978
Fix overflow bug in FORCNV_IN_I, FORCNV_IN_L, FORCNV_IN_O, FORCNV_IN_Z.

ECO06 SBL0006 07-DEC-1978
Supply reentrant versions of LIB$GET_VM and LIB$FREE_VM.

ECO07 SBL0007 07-DEC-1978
Make sequential I/O a bit faster.

ECO08 SBL0008 11-DEC-1978
Supply new routines LIBGET_EF, LIBFREE_EF and LIB$RESERVE_EF.

ECO09 SBL0009 08-JAN-1979
Fix overflow bug in FOR$CNV_OUT_I.

ECO010 SBL21789 18-JAN-1979
Fix problem with omitted values in FORTRAN list directed input.

XMDRIVER

ECO01 ADE001 1-NOV-1978
Fixes a race condition which cause a receive request to get lost at 1 Mbaud.

ECO02 ADE001 1-NOV-1978
Fixes a failure to check the return status of a request for a buffer allocation. This could cause the system

to crash if memory is scarce and the user does not have resource wait mode enabled.

ECO03 ADE003 1-NOV-1978
Causes the maximum xmit buffer size ot be 4095 bytes which is the DMC limit. It had been tied to the size of the preallocated receive buffers established at device initialization.

ECO04 ADE004 16-NOV-78
Disallow receives at FDT time unless the device is active. This is because the receives have a separate listhead in the UCB which may contain garbage. A subsequent start of the device will initialize this listhead.

ECO05 ADE005 16-NOV-78
Enter IO$_WRITEVBLK into the dispatch table.

ECO06 ADE006 19-DEC-78 15:00
The buffer address specified by the DMC to identify the receive buffer being returned may be invalid in bits 17 or 16. Use buffers aligned on even longwords for the first receive control block, and aligned on odd longwords for the second control block. Thus the proper block can be indentified by checking bit 2 of the returned receive buffer address.

ECO07 ADE007 22-JAN-79 14:00
Modify the cancel I/O routine to flush all attention ASTs associated with the process and channel issuing the cancel.

ECO08 ADE008 9-MAR-79 13:30
Modify data path allocation for receives.

ECO09 ADE009 9-MAR-79 16:30
Prevent fatal errors from going unnotified.

BACKTRANS

ECO1 Updated to support COBOL/C74 and COBOL/RSX.

DEBUG

ECO2 Install new DEBUG for COBOL,BLISS support

MACRO32

Install native-mode assembler

MSGFIL

Install new System Message file (SYSMSG.MPF)

Multi-volume support

The following 6 images are updated to support multi-volume disk devices:

ECO1 DSC2
ECO1 VFY2

```
        ECO3    F11BACP
        ECO1    SYSINIT
        ECO1    VMOUNT
        ECO1    DISMOUNT


RMS

        ECO4    Install ISAM and file sharing support


TRACE

        ECO1    Install new TRACE for COBOL,BLISS support


SHUTDOWN

        Fixes problem where a process with the string ACP in its
        name would not be stopped.


SORT32

        Install Sort-32


SYSUPD

        Copy new versions of:

                VMSKITBLD.COM - Updated list of VMS images
                VMSKITCPY.COM - Updated list of VMS images
                VMSUPDATE.COM - Correct wording of instructions
                LINEPAGE.COM  - Fixes patch for MACRO32 image


UETP (VMS test package)

        WARNING: This section of the update could not be placed
                 on a single floppy disk.  Therefore, this is the
                 first part of update -- the second part is on the
                 next floppy.

        ECO16   TLC0016         11-DEC-78
                New control script for VMS system services
                        testing to avoid redundant tests.

        ECO18   RAB0018         13-DEC-78
                Install new VAX-11 SORT test


UETP    (VMS test package)

        This is the second part of the UETP update.

        ECO15   RAB0015         07-NOV-78
                Delete obsolete files (only if purging option specified)

        ECO17   RAB0017         13-DEC-78
                Install new RSX-11M Executive Directives test
```

ECO19 RAB0019 13-DEC-78
Install revised load test scripts, tape and disk tests, RMS test, COBOL, ISAM test, begin and end messages, error handling, and control scripts for the above.

DCL

ECO1 Install new DCL for MACRO, SORT, COBOL, multi-volume files (MOUNT and DISMOUNT), plus assorted bug fixes.

MTAAACP

ECO1 New version containing various bug fixes, including the capability of handling I/O Cancel on tape searches.

SYE

New version of SYE with support for additional devices.

Sample drivers

Install 3 sample drivers on [SYSEXE]. The drivers are:

ADDRIVER.MAR - AD11-K driver
DADRIVER.MAR - DA11-B driver
TDRIVER.MAR - template driver

HELP

Install new SORT,BLISS,COBOL help files

APPENDIX E

INSTALLING AND USING THE SYSTEM DUMP ANALYZER

The System Dump Analyzer (SDA) is an unsupported utility that lets you inspect the contents of the data structures and physical memory at the time of a system crash, and with this information, helps you determine the cause of the crash.

You can use SDA interactively or you can have the output sent to a listing file. The following list summarizes the operations you can perform with SDA:

- print a list of all processes at the time of the crash
- display information about any given process
- display the system page table
- display the nonpaged dynamic storage pool
- display saved hardware context
- display any process stack or interrupt stack
- display the value of a system symbol and the contents at that location
- examine memory for any process

This appendix contains information describing the installation and use of SDA.

INSTALLATION REQUIREMENTS

When the system crashes, the kernel routine writes the current state of the error log buffer, processor registers, and physical memory to a predefined contiguous file named [SYSEXE]SYSDUMP.DMP. However, you may need to recalculate the size of this file to accommodate all the crash dump information. Calculate the correct size of this file using the following formula:

```
blocks = physical-memory-size-in-pages + 4
```

After determining the new file size, you invoke the [SYSUPD]SWAPFILES.COM file to change the length of the file. Figure E-1 is an example that steps you through the procedure for changing the file size for a 2 megabyte system.

```
$ @[sysupd]swapfiles
          .
          .
          .
To leave a file size at its current value type a
carriage return <CR> in response to its file size.
Current file sizes are:


DIRECTORY DB0:[SYSEXE]
8-NOV-78 10:07

PAGEFILE.SYS;2       48000.  C  22-AUG-78 11:24
SWAPFILE.SYS;2       36000.  C  22-AUG-78 11:24
SYSDUMP.DMP;2        400.    C  22-AUG-78 11:24

TOTAL OF 88104./88121.BLOCKS IN 3.  FILES

Enter new size for paging file:  <CR>
Enter new size for swapping file:  <CR>
Enter new size for system dump file:  4104

$
```

Figure E-1

When SWAPFILES.COM displays the files contained in the [SYSEXE] directory, it is the second column of the directory listing that indicates the size of each file. SWAPFILES.COM then prompts for a new file size for the paging file, swapping file, and system dump file. You supply the new file size to the prompt associated with the system dump file.

Keep in mind, however, that the new file size does not take effect until the next system start-up.

It is also a good idea to ensure that a dump listing is printed for every system crash. There are several possible ways to accomplish this; however, the most practical way is to arrange the system start-up file in such a way that it invokes SDA when the system is booted. When invoked by the start-up procedure, SDA will execute the given commands only if the system has just crashed. The following example shows the commands that could be added to the system start-up file to print an analyzer listing after a crash occurs. DB0 is assumed to be the system disk.

```
$ !
$ !     Print dump listing if we just crashed
$ !
$ RUN SYS$SYSTEM:SDA
DB0:
SET OUTPUT LPA0:SYSDUMP.LIS !Make listing file
SHOW CRASH                  !Display crash information
SHOW STACK                  !Show current operating stack
SHOW SUMMARY                !List all active processes
SHOW PROCESS/PCB/PHD/REG    !Display current process
SHOW SYMBOLS/ALL            !Print system symbol table
SHOW POOL                   !Dump entire nonpage pool
EXAMINE/SYSTEM              !Dump writable system region
EXIT
```

You can invoke SDA for an interactive session if you need further information.

You should save the SYSDUMP.DMP file after a crash so that if the system crashes again, the dump will not be overwritten. The saved dump may be inspected by typing its file specification at the initial SDA prompt.

USING SDA

The SDA utility is executed via directives obtained from SYS$INPUT (terminal or procedure data). The output can be sent to a listing file or it can be examined at your terminal via interactive use. You invoke SDA by typing:

```
$ RUN SYS$SYSTEM:SDA
```

Control is then passed to SDA, and SDA, in turn, displays the following prompt at your terminal:

```
Enter the device containing the dump>
```

You respond by typing the name of the device that contains the dump file or a file specification if the dump was saved under another name. Usually, this device is the system disk. SDA then builds the system symbol table and upon completion issues the following prompt to indicate that it is ready to accept commands:

```
SDA>
```

SDA COMMANDS

SDA commands let you inspect the contents of the dump file in an organized manner. You can obtain help on any of the SDA commands by typing:

```
SDA> help <command-name>
```

```
The names of the commands are:

DEFINE     EXAMINE     EXIT     SET     SHOW
```

To receive information pertaining to the installation and use of SDA, you type:

```
SDA> help sda
```

You enter SDA commands in the following general format:

```
command[/qualifier]  [parameter]     [!comment]
```

command
: Specifies the name of an SDA command that tells SDA what you want it to do. SDA commands can optionally contain qualifiers and parameters.

/qualifier
: Specifies a command qualifier that modifies the action of the SDA command or supplies additional information needed for SDA processing. Multiple qualifiers are allowed to follow a single command; however, each qualifier must be preceded by the slash character (/).

parameter
: Specifies the target of the command, such as a range of memory locations.

!comment
: Specifies a comment. SDA ignores all characters starting with and including the exclamation character (!).

The following is a brief description of the SDA commands.

DEFINE symbol = value

Defines a temporary symbol with the specified value. The symbol can be used in expressions exactly like a system symbol. Each subsequent definition of the same symbol overrides any previous definitions.

EXAMINE location[:location]

Displays the contents of the specified memory location or range of locations. The location can be specified as an arithmetic expression using the following operators:

+	addition
-	subtraction
*	multiplication
/	division
.	displays the current address
@	a prefix operator that means "display the contents of the address"

The expression may contain system symbols or temporary symbols (see the DEFINE command).

In addition, you can dump the entire contents of a memory region by specifying one of the following qualifiers:

/P0	Process region
/P1	Control region
/SYSTEM	System region
/ALL	All of the above

EXIT

Exits the current display when typed at the screen overflow prompt, or exits the program if at normal prompt.

SET

This command is a nonprinting directive. The following types of SET commands are available:

```
SET PROCESS name [/INDEX=n]
SET OUTPUT filespec
```

SET PROCESS name [/INDEX=n]

Selects a process to be used as the current process for later commands.

SET OUTPUT filespec

Specifies that all remaining output should be sent to the given listing file (filespec).

SHOW

Displays formatted data structures and/or memory. The following types are available:

```
SHOW CRASH
SHOW DEVICE
SHOW PAGE_TABLE
SHOW PFN_DATA
SHOW POOL
SHOW PROCESS
SHOW STACK
SHOW SUMMARY
SHOW SYMBOL
```

SHOW CRASH

Displays the following general information about the crash:

- process name
- image name
- date and time of crash
- processor registers
- reason for BUGCHECK exception

SHOW DEVICE name

Displays the I/O data structures associated with a generic device name. The device name may be a generic device that shows all I/O structures associated with that device type (for example, DB), or, it may be a specific device name that shows information pertaining to only that device (for example, DBA1).

This command is mainly used for debugging user-written I/O drivers.

SHOW PAGE_TABLE

Displays a formatted listing of the system page table.

SHOW PFN_DATA

Displays a listing of the free, modified, and bad page lists, as well as the entire PFN data base.

SHOW POOL

Displays a formatted dump of the nonpaged dynamic storage pool. This command also attempts to identify each block by its block type. The following qualifiers can be appended to this command:

/IRP	prints only the IRP lookaside storage
/DYNAMIC	prints everything but the lookaside list
/ALL	prints everything; this is the default condition

SHOW PROCESS name [/INDEX=n]

Displays information about a particular process. Either the "name" or the "index" should be specified to indicate the desired process. You must specify the process index in hexadecimal representation. The following qualifiers can be appended to this command to determine what information SDA displays. If no qualifiers are specified, SDA displays the process control block.

/WORKING_SET_LIST	displays working set list
/PROCESS_SECTION_TABLE	displays process section table
/PAGE_TABLES	displays page tables
/REGISTERS	displays registers
/PCB	displays process control block
/PHD	displays process header
/ALL	displays all of the above

SHOW STACK

Displays a stack for the current process. You can specify one or more of the following qualifiers to display the corresponding stack. If no qualifiers are specified, SDA displays the current operating stack.

/INTERRUPT	System-wide interrupt stack
/KERNEL	Kernel mode stack
/EXECUTIVE	Executive mode stack
/SUPERVISOR	Supervisor mode stack
/USER	User mode stack
/ALL	All stacks

SHOW SUMMARY

Displays a list of all processes in the system at the time the sytem crashed.

SHOW SYMBOL symbol

Displays the value of the given system symbol and the contents of that memory location (if possible). The following qualifier can be appended to this command:

/ALL	prints all symbols and sorts them by name and by value

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Please cut along this line.

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code or Country_____

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

RT/C SOFTWARE PUBLICATIONS TW/A14
DIGITAL EQUIPMENT CORPORATION
1925 ANDOVER STREET
TEWKSBURY, MASSACHUSETTS 01876

Do Not Tear - Fold Here

Cut Along Dotted Line

digital

PRINTED IN USA