VMS Version 5.2 Release Notes

Order Number: AA–LB22B–TE

June 1989

This document describes changed software features, problems and restrictions to software, and changes to documentation for VMS Version 5.2.

Revision/Update Information:	This manual supersedes previous VMS Release Notes
Operating System and Version:	VMS Version 5.2
Software Version:	VMS Version 5.2

digital equipment corporation
maynard, massachusetts

June 1989

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

ZK5194

Production Note

This book was produced with the VAX DOCUMENT electronic publishing system, a software tool developed and sold by Digital. In this system, writers use an ASCII text editor to create source files containing text and English-like code; this code labels the structural elements of the document, such as chapters, paragraphs, and tables. The VAX DOCUMENT software, which runs on the VMS operating system, interprets the code to format the text, generate a table of contents and index, and paginate the entire document. Writers can print the document on the terminal or line printer, or they can use Digital-supported devices, such as the LN03 laser printer and PostScript printers (PrintServer 40 or LN03R ScriptPrinter), to produce a typeset-quality copy containing integrated graphics.

Contents

TABLES

Preface

This manual describes software enhancements and corrections in Version 5.2 of the VMS operating system.

These release notes supersede all release note documentation for previous versions of the VMS operating system. They include, or update, any previous release notes that are still pertinent to the Version 5.2 release. The source for each note is indicated by a marginal label designating the version number of the original release, for example, "V5.0."

For information about the new features included in VMS Version 5.2, see the *VMS Version 5.2 New Features Manual*.

Intended Audience

This manual is intended for all system users. Read this manual before you install, upgrade, or use VMS Version 5.2.

Document Structure

This manual contains the following chapters:

- Chapter 1 contains information about license management.
- Chapter 2 contains release notes intended for general users of the VMS operating system and VMS DECwindows.
- Chapter 3 contains release notes intended for system managers.
- Chapter 4 contains release notes intended for programmers.
- Chapter 5 contains additions and corrections to the VMS documentation set.

Conventions

The following conventions are used in this manual:

mouse	The term *mouse* is used to refer to any pointing device, such as a mouse, a puck, or a stylus.
MB1, MB2, MB3	MB1 indicates the left mouse button, MB2 indicates the middle mouse button, and MB3 indicates the right mouse button. (The buttons can be redefined by the user.)
Ctrl/X	A sequence such as Ctrl/X indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.

PF1 x	A sequence such as PF1 x indicates that you must first press and release the key labeled PF1, then press and release another key or a pointing device button.
V5.x	A label such as V5.x in the margin indicates the number of the release of VMS in which the corresponding information first appeared. V5.x indicates VMS Version 5.x.
Return	A key name is shown enclosed to indicate that you press a key on the keyboard.
. . .	In examples, a horizontal ellipsis indicates one of the following possibilities: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
( )	In format descriptions, parentheses indicate that, if you choose more than one option, you must enclose the choices in parentheses.
[ ]	In format descriptions, brackets indicate that whatever is enclosed is optional; you can select none, one, or all of the choices.
{}	In format descriptions, braces surround a required choice of options; you must choose one of the options listed.
	Red ink indicates information that you must enter from the keyboard or a screen object that you must choose or click on. For online versions, user input is shown in **bold**.
boldface text	Boldface text represents the introduction of a new term or the name of an argument, an attribute, or a reason.
italic text	Italic text represents information that can vary in system messages (for example, Internal error *number*).

UPPERCASE TEXT	Uppercase letters indicate that you must enter a command (for example, enter OPEN/READ).
	Uppercase letters indicate the name of a routine, the name of a file, the name of a file protection code, or the abbreviation for a system privilege.
-	Hyphens in coding examples indicate that additional arguments to the request are provided on the line that follows.
numbers	Unless otherwise noted, all numbers in the text are assumed to be decimal. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

1 License Management

V5.2

The following sections include information to supplement the *VMS License Management Utility Manual* provided with VMS Version 5.2. Although most of the information in this chapter is for managing VMS and System Integrated Product licenses, some of the information provided pertains to managing layered product licenses.

Note: In VMS Version 5.2 you *must* register your VAXcluster license before you can use your VAXcluster software. VMS VAXclusters users, see Section 1.8 for important new information.

1.1 Registering Your Licenses

V5.2

Before VMS Version 5.0, you were required to install keys for certain products. For example, if you had a MicroVAX computer, you had to install the VMS multiuser key. If you had any System Integrated Products (SIPs) such as DECnet or VAX Volume Shadowing, you also used keys that were shipped on separate distribution media. Since VMS Version 5.0, these products and many other products are enabled only when you register their license information in the LICENSE database and activate it. Products may limit use or produce error messages if you fail to register and activate a license.

After you install the VMS operating system, you must register a VMS license. A VMS license lets you use the VMS operating system. You must also register the licenses for any of the following system integrated products (SIPs) you have purchased:

- VAXclusters
- DECnet–VAX
- VAX RMS Journaling
- VAX Volume Shadowing

In addition to VMS and the SIPs, many layered products that run on VMS Version 5.2 also require license registration. See the *VMS License Management Utility Manual* for a full explanation of license registration. This section provides an overview of the license registration procedure and offers a step-by-step review of the VMSLICENSE command procedure.

To register a license, you need to obtain a Product Authorization Key (PAK). A typical PAK is a piece of paper, provided by Digital, that includes the appropriate information to authorize access to software on a VAX computer or VAXcluster environment. Obtain a PAK from a Digital representative just as you obtain software. Your PAK should resemble the one shown in Example 1–1.

Example 1–1 A Sample Product Authorization Key (PAK)

```
 - - - - - - -
| | | | | | | |                                         _____________________
|d|i|g|i|t|a|l|  PRODUCT AUTHORIZATION KEY (PAK)      | DOCUMENT ISSUE DATE |
| | | | | | | |                                       |    10-MAY-1989      |
 - - - - - - -                                        |_____________________|
Digital Equipment Corporation
Maynard, Massachusetts
 ----------------------------------
| LICENSE ADMINISTRATION LOCATION:  |  ORDERED BY: Vacuum Module Systems INC.
|                                   |              Mr. S. A. Manteno
| Digital Equipment Corporation     |              32 Times Blvd.
| Maynard, Massachusetts            |              Vera City, Connecticut
|                                   |              12321
|                                   |
|                                   |
 ----------------------------------
*************************************************************************
 PAK ID:
                       Issuer: DEC
         Authorization Number: USA1956

 PRODUCT ID:
                 Product Name: VAX-VMS
                     Producer: DEC

 NUMBER OF UNITS:
              Number of Units: 400

 KEY LEVEL:
                      Version: 5.2
         Product Release Date:

 KEY TERMINATION DATE:
         Key Termination Date: 23-OCT-1989

 RATING:
      Availability Table Code: A
          Activity Table Code:

 MISCELLANEOUS:
                  Key Options: MOD_UNITS,NO_SHARE
                Product Token:
                  Hardware-Id:
                     Checksum: 1-COOD-AHGO-NEFI-CHIN
*************************************************************************
```

If you have a service contract with Digital, a VMS license may be registered for you during the VMS installation or upgrade procedure. For details on this kind of VMS license registration, see Section 1.3.

Register your licenses in the following order:

1 Register the VMS license for the VAX computer on which you have just installed the VMS operating system. If you have a VAXcluster environment, you then register a VMS license for each additional VAX computer in the cluster. Each registered VMS license is assigned to one of the nodes for the cluster.

2 Register the licenses for the System Integrated Products (SIPs) that you purchased.

Section 1.2 describes how to respond to the prompts of the command procedure SYS$UPDATE:VMSLICENSE.COM. You can use this procedure to register a license for any DIGITAL product that supplies a PAK. You can also register licenses with the LICENSE REGISTER command. See the *VMS License Management Utility Manual* for examples of license registration, using VMSLICENSE.COM and LICENSE REGISTER commands. This manual, which is a part of the VMS Version 5.2 Base Documentation Set, provides details about all of the LICENSE commands, the error messages, and recovery procedures for licensing tasks.

1.2 Registering a License Using the Command Procedure VMSLICENSE.COM

V5.2

To register each license, use the following procedure:

1 If you have not already done so, log in to the SYSTEM account. At the dollar sign ($) prompt, enter the following command and press RETURN:

```
$
```

The procedure displays the following menu:

```
VMS License Management Utility Options:

     1. Register a Product Authorization Key
     2. Amend an existing Product Authorization Key
     3. Cancel an existing Product Authorization Key
     4. List Product Authorization Keys
     5. Modify an existing Product Authorization Key

     9. Exit this procedure

Type '?' at any prompt for a description of the information
requested.

Enter one of the above choices [1]:
```

2 Type 1 and press RETURN. The procedure displays the following message:

```
* Do you have your Product Authorization Key? [YES]:
```

Make sure you have a PAK for the license you are registering. Type Y and press RETURN.

3 The procedure displays the following information and prompts:

```
   The REGISTER option allows you to add a new license to a license
   database.  A Product Authorization Key (PAK) provides the product
   name and information you need to register the license.  You must
   enter all the information provided by your PAK exactly as specified.

PAK ID:
                       Issuer [DEC]:
         Authorization Number []:
```

a. If DEC is the issuer, press RETURN. Otherwise, enter the name given on the PAK.

b. Enter the authorization number that appears on the PAK and press RETURN.

4 The procedure asks for the following information:

```
PRODUCT ID:
                Product Name []:
                    Producer [DEC]:
```

a. Enter the product name that appears on the PAK and press RETURN.

b. Press RETURN to specify DEC as the producer.

5 The procedure asks for the number of units:

```
NUMBER OF UNITS:
              Number of Units []:
```

If the PAK lists the number of units, enter the number and press RETURN.

If the PAK does *not* list the number of units, do not enter a value. Press RETURN and go to the next step.

6 The procedure asks for the key level:

```
KEY LEVEL:
                        Version []:
```

a. If the PAK lists a version number, enter the number. Press RETURN and go to step 7.

b. If the PAK does *not* list a version number, do not enter a value. Press RETURN. If you do not enter a version number, the procedure asks for the product release date:

```
        Product Release Date  []:
```

Enter the product release date that appears on the PAK and press RETURN.

7 The procedure asks for the key termination date:

```
KEY TERMINATION DATE:
         Key Termination Date []:
```

If the PAK lists a key termination date, enter the date and press RETURN.

If the PAK does *not* list a key termination date, do not enter a value. Press RETURN and go to the next step.

8 The procedure asks for the following information:

```
RATING:
      Availability Table Code []:
          Activity Table Code []:
```

a. If the PAK lists an availability table code value or the word CONSTANT followed by an integer, enter the information and press RETURN. If the PAK does *not* list this information, do not enter a value. Press RETURN after the prompt.

b. If the PAK lists an activity table code value or the word CONSTANT followed by an integer, enter the information and press RETURN. If the PAK does *not* list this information, do not enter a value. Press RETURN after the prompt.

Usually, a PAK lists information for either the availability table code *or* the activity table code.

9 The procedure asks for the following information:

```
MISCELLANEOUS:
                    Key Options []:
```

a. If the PAK does *not* give values for this item, press RETURN after the prompt and go to the next step.

If the PAK gives values for this item, enter the values and press RETURN. For example, if you are running the VMS operating system in a VAXcluster environment, the PAK lists MOD_UNITS and NO_SHARE as key options. Enter these options and press RETURN as shown in the following example:

```
                    Key Options []: MOD_UNITS,NO_SHARE
```

b. If you are registering a VMS license, the procedure displays the following message:

```
   This Product Authorization Key has been provided with the NO_SHARE
   option.  This requires that this key be restricted to a specific
   node within a cluster.

   Enter the node name of the VAXcluster member to which this license
   is restricted.

Is this PAK restricted to a cluster member node? [YES]:
```

If you press RETURN after the prompt, the procedure asks for the name of the node to which the license is restricted. The display includes the current node name as the default:

```
Node this PAK is restricted to (SCS node name) [JUPITR]:
```

If you are registering this PAK for the current node, press RETURN. Otherwise, enter the node name of the VAX computer for which you are registering this PAK and press RETURN. Use the same name that you intend to use for the SYSGEN parameter SCSNODE.

10 The procedure asks for the following information:

```
                  Product Token []:
                    Hardware-Id []:
```

If the PAK lists values for product token and hardware ID, enter them and press RETURN. Otherwise, press RETURN.

11 The procedure prompts for the checksum:

```
                    Checksum []:
```

Enter the checksum given on the PAK and press RETURN.

Note: Currently the checksum string always begins with the number 1, which is the only number in the string. The other sixteen positions are always alphabetic characters from A through P.

12 The procedure displays the information that you entered. For example:

```
      License Database File:  SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB
                     Issuer:  DEC
              Authorization:  USA126087
                   Producer:  DEC
                Product Name:  VAX-VMS
                       Units:  460
                        Date:
                     Version:  5.2
           Termination Date:  31-DEC-1988
               Availability:  E
                   Activity:
                    Options:  MOD_UNITS,NO_SHARE
                      Token:
                Hardware ID:
                   Checksum:  1-ADEB-DOCJ-NENC-KDBM

    This authorization key is restricted to: JUPITR

Is this information correct? [YES]:
```

Carefully compare the information on the screen with the information on the PAK. If the information is correct, type Y and press RETURN.

If it is incorrect, type N and press RETURN.

Note: If you enter any of the information incorrectly, an error message is displayed and the license is not registered. Keep in mind that a CHECKSUM error can result when you enter incorrect information for the other items on the PAK. If an error message is displayed, carefully check all the data that you entered.

When you indicate that the information is incorrect, the procedure displays the following question:

```
Do you wish to make corrections? [YES]:
```

To correct the data, type Y (for YES) and press RETURN.

13 If you choose to make corrections, the procedure steps you through all of the questions again, but supplies the data entered previously as defaults for each data field. Each time the procedure displays correct information, press RETURN. If the procedure displays incorrect information, enter the new data to be used and press RETURN. To cancel the current data without entering new data, enter the backslash (\) character and then press RETURN.

If you entered all the information correctly, the procedure displays a confirmation such as the following:

```
Registering VAX-VMS license in SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB...
```

If you entered some information incorrectly but do not choose YES to make corrections, the procedure may display the following:

```
Registering VAX-VMS license in SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB...
%LICENSE-F-BADCHK, checksum does not validate

Do you wish to make corrections? [YES]:
```

To correct the data, type Y (for YES) and press RETURN:

If you enter an incorrect checksum string the procedure responds as follows:

```
 1-ADEB-DOCJ-NENC-KDBX is not a valid license checksum string.

    The license checksum is a 17-character verification string created by
    the PAK issuer for each PAK.  The checksum string is presented in
    the format n-cccc-cccc-cccc-cccc, where n is an integer and c is a
    character from A through P.  A PAK presents the checksum string with
    hyphen (-) characters for readability.  Because the LMF does not count
    them for authorization, you can leave them out.  Otherwise, you must
    enter the checksum string exactly as specified as your PAK or PAAM.

    If a default value is displayed and you wish to use it just press
    the RETURN key.   The license checksum is a required field for the
    REGISTER and AMEND options.

                      Checksum []:
```

Enter the correct checksum at the prompt and press RETURN.

14 After the license is successfully registered, the procedure asks if you want to activate the license on the current node, as follows:

```
Do you want to LOAD this license on this system? [YES]:
```

If you registered the PAK on a standalone system and you want to make the software available (active) immediately, type Y and press RETURN.

If you registered the license in a VAXcluster environment but do not want to make it available (active) on the current node, type N and press RETURN. After you exit this procedure, you can enter a LICENSE LOAD command to activate the license on a different node.

License activation is a process that makes a registered license known to a system. For details of license activation in a VAXcluster environment, see the *VMS License Management Utility Manual.*

If you type Y and the license is successfully activated, the procedure displays an information message and returns you to the first menu and prompt as follows:

```
 %LICENSE-I-LOADED, DEC VAX-VMS was successfully loaded with 460 units

    VMS License Management Utility Options:

        1. Register a Product Authorization Key
        2. Amend an existing Product Authorization Key
        3. Cancel an existing Product Authorization Key
        4. List Product Authorization Keys
        5. Modify an existing Product Authorization Key

        9. Exit this procedure
```

```
    Type '?' at any prompt for a description of the information
    requested.

Enter one of the above choices [1]:
```

15 To register another PAK, type 1 or press RETURN. Then respond to the questions, again entering information from a license PAK. Note that this time the procedure provided a default option [1] with the first menu. The procedure automatically supplies the data previously entered when each question is asked a second time. When you register a second PAK, you need only enter data that is different from the last data entered. Some information, such as table codes and options, may be common to multiple PAKs. You can save typing time by registering all your PAKs during one VMSLICENSE session.

16 To exit the procedure, type 9 and press RETURN. The procedure finishes and returns you to the dollar sign ($) prompt.

If you received an error message when the procedure attempted to load a license, this does not affect the license registration, only license activation. Read the sections of the *VMS License Management Utility Manual* that describe activating a license. For example, read the LICENSE LOAD command description.

Note: If you are registering your licenses as part of a VMS operating system installation, refer to the installation guide that came with your VAX computer. This guide contains information on additional tasks that you need to complete (such as running UETP) before using the system.

1.3 Managing VMS Service Update Kit Licenses for Service Customers

V5.2

If you are a VMS Service Customer, read this section for information about LMF$CONFIG.COM, a command procedure provided to help in the transition to VMS Version 5.2. If you do not have a VMS Service Update Kit the information in this section may not apply to you.

LMF$CONFIG.COM is a command procedure that generates a VMS Service Update PAK (SUP) when you apply the VMS Service Update Kit Mandatory Update (MUP) software during VMS installation or upgrade. LMF$CONFIG.COM is copied from the MUP to the SYS$UPDATE directory and executed, providing a valid VMS license without you having to manually register a VMS PAK.

LMF$CONFIG.COM, provided since VMS Version 5.0, generates SUPs for all processors supported by VMS Version 5.0. For any processor released since VMS Version 5.0, you must manually register the VMS PAK provided with your VMS kit, using VMSLICENSE.COM or the LICENSE REGISTER command. This means VAXclusters that include older *and* newer processors together have some VMS licenses generated by LMF$CONFIG and some entered manually. When LMF$CONFIG attempts to generate a SUP for a newer processor, it displays an error message reminding you to register a PAK for that processor:

```
The LMF$CONFIG.COM procedure does not support the VAX 6320.

This procedure was developed to transparently register VAX-VMS
Product Authorization Keys (PAKs), for systems released prior to
VMS Version V5.0.

Please use the SYS$UPDATE::VMSLICENSE.COM or LICENSE commands to
register your VAX-VMS PAK. These procedures are documented in the
VMS License Management Utility Manual.
```

The system displayed (VAX 6320 in this example) differs according to hardware type. You must register the VMS PAK provided with your newer VAX computer. Use VMSLICENSE.COM or the LICENSE REGISTER and LICENSE MODIFY commands.

LMF$CONFIG.COM performs the following actions on supported systems:

- Creates a system specific LICENSE database called SYS$SPECIFIC:[SYSEXE]LMF$SYSTEM.LDB. Any previous versions of this database are deleted.
- Determines the appropriate VMS SUP for your system.
- Registers the SUP in the system specific database, and displays a confirmation message upon success.
- During a cluster upgrade, runs on the other members of the cluster as they reboot following the upgrade. If you add a new member to the cluster, LMF$CONFIG.COM is run as part of the CLUSTER_CONFIG ADD option.

Each supported system in a VAXcluster environment can have a system specific database. The databases are *only* for the VMS licenses registered by LMF$CONFIG.COM.

Each time LMF$CONFIG runs on a system, it creates a new system specific license database. If you accidentally modify a system specific license database, the modifications are lost when LMF$CONFIG creates the new database.

Register all VMS PAKs for newer processors, your VAXcluster SUP, and the PAKs for any other products in the default common LICENSE database SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB. Use License Management Utility (LICENSE) commands, or the command procedure VMSLICENSE.COM to access this default LICENSE database; LMF$CONFIG never modifies it. When you register VMS PAKs manually, be sure to assign them to the proper SCS node name as documented with VMSLICENSE.COM or the LICENSE MODIFY command.

Each time a system starts up the License Management Facility (LMF) activates all licenses from the common database as well as any VMS licenses generated by LMF$CONFIG. License activation is a process that makes a license known to the current computer by loading information into the computer's memory. Although LMF generally displays a confirmation message when a registered license is activated, VMS SUPs in system specific databases do not produce confirmation messages.

The procedure LMF$CONFIG may ask you to supply some information as follows.

- When you run LMF$CONFIG on a MicroVAX, the procedure asks you to specify the number of users for which you are licensed. Be sure to specify the choice that represents the number of users on your VMS license.
- If you have not assigned a system communications services (SCS) node name to your computer, the command procedure LMF$CONFIG.COM asks you to supply it. Because all VMS SUPs must be associated with a particular VAX computer, LMF$CONFIG attempts to determine the computer's SCS node name. The SCS node name is defined by the SYSGEN parameter SCSNODE. If the procedure cannot determine the SCS node name, you are asked to supply it. If the SUP is for a standalone VAX computer, you do not need an SCS node name. If the node will become part of a VAXcluster environment, however, you must enter the correct SCS node name. Otherwise, LMF cannot activate the VMS SUP on the VAX computer in a cluster.

If you change the SCS node name after the procedure LMF$CONFIG.COM registers a license, you must modify the automatically-registered license on that node to reflect the change. From the node on which you changed the name, enter a command of the following format:

```
$
_$
```

1.4 Modifying License Units with the License Management Utility

V5.2

The following information is provided as a supplement to other License Management Facility (LMF) documentation. Before you use this information, you should read the *VMS License Management Utility Manual*, and you should understand the terms and conditions of your license agreement.

For VMS Version 5.2, many products require a PAK that includes license data to be registered in the LICENSE database. Some PAKs provide a MOD_UNITS option, which lets you modify the size of the registered licenses. If you have registered a license with the MOD_UNITS option, you can modify the size of the license to match the product to your VAX computer or VAXcluster environment. You can modify all licenses with the MOD_UNITS option, including those that specify the following:

- A size of 0 license units (unlimited availability)
- A predetermined size with a number of license units

To determine the options and size specified for your license, enter a command in the following format and press RETURN:

```
$
```

The following display of a license list appears:

```
Use Ctrl/Z to exit, PF3-PF4 for Previous-Next Screen and Arrow keys to
Scroll.


 License Management Facility

 License Database File:       ART::SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB
 Created on:                  17-AUG-1989
 Created by user:             MONET
 LMF Version:                 V1.0

 ------------------------------------------------------
 Issuer:                      DEC
 Authorization:               USA-10
 Product Name:                FORTRAN
 Producer:                    DEC
 Units:                       900
 Version:                     5.2
 Date:                        (none)
 Termination Date:            21-DEC-1991
 Availability:                F (Layered Products)
 Activity:                    0
 Options:                     MOD_UNITS
 Hardware ID:

 Revision Level:              1
 Status:                      Active
 Command:                     REGISTER
 Modified by user:            DEGAS
 Modified on:                 21-AUG-1989 14:32:23.41
```

This display of a modifiable license includes MOD_UNITS next to the *Options* label. The size of the license is displayed next to the *Units* label. The license for this example provides the MOD_UNITS option and 900 license units.

You cannot activate licenses registered with fewer license units than a VAX computer requires. If your VAX computer or VAXcluster environment needs a different number of license units from the number registered with your license, find an appropriate license unit value for your VAX computer, and change the size of your license, as follows:

1. Enter the LICENSE LIST/FULL *product-name* command to display the license. Examine the display. Look for a code A through F, next to either the *Availability* label or the *Activity* label in the LICENSE LIST display. The codes, which designate license type, determine the appropriate license size for your VAX computer.
2. Find the name of your VAX computer in the first column of the VMS Version 5.2 License Unit Requirement Table (LURT) (Table 1–1).

Table 1–1 License Unit Requirement Table (LURT)

	License Types by Code					
	VMS				SIP	LP
System Marketing Model	A	B	C	D	E	F
VAX 11/730	10	NA	NA	NA	230	50
VAX 11/750	12	NA	NA	NA	230	100
VAX 11/780,785	13	NA	NA	NA	230	100
VAX 6210,6310	58	NA	NA	NA	230	300
VAX 6220,6320	69	NA	NA	NA	230	600
VAX 6230,6330	81	NA	NA	NA	400	900
VAX 6240,6340,6350	93	NA	NA	NA	400	1200
VAX 8200,8250	20	NA	NA	NA	230	100
VAX 8300,8350	25	NA	NA	NA	230	200
VAX 8530	65	NA	NA	NA	230	400
VAX 8550,8700,8810	72	NA	NA	NA	400	600
VAX 8600,8650	28	NA	NA	NA	230	400
VAX 8800,8820	93	NA	NA	NA	400	1200
VAX 8830,6360	119	NA	NA	NA	600	1800
VAX 8840	143	NA	NA	NA	600	2400
MicroVAX II	18	NA	100	NA	230	50
MicroVAX 2000	18	NA	100	NA	230	20
MicroVAX 3500,3600,3800, 3900	60	NA	100	NA	230	300
MicroVAX 3300,3400	60	NA	100	NA	230	100
VAXstation II,II/GPX	NA	NA	NA	100	50	10
VAXstation 2000,2000/GPX	NA	NA	NA	100	50	10
VAXstation 3100,3200,3500, 3520,3540,8000	NA	NA	NA	100	50	10
VAXserver 2000	NA	52	NA	NA	50	10
VAXserver 3300,3400,3500, 3600,3900	NA	100	NA	NA	50	10
VAXserver 6210,6310	NA	1443	NA	NA	230	200
VAXserver 6220,6320	NA	1737	NA	NA	230	400

Key to License Type Codes

A–VMS Capacity
B–VMS Server
C–VMS Concurrent User
D–VMS Workstations
E–System Integrated Products
F–Layered Products

3 Find the row with the appropriate name and the column with the code corresponding to the type of license you have. Note the value at the

intersection of the row and column. For example, find the intersection of VAX 8800 with column F (the value shown is 1200). Unless NA (meaning a value is not applicable for this license type) appears, the value that you find is the number of units required for your VAX computer and type of license.

4 Modify your license by entering the value found in Table 1–1 as the parameter in a LICENSE MODIFY/UNITS=*number* command. For example, to modify a FORTRAN license running on a VAX 8800, enter the following:

```
$
```

If you have entered the correct value for your license and VAX computer, the license should activate with the next LICENSE LOAD command. To activate the modifications immediately on a previously activated license, enter the following commands as well:

```
$
$
LMF-I-LOADED, DEC FORTRAN was successfully loaded with 1200 units
```

For details, see the *VMS License Management Utility Manual.*

The license unit requirements provided by the License Unit Requirement Table are subject to change.

1.5 License Management Facility Notes

V5.2

The following list is offered to help new users with some common concerns and questions regarding the License Management Facility (LMF). For full explanations of these issues, see the *VMS License Management Utility Manual.*

- If you do not have a valid VMS license that is registered and activated, the system displays a warning message as part of system startup and restricts system use to the operator's console, OPA0.
- If a checksum error is displayed when you register a license, check *all* the fields of data that you have entered.
- After your PAKs are registered, they are automatically activated (loaded) as part of each system startup.
- If a VMS availability license is registered with insufficient license units for the specified VAX computer, the system displays a warning message at system startup but allows normal system use.
- If a VMS activity license is registered with insufficient license units, the system displays the following message when the user (process) attempts to log in:

```
%LICENSE-F-EXCEEDED, licensed product has exceeded current license limits
```

Users can always log in to the operator's console, OPA0, however.

- The default LICENSE database is located in the file SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB. You can move the database, although Digital does not recommend it. If you move the database, you must either define the logical name LMF$LICENSE at the system level to point to the new database, or use the /DATABASE=*filespec* qualifier with all LICENSE commands. To redirect LMF to another database location on a more permanent basis, insert the following line in the command procedure SYS$MANAGER:SYLOGICALS.COM:

  ```
  $
  ```

  If you specify a device other than SYS$SYSDEVICE, you must also mount the specified disk from the SYLOGICALS.COM command procedure.

- If you have a service contract with Digital for a VMS Version 5.2 upgrade, you may have VMS licenses registered in a separate LICENSE database located in the file SYS$SPECIFIC:[SYSEXE]LMF$SYSTEM.LDB. Each node of a VAXcluster environment may have a separate database containing only a VMS license. To access these VMS licenses, you must use the /DATABASE qualifier with your LICENSE commands. You should not need to modify these licenses unless you change the SCS node name of the VAX computer. For information about these licenses and the LMF$CONFIG command procedure that created them see the *VMS License Management Utility Manual.*

- If you have multiple system disks in a VAXcluster environment, where all the systems can access one of the system disks, put your common LICENSE database on the readable disk. For any systems that boot from a separate system disk, you must redirect LMF to the LICENSE database. Define the logical name LMF$LICENSE as the disk containing the database.

  If you have multiple system disks in a VAXcluster environment, where some systems cannot access one of the system disks, and where you must keep separate LICENSE databases, keep the databases identical. Whenever one database is modified, you must copy it to update the other databases.

- Each VMS license is restricted to a single node. You must assign a System Communications Services (SCS) name to the license when you register with the VMSLICENSE.COM command procedure, or you must enter a LICENSE MODIFY/INCLUDE=*node-name* command after you register the license. Although you can successfully activate an unassigned VMS license on a standalone system, you cannot activate one in a VAXcluster environment.

Note: The SCS node name is not necessarily the DECnet node name. SCSNODE is a System Generation Utility (SYSGEN) parameter.

1.6 VMS License Types

V5.2

The VMS operating system uses one of the following four different kinds of licenses depending on the hardware and software configuration used and currently supported:

- VMS Availability License

  This type of license provides unlimited access to the users on a VAX computer or VAXcluster environment. These licenses are sometimes referred to as capacity licenses or clusterwide licenses. VMS availability licenses are sized according to License Unit Requirement Table entries for each VAX computer.

- VMS Multiuser License

  This type of license provides use according to a specified number of concurrent users. This is an activity-based license.

- VMS Workstation License

  This type of license provides use for a single user on a VAX workstation. Workstation licenses actually use the same licensing formulas as the VMS multiuser license. This is an activity-based license.

- VMS Server License

  This type of license provides use for a single user on a VAXserver. Server licenses actually use the same licensing formulas as the VMS multiuser license. This is an activity-based license.

1.7 License Activity Use Definition for VMS Licenses

V5.2

The *VMS License Management Utility Manual* describes a type of license based on the number of concurrent users called an activity license. Every product has the option to define an activity as related to the License Management Facility. VMS defines activities, sometimes referred to as VMS users, as follows:

- Each remote terminal connection is considered an activity. This is true even if you set host to your local node (SET HOST 0).
- Each connection from a terminal server is considered an activity.
- A multiple-window session on a workstation is considered one activity, regardless of the number of windows.
- A batch job is *not* considered an activity.
- A remote network connection (other than a remote terminal connection) is *not* considered an activity.

VMS determines the number of activities through the LOGINOUT image as part of initiating a new interactive process.

1.8 VAXcluster License Notes

V5.2

In VMS Version 5.2 you *must* register your VAXcluster license before you can use your VAXcluster software. If you are a service customer with a VMS Service Update Kit (W-KIT), you must register the Software Update PAK (SUP) supplied with your kit. During system startup, each VAXcluster node checks for a valid VAXcluster license. If you have not registered a VAXcluster license for a VAXcluster environment, each node displays the following message at system startup time:

```
%LICENSE-E-NOAUTH, DEC VAXCLUSTER use is not authorized on this node
-LICENSE-F-NOLICENSE, no license is active for this software product
```

Starting with VMS Version 5.2, you cannot log into any terminal other than the operator's console, OPA0. If you attempt to log in to a VAXcluster node before you register and activate a VAXcluster license, the system displays the following message:

```
DEC VAXCLUSTER license is not active
```

Note: Although interactive logins are disabled, your VAXcluster software performs normally as configured by startup procedures and system parameters. Systems not running VAXcluster software are unaffected.

1.9 DECnet–VAX License Notes

V5.2

There are two DECnet-VAX licenses, the end node license named DVNETEND, and the routing node license named DVNETRTG. All routing nodes must have a routing license. Each end node can have either an end node license or a routing license. If neither license is registered and activated, DECnet will not start, limiting your use to local DECnet only (SET HOST 0). If DECnet is running when you register your license, you must stop and restart DECnet.

You can control which VAXcluster nodes have access to each kind of license. Using the LICENSE MODIFY/INCLUDE=*(node-name[,node-name,...])* command, you can assign licenses to nodes and limit access as needed. For example, you can assign a routing node license to only one VAXcluster node, and assign the end node licenses to the remaining VAXcluster nodes. If you choose this approach, make sure that you assign each end node license to the same list of nodes. That is, specify identical *include lists* for each license of the same type. For details, see the *VMS License Management Utility Manual.*

1.10 VAX RMS Journaling License Notes

V5.2

On systems that do not have journaling licenses registered and activated, users cannot access any files marked for journaling.

1.11 VAX Volume Shadowing License Notes

V5.2

If you have not registered and activated a license for VAX Volume Shadowing, each node using volume shadowing displays the following message at system startup time:

```
%LICENSE-E-NOAUTH, DEC VOLSHAD use is not authorized on this node
-LICENSE-F-NOLICENSE, no license is active for this software product
-LICENSE-I-SYSMGR, please see your system manager
```

No further shadow-set mount operations will succeed.

2 General User Release Notes

This chapter discusses information about the VMS Version 5.2 operating system that is of interest to the general user.

For information about the new features included in VMS Version 5.2, see the *VMS Version 5.2 New Features Manual*.

2.1 $ENTRY — New Symbol

V5.2

In VMS Version 5.2, successful execution of a PRINT or SUBMIT command creates or updates the global symbol $ENTRY. The value of this symbol is a string containing the entry number of the print or batch job just entered. To retain a job's entry number for later reference, immediately store its value in another symbol to avoid losing the information when $ENTRY is updated to reflect the most recent job queued. The following example illustrates its use:

```
$
Job TEST (queued CLUSTER_BATCH, entry 493) pending
$



$
```

2.2 Command Procedures — Restriction

V5.0

The VMS operating system now requires that all commands, full-line comments, and labels in command procedures be preceded by the dollar sign ($) character. Although users have always been instructed to place a dollar sign before commands and labels, command procedures that omitted dollar signs before labels did not necessarily stop executing. VMS Version 5.0, however, treats labels without dollar signs as data lines. Any reference to a label without a dollar sign will not execute as expected.

2.3 DECwindows Notes

The following sections describe release notes applicable to DECwindows.

2.3.1 DECterm Notes

V5.1-1

DECwindows includes a VT300 series DECterm Terminal Emulator. You can invoke this terminal emulator only through the Session Manager's Create/Terminal Window menu or the DECTERM_PORT routine. The DECTERM_PORT routine is described in detail in Section 4.8.2.

The following sections describe relevant DECterm information.

2.3.1.1 Initialization

V5.1-1

The following information is specific to DECterm initialization:

- To avoid having your DECterm windows shrink to the default size of 80 characters by 24 lines unexpectedly, Digital suggests that system-wide and user login command procedures (SYS$SYLOGIN.COM and LOGIN.COM) not execute the SET TERMINAL/INQUIRE command on DECterm windows.

  The DECterm controller provides VMS with the proper characteristics and size of DECterm windows, and hence SET TERMINAL/INQUIRE is unnecessary.

  To make login procedures work correctly on both DECterms and non-DECterms, use the following commands:

```
$
$
$
$
$
$
$
$
$
$
$
```

  This routine bypasses the SET TERMINAL/INQUIRE on DECterm, SET HOST, and VWS. It works because DECterm's initial terminal type is TT_DECCRT3.

- If you attempt to resize a DECterm window before you see a prompt in the window, the window may disappear.

2.3.1.2 Keyboards and Languages

V5.1-1

The following information is specific to DECterm keyboards and languages:

- To create a composed character in DECwindows, you must press the COMPOSE and space keys at the same time, not the COMPOSE key alone as on character-cell keyboards. The COMPOSE key is a modifier and is used like a SHIFT key.
- Keyclick, auto-repeat, keyboard dialects (national keyboards), keyboard usage mode (data processing / typewriter mode) and caps lock / shift lock should be changed in the Session Manager on a workstation-wide basis, rather than through DECterm.
- National replacement character sets (NRCS) are selected in the Customize / 7-bit NRCS Selection menu in DECterm. Their selection is independent of both the keyboard dialect and the keyboard usage mode. For example, you must change both the Session Manager and DECterm to use the French NRCS with the French keyboard.
- The Dutch NRCS is implemented, even though its use is no longer recommended. It is likely to be removed from future versions of DECterm. The florin character is drawn as a currency sign, ¤ (character 164 in ISO Latin-1 or 168 in DEC Multinational).

- When the keyboard becomes locked (for example, because the input silo is full), DECterm rings the bell for each character that comes in until the lock condition is cleared, rather than disabling keyclick as is done on the VT320/VT330/VT340.

V5.2

- In some situations DECterm can continue to auto-repeat a key even after the key has been released; this is especially a problem with WPS-PLUS under ALL-IN-1. To avoid this problem, activate the Lock User Features toggle button in the Customize General dialog box.

2.3.1.3 Fonts and User Interface

V5.1-1

The following information is specific to DECterm fonts and user interface:

- The Big font / Little font radio buttons in the Customize Window have no effect when using the 100 dot per inch fonts; this is because the 75 dpi big font is used for both the big and little fonts on 100 dpi monitors.
- The Dismiss function in DECterm's dialog boxes is equivalent to the Cancel function in other DECwindows applications.

2.3.1.4 Text

V5.1-1

The following information is specific to DECterm text:

- User loadable characters (DRCS), print screen, local mode and control representation mode are not implemented.
- RIS (a <ESC> c sequence) sets certain parameters to their factory defaults instead of to their saved settings. As usual, its use is discouraged.
- The checkerboard character (character 97 in the special graphics set) is used as an error character instead of the reverse question mark, and in some cases it is not displayed at all.

2.3.1.5 Graphics

V5.1-1

The following information is specific to DECterm graphics:

- The dialog box will be black on a 4-plane color system if you bring up a dialog box in the Customize window while DECterm is using its private color map. To avoid this problem, either bring up all the dialog boxes you plan to use before displaying any graphics or exit from the graphics, clear, and reset the window (this restores the default color map) before bringing up a dialog box.
- There are various problems when displaying ReGIS or sixel graphics while recording lines off the top. In some cases, recorded lines can overprint each other when you use the vertical scroll bar, and ReGIS pictures can be scaled to the wrong size. To avoid these problems, turn off the "Record lines off top" button in the Customize Window menu when displaying ReGIS or sixel pictures.
- Only graphics, not text, are written to the graphics backing store. When part of the window has to be redrawn, DECterm will draw the graphics portion first and then overlay it with text. As a result, the window might not look the same when redrawn as it did when it was occluded.

- ReGIS standard character sizes are different from those in the VT340. The VT340 uses a character size of 8 by 20 pixels and a display size of 9 by 20 pixels, while the ReGIS character size in DECterm depends on the font being used. This means text labels are not aligned correctly in applications such as DECgraph and DECslide. The standard character sizes are likely to change in future versions of DECterm in an attempt to correct this problem.
- ReGIS addresses the entire window, resulting in differently scaled pictures than on a VT330/VT340, unless the window height is exactly 24 rows.
- ReGIS pictures can be scaled incorrectly when the window is resized, or when the font size is changed.
- Terminate one-shot input mode (R(I0)) from the ReGIS locator reporting by pressing a button on the mouse. Pressing one or more keys on the keyboard will send information to the application, but it will not be followed by ReGIS coordinates. You cannot control the locator position with the arrow keys, you must use a mouse (unless you use CTRL/F3 to do this for the entire workstation). In addition, this version of DECterm does not support DECLKB (called DECLBD in the VT330/VT340 programmer reference manual), so pressing a button will send an escape sequence followed by the coordinates of the mouse position.

  Application writers modifying their applications to support DECterm should consider using the text locator mode, described in Section 4.8.4.
- The following are not implemented: ReGIS command display mode, scrolling, hardcopy, and output cursors. The workstation mouse is used as the input cursor, but its shape cannot be changed with the S(C(I)) command.
- Both monochrome and color systems emulate a 16-entry color map, as on the VT340, rather than a 4-entry color map as on a VT330 (or VT240).
- ReGIS always uses the DEC Multinational character set, regardless of whether ISO Latin-1 was chosen in the Customize/General menu.
- There is no Customize option to disable the reporting of macrograph contents, as on the VT330/VT340. Proper care should be taken if sensitive information is stored in macrographs.
- Sixel files can over-write the window borders. To restore the borders, shrink the window to an icon and then expand the icon to a window. There is also a problem on single plane systems (such as the VS2000) where sixel pictures are redrawn with reversed colors when they are refreshed.
- Sixels do not scroll when they reach the bottom of the window, so it is best to position the cursor at the top of the window before displaying a sixel file.
- The vertical position after leaving sixel mode is the same as it was on entry, and does not account for the length of the sixel picture.

- The background parameter in the sixel device control string (DCS) has no effect; lines actually over-written by the picture are erased, but the rest of the window is unaffected, regardless of the setting of the parameter.
- The horizontal and vertical extent parameters in the set attributes command (", ASCII code 34) are not implemented.

2.3.2 DECterm Window Notes

V5.1

The following notes pertain to DECterm windows:

- DECterm cannot use the logical name DECW$USER_DEFAULTS if it is defined in your LOGIN.COM file. DECW$USER_DEFAULTS must be defined in the process in which the controller is running; it is normally defined in LNM$SYSTEM.
- On a 4-plane color system, DECterm might create a private color map for each window in order to emulate a 4-plane VT340 on a 4-plane workstation. DECterm creates the color map when any ReGIS or sixel graphics are displayed in the window. The window manager loads the private color map when that window has the input focus, and it loads the default color map when the window loses the input focus. When the private color map is loaded, other windows are not displayed with their correct colors.

  To restore a DECterm window to using the default color map, first clear the window by selecting Clear Display from the Commands menu and then reset the terminal by selecting Reset Terminal from the Commands menu.

2.3.3 Desktop Application Notes

The following sections provide notes about the DECwindows Desktop Applications.

2.3.3.1 Clock Note

V5.1

The clock does not notice that you have changed the system time backward while the clock was running. You must exit and rerun the clock if you change the system time.

2.3.3.2 DDIF Viewer Restrictions

V5.1

The DIGITAL Document Interchange Format (DDIF) Viewer has the following known restrictions:

- Complex document layout, including page and galley layout, is not supported. Some text display attributes are also not processed when you display a document.
- If, on the command line, you specify a file that contains a graphic that displays in the first window, the Viewer menu bar options and scroll bar arrows do not appear. This problem can be avoided by selecting such files from the Viewer open file selection box. Another option is to

use the Resize button to change the window size so that the Viewer menu bar options and scroll bar arrows appear.

- The Viewer might exit if you open several DDIF files containing graphics in a single session. Another option is to limit the number of graphics files viewed and reinvoke the Viewer to view more files.
- Graphics displayed on a monochrome screen use a black foreground and white background. If the default user settings include a default dark background, the graphics will probably not be seen until they are re-exposed; that is, the screen will appear blank because the graphics have been drawn black on black. Another option is to use the Resize button to change the window size.
- If scrolling an image is not possible, it might be represented by an improperly specified DDIF bounding box. A workaround is to use the resize button to change the window size.
- The Compound Document Architecture (CDA) Toolkit requires that documents referred to by DDIF document external references exist in the current default directory if a full, file path name is not specified in the DDIF file external reference description.

2.3.3.3 EVE Procedure EVE$DECLARE_HELP_LIBRARY — Restriction

V5.1

The third parameter to the EVE procedure EVE$DECLARE_HELP_LIBRARY is optional. However, if you do not specify a value for the parameter, you must supply a null string ("") as a placeholder. This restriction will probably be removed in a future version of VMS.

2.3.3.4 DECwindows Mail — Problems and Restrictions

V5.1

DECwindows Mail has the following known problems and restrictions:

- SYS$SCRATCH defines the directory where any temporary files are created. If DECwindows Mail is run in a detached process, SYS$SCRATCH must be explicitly defined in that process.
- When you read a mail message while the Auto Refile option is set, the message is immediately moved to the MAIL folder. If you delete or move the message, the action does not take place until the INBOX folder is closed. Therefore, if you read a message, delete it, and open the MAIL folder without first closing the INBOX or NEWMAIL folder, you see the deleted message listed in the MAIL folder. When you eventually close the INBOX or NEWMAIL folder, the deleted message is deleted from the MAIL folder.
- Printing is performed through the standard DECwindows print function, which does not automatically use print defaults specified in your VMS Mail Utility user profile. Any modifications made to the print options using either the Print . . . menu item from the File pull-down menu or the Modify Print Defaults . . . menu item from the Customize pull-down menu are remembered as long as DECwindows Mail is active, but are not remembered from one DECwindows Mail session to another.
- When using small icons (the Session Manager default), the DECwindows Mail icon does not darken when new mail arrives. The icon picture does change to show *two* overlapping envelopes.

- The Delete Drawers . . . menu item deletes all messages and folders within a drawer but does not delete the underlying mail file itself. To reclaim the disk space used by the drawer and prevent the drawer from being listed again in the index by a subsequent scan, delete the file using FileView or DCL.
- When you use a dialog box to move or copy messages selected in the main window, selecting text within the dialog box (which occurs automatically when using TAB to move to a nonempty text field) removes the main window selection. Clicking on the OK button at this point gives a warning that messages must be selected. If the messages become unselected, click on the main window title bar to restore the selection before clicking on the OK button in the dialog box.
- System managers should be sure to set MAIL$SYSTEM_FLAGS at system startup time as described in the *VMS Mail Utility Manual*.
- DECwindows Mail enables deleted-mail purging by default. This is true even if you have set NOAUTO_PURGE through the VMS Mail Utility.

2.3.3.5 DECwindows Mail — Problems Corrected

V5.2

The following problems with DECwindows Mail have been corrected:

- Increasing the vertical size of DECwindows Mail's main window in the pane interface could result in commands operating on the wrong messages, and in some cases caused access violations.
- In some instances, DECwindows Mail aborted with an access violation if no message was selected during the extract, print, reply, and forward operations. In VMS Version 5.2, a message box asks the user to select a message in this case.
- Deselecting (with Shift/MB1) the last of a group of selected messages could lead to a subsequent access violation.
- If an access control string was used in a distribution list file specification, the password was not replaced and all recipients saw the username and password used to access the distribution list. In VMS Version 5.2, the string "password" is substituted for the real password.

2.3.3.6 DECwindows Bookreader Restriction

V5.1

The DECwindows Bookreader has the following restrictions:

- For books with large indexes, you cannot drag-scroll from the the top of the index to the bottom, or from the bottom to the top, in a single operation. You can get to the bottom or top of the index by using two drag operations or by using other scroll-bar functions.
- When the Bookreader starts, it looks for the file DECW$BOOK:LIBRARY.DECW$BOOKSHELF. If LIBRARY is defined as a logical name, the Bookreader is not able to open the library file. To work around this, define a logical, DECW$BOOKSHELF to be LIBRARY.DECW$BOOKSHELF before invoking the Bookreader.

2.3.4 FileView Notes

V5.1

FileView gives you access to DECwindows applications and provides commands for you to work with files. See the *VMS DECwindows User's Guide* for more information about FileView.

FileView has the following restrictions:

- If you start FileView from the Session Manager window, you cannot have DCL INQUIRE statements in your LOGIN.COM or SYLOGIN.COM files. FileView executes both your LOGIN.COM and SYLOGIN.COM command procedures and is an interactive mode process. It cannot, however, handle input or output from your command files. A "No condition handler found" error message appears in the Session Manager control panel if FileView fails due to an INQUIRE statement in one of the login command procedures.
- FileView runs your tasks as subprocesses; therefore, your process quotas that are depleted by subprocess creation dictate how many FileView tasks you can run simultaneously.

  Before creating a new process, FileView checks these quotas and displays a warning in a dialog box if any are too low. The quota name is included in the message and can be one of the following:

  — ASTLM

  — BIOLM

  — BYTLM

  — FILLM

  — PGFLQUOTA

  — PRCLM

  — TQELM

  The most likely quotas to be consumed are your process limit (PRCLM) and buffered I/O byte count (BYTLM). To run a single task from FileView, your BYTLM quota should be a minimum of 10000. Add an additional 5000 for each task you want to be able to run simultaneously. So, to be able to run five simultaneous tasks, your PRCLM quota must be at least 5, and your BYTLM quota must be at least 30000. Process creation can reduce remaining ASTCNT and BIOCNT by 3, and FILCNT by 2. PGFLQUOTA usage is highly dependent on the task.

  FileView checks these quotas when creating its subprocesses. However, some quotas such as PGFLQUOTA are not consumed until the application is running. Therefore, if several applications are invoked at once, it is possible that PGFLQUOTA will be exhausted once the applications start up, without the error being detected by FileView. In this case, the applications can crash when the quota is exceeded.

When process creation fails due to quota exhaustion, FileView marks the task as Pending in the Work in Progress box until one of the running tasks has completed. The Pending task then becomes Active. If you try to start an additional task after the quota message has been displayed, the task is marked Pending, and the Work in Progress box pops up without a further warning message.

- If you paste a large amount of text into a FileView Task Output box while a text editor is running, the text might appear incorrectly. Press Ctrl/W to correctly display the text.
- If the total length of all the filenames selected in the FileView window (including the device and directory name on each file) exceeds 65535 characters, only a subset of the files are operated on when a verb is selected from a menu.

2.3.5 Session Manager Problems Corrected

V5.2

The following Session Manager problems have been corrected in VMS Version 5.2:

- The Session Manager occasionally created one fewer than the selected number of terminal emulator windows during startup.
- The session manager could abort with an access violation on the second and subsequent attempts to use the print screen function.

2.3.6 Startup Problem

V5.2

If you start DECwindows before starting DECnet, and subsequently start DECnet, you will not be able to create additional terminal emulator windows. To solve the problem, quit the current DECwindows session and log in again.

Note that the initial system boot after installing VMS Version 5.2 brings up DECwindows without starting DECnet, which is exactly the sequence that causes the problem.

2.3.7 SYSGEN Parameter PQL_MPRCLM and Captive Accounts

V5.1

When you run AUTOGEN on a workstation that is running DECwindows, PQL_MPRCLM (the process quota Minimum Process Limit) is set to 8 to allow FileView to function with its subprocesses. Note that this parameter affects only workstations running DECwindows, even in a mixed cluster of workstations and non-workstations.

The *Guide to VMS System Security* recommends that you set the process limit to 0 for a captive account. This prevents a user from accessing DCL when running an application that allows a SPAWN in a captive account. However, Mail no longer allows a SPAWN if the CAPTIVE flag is set in the account record; following this recommendation is unnecessary for captive accounts running Mail only.

If you are setting up a captive account with access to other applications, you should check them to see if SPAWN is allowed.

To override the DECwindows setting, add the following line to your SYS$SYSTEM:MODPARAMS.DAT file:

```
PQL_MPRCLM = 0
```

Edit the SYS$MANAGER:DECW$CHECK_PARAMS.COM file so that it does not check for the setting of this SYSGEN parameter.

2.3.8 Running ULTRIX Applications From A Workstation

V5.1

When a DECnet connection initiated by an ULTRIX client is received by a VMS server, the user name associated with the connection is not the ASCII name of the user (for example, jqpublic) but the user identification (UID) number in ASCII form (for example, 517). Using this example, you can enter either of the following authorization strings to the Security Customization menu to allow user jqpublic access to the VMS server (assume the ULTRIX node name is ultrix):

ultrix::517
ultrix::*

Note that the string using a wildcard character could allow other users to connect to the server and is a dangerous security practice.

2.4 DIGITAL Command Language (DCL) — Notes

The following sections contain information concerning the Digital Command Language (DCL).

2.4.1 DCL Command Verb and Qualifier Length

V5.2

DCL currently checks only the first four characters of command verbs and qualifiers. Because of the continuing growth in the number of VMS products that use DCL command syntax, VMS is considering a change in which four characters may not be enough to identify all verbs or qualifiers. A sufficiently long transition period would precede any such change. Further details would be made available when the transition period begins.

When writing or modifying command procedures, or creating symbols for shorthand interactive use, it is important that you spell out the command syntax correctly. It is also recommended that you spell out the command in its entirety. This would prevent any problems or confusion should the four character restriction be relaxed. This is also a good practice to follow in general, as it helps to comment the command procedure and prevents ambiguity as new or updated products are installed on your system.

2.4.2 ANALYZE/ERROR_LOG Command — New Error-Logging Format

V5.2

The format of the system error messages logged to SYS$ERRORLOG:ERRLOG.SYS has changed in VMS Version 5.2. Error-log messages generated on systems running VMS Version 5.2 cannot be analyzed (using the command ANALYZE /ERROR_LOG) by systems running versions of the VMS operating system prior to Version 5.1-1. Please analyze these messages on systems that are running VMS Version 5.2.

2.4.3 ANALYZE/IMAGE Command

V5.2

Prior to VMS Version 5.2, when an image linked against the system symbol table was analyzed, the ANALYZE/IMAGE command displayed only the values of the system version categories for which the image was originally linked. In VMS Version 5.2, the ANALYZE/IMAGE command displays both the values of the system version categories for which the image was originally linked and the values for the currently running system. This change allows you to identify changes in the system since the image was last linked.

The following example is part of the output produced by analyzing the image SYS$LOADABLE_IMAGES:LOCKING.EXE. Since the current system value is higher than the original image value, the image may have to be relinked to execute properly in the current system.

```
system version array information: (Image Value / Current System Value)
        SYS$K_MEMORY_MANAGEMENT : (1.0 / 1.1)
        SYS$K_PROCESS_SCHED : (1.0 / 1.1)
        SYS$K_SYSGEN : (1.0 / 1.1)
        SYS$K_CLUSTERS_LOCKMGR : (1.0 / 1.1)
        SYS$K_COUNTERS : (1.0 / 1.1)
        SYS$K_STABLE : (1.0 / 1.1)
        SYS$K_MISC : (1.0 / 1.1)
        SYS$K_VOLATILE : (1.0 / 1.1)
        SYS$K_SHELL : (1.0 / 1.1)
```

2.4.4 CANCEL Command

V5.2

With VMS Version 5.2, the CANCEL command is able to cancel wakeup calls for processes on remote nodes in a VAXcluster. When you specify the name of the process for which wakeup requests are to be canceled, you can specify a node name of as many as eight alphanumeric characters in addition to the process name.

For more information about the CANCEL command, see the *VMS DCL Dictionary*.

2.4.5 DEFINE/FORM Command /SHEET_FEED Qualifier — Restriction

V5.0

Do not use the /PAGES qualifier to the PRINT command when submitting jobs to queues on which the DEFINE /FORM/SHEET_FEED command has been issued. When used with the /SHEET_FEED qualifier, the /PAGES qualifier causes the print symbiont to enter an infinite loop. The last page of the document prints repeatedly; the symbiont pauses after each page prints. If you encounter this problem, enter the following commands to stop and restart the queue:

```
$
$
```

2.4.6 DELETE/xxx/LOG Command

V5.2

In previous versions of VMS, no valid qualifiers were documented for the DCL commands that delete queue-related objects (DELETE/CHARACTERISTIC, DELETE/ENTRY, DELETE/FORM, and DELETE/QUEUE). However, these DELETE commands still executed if you specified the /LOG qualifier with the command.

In VMS Version 5.2, /LOG has been made a valid qualifier for these four DELETE commands. If you include the /LOG qualifier with a command to delete a characteristic, entry, form, or queue, you now receive an informational message after the successful processing of your DELETE command.

Also, the error messages displayed in response to an attempt to delete a characteristic, entry, form, or queue using VMS Version 5.2 are more descriptive than the messages displayed in previous versions of VMS.

2.4.7 DIRECTORY/FULL Command

V5.2

With VMS Version 5.2, where applicable, the DIRECTORY/FULL command provides the value of the stored semantics tag as part of the file information returned to the user.

This is the recommended method for determining whether or not a file is tagged. The following example illustrates how the DIRECTORY/FULL command returns the RMS attributes for a DDIF file named X.DDIF:

```
$
   X.DDIF;1                        File ID:  (767,20658,0)
   .
   .
   .
   RMS attributes:     Stored semantics: DDIF
   .
   .
   .
```

For more information about file tags, see the *VMS Version 5.2 New Features Manual*. For more information about the DIRECTORY/FULL command, see the *VMS DCL Dictionary*.

2.4.8 OPEN Command — Problem Negating Qualifiers

V5.0

Currently, if you negate a qualifier using the DCL command OPEN, DCL appears to negate the qualifier when, in fact, it does not. For example, the following two commands are erroneously processed as the same command:

```
$
$
```

Both commands allow shared access to FOO.TMP. In a future release, OPEN qualifiers will be made nonnegatable and an error message will be displayed.

2.4.9 SET ACL Command

The following sections pertain to the DCL command SET ACL.

2.4.9.1 Use of the /LIKE Qualifier

V5.0-1

In Version 5.0-1 of the VMS operating system, use of the /LIKE qualifier with the SET ACL command no longer requires exclusive access to the source object.

2.4.9.2 Wildcard Problem Fixed

V5.2

In VMS versions prior to Version 5.2, if you attempted to use the DCL command SET ACL to delete an Access Control Element (ACE) from a group of files using wildcards, the process aborted if any file in the group either did not include that ACE in its Access Control List (ACL) or did not have an ACL. This problem has now been corrected.

2.4.10 SET DEVICE/[NO]AVAILABLE Command

V5.2

VMS Version 5.2 supports the SET DEVICE/[NO]AVAILABLE command for use with dismounted magnetic tapes as well as dismounted disks. You must dismount the disk or magnetic tape before entering the SET DEVICE/[NO]AVAILABLE command.

For more information about the SET DEVICE/[NO]AVAILABLE command, see the *VMS DCL Dictionary*.

2.4.11 SET HOST/DTE Command

V5.2

SET HOST/DTE uses the VMS terminal driver to provide flow control to other systems. This corrects a resource contention problem in VMS Version 5.0 that occasionally caused workstations to crash. (See the section "SET HOST/DTE Command Causes System Failures on Workstations" in the *VMS Version 5.0 Release Notes*.)

Once SET HOST/DTE has received 100 buffers worth of data, it stops reading from the specified terminal. As soon as the type-ahead buffer is full, the VMS terminal driver sends an XOFF flow control message. Once SET HOST/DTE has displayed most of the data, it starts reading from the terminal again.

Although flow control is enabled by default, it might be undesirable for connection to some systems. For example, on remote connections that do not use XON or XOFF flow control, a SS$_DATAOVERRUN error might be displayed after a long burst of output from another system.

You can disable the flow control feature of SET HOST /DTE by using the logical name RTPAD$BUFFERS to specify the maximum number of buffers that RTPAD uses. You can specify a value as large as 2147483647. If you specify a value that is smaller than the default value of 100 buffers, the default value is used. This controls the amount of virtual memory that RTPAD uses.

For more information about the SET HOST/DTE command, see the *VMS DCL Dictionary*.

2.4.12 SET PROCESS Command

V5.2

With VMS Version 5.2, the SET PROCESS command is able to change execution characteristics associated with processes on remote nodes in a VAXcluster. When you specify the name of the process for which characteristics are to be changed, you can specify a node name of as many as eight alphanumeric characters in addition to the process name.

For more information about the SET PROCESS command, see the *VMS DCL Dictionary*.

2.4.13 SET PROCESS/CPU=[NO]ATTACHED Command No Longer Supported

V5.0

Support for the DCL command SET PROCESS/CPU=[NO]ATTACHED has been removed. This command was a part of asymmetric multiprocessing (ASMP) support designed to help minimize scheduling inefficiencies. It has no counterpart under symmetric multiprocessing (SMP).

2.4.14 SHOW PROCESS Command

V5.2

With VMS Version 5.2, the SHOW PROCESS command is able to display information about processes on remote nodes in a VAXcluster. When you specify the name of the process about which information is to be displayed, you can specify a node name of as many as six alphanumeric characters in addition to the process name.

For more information about the SHOW PROCESS command, see the *VMS DCL Dictionary*.

2.4.15 DCL Lexical Functions

The following sections provide information about DCL lexical functions.

2.4.15.1 F$CONTEXT Lexical Function — Restriction

V5.2

There is a temporary restriction when creating a process context with the DCL lexical function F$CONTEXT concerning the use of the USERNAME and ACCOUNT selection-items.

For values not containing wildcard characters or values containing the single substitution character (%), the selection value for USERNAME must be a blank padded string of 12 characters, like that which you would obtain from an F$GETJPI(" ", "USERNAME") call. Similarly, the ACCOUNT selection value must be a blank padded string of 8 characters. To ensure that your strings are in an accepted form, use the following as a guide:

```
 . . .
$! Initialize a string of blanks
$!
$ blanks = "            "
 . . .
$! Make sure username is 12 characters, blank padded
$!
$ username = f$extract(0,12,username+blanks)
 . . .
```

For values containing the multiple substitution wildcard character (*), the selection value for both ACCOUNT and USERNAME must not be blank padded and must end with *, for example, *SMI**, not *SMI** . To accomplish the equivalent of *BR*N*, use the string *BR*N* * (Adding the space character between the *N* and the * avoids selecting strings such as "BRANNER").

These restrictions will be removed in a future release of VMS.

2.4.15.2 F$FAO Lexical Function

V5.2

Case statement directives have been added that allow different text to be included depending on the value of a given argument. The case statement is useful for inserting irregular plurals.

For more information about the F$FAO lexical function, see the *VMS DCL Dictionary*.

2.4.15.3 F$PID Lexical Function

V5.2

The F$PID lexical function returns all the PIDs of processes in your group (if you have GROUP privilege) or on the system (if you have WORLD privilege), using criteria set up by the F$CONTEXT lexical function, if any. The F$CONTEXT function also enables the F$PID function to retrieve processes from any node in a VAXcluster.

For more information about the F$PID and F$CONTEXT lexical functions, see the *VMS DCL Dictionary*.

2.4.15.4 F$TYPE Lexical Function

V5.2

The F$TYPE lexical function returns the data type of a symbol produced by a call to the F$PID or F$CONTEXT lexical functions, if any.

For more information about the F$TYPE, F$PID, and F$CONTEXT lexical functions, see the *VMS DCL Dictionary*.

2.4.15.5 F$VERIFY Lexical Function — Symbol Substitution

V5.0

DCL does not normally perform symbol substitution for symbols that appear after a comment character. Symbol substitution does occur for the F$VERIFY lexical function, however, even if it appears after a comment character and the F$VERIFY lexical function executes. This is the only lexical function and the only symbol substitution that can be processed when the comment character is present. An example of this is as follows:

```
$! 'F$VERIFY(1)'
```

In this example, symbol substitution would be performed for the F$VERIFY function.

This feature will be documented with a future version of the VMS operating system.

2.5 DIGITAL Standard Runoff (DSR)

V5.2

In DIGITAL Standard Runoff, the maximum length of a variant name (/VARIANT=string qualifier) has been changed from 15 characters to 31.

2.6 Directory Entries — Primary as Opposed to Alias

V5.0

VMS Version 5.0 distinguishes between primary directory entries (the directory entries created when files are created) and alias directory entries (entries created with the DCL command SET FILE/ENTER or similar operations). Every file has a back link that identifies its primary directory entry.

Following is a summary of the directory entry changes in VMS Version 5.0:

- ANALYZE/DISK_STRUCTURE no longer reports "invalid back link" errors on files with alias directory entries.
- Only primary directory entries are subject to the special directory entry protection rules described in the *Guide to VMS System Security*.
- When a primary directory entry is deleted, the file contents are deleted. Any remaining alias directory entries are left pointing to a nonexistent file.
- When an alias directory entry is deleted, either by the DELETE or the SET FILE/REMOVE command, only the directory entry is removed; the file contents remain.
- When a primary directory entry is removed with a SET FILE/REMOVE command, the file remains but no longer has a back link. From then on, alias directory entries are treated as if they were primary entries.
- If a new directory entry is made for a file with no back link, for example by SET FILE/ENTER or RENAME, the new directory entry becomes the primary entry.

A problem exists in VMS Version 5.0 in the treatment of multiple directory entries for the same file when they are located in the same directory file. When a file or directory entry is being deleted, the DELETE command always locates the directory entry of the file that occurs first in the directory, not necessarily the one named in the command. Consider the following sequence of commands:

```
$
$
$
```

In this case, rather than removing the alias entry Y.Y, the DELETE command removes the primary entry X.X and deletes the file contents. This problem will be corrected in a future release of the VMS operating system.

2.7 EDT Editor — Notes

The following notes pertain to the EDT editor.

2.7.1 EDT Problem Renaming TMP Files

V5.0

Because VMS Version 5.0 requires the owner of a file to have delete access to that file in order to allow renaming, EDT will not function properly if the original file does not have delete access granted to the owner.

The problem occurs when you edit a file and EDT creates a TMP file with the same protection mask as your source file. When you try to exit, EDT tries to rename the TMP to the next higher version of the source file and subsequently fails.

This problem will be corrected in a future release of the VMS operating system. You can avoid the problem by making sure you have delete access to the file before you begin your editing session. Also, you may use the EDT command WRITE followed by the EDT command QUIT. WRITE creates a new file without renaming, and QUIT exits without saving (by renaming).

2.7.2 EDT Problems Corrected

V5.2

The following problems have been corrected in the EDT editor for VMS Version 5.2:

- In certain cases the EXIT command was incorrectly parsed and created an erroneous new file name. For example, if the user typed EX9T, EDT exited, creating a file named 9T.
- The SET ENTITY WORD command did not accept lower case arguments. The command SET ENTITY WORD “a” would fail.
- The DELETE command followed by an UNDELETE command did not work in all cases.
- If a buffer that had a SELECT command active was cleared, recreated, and reset, a fatal internal error was generated.

- The SUBSTITUTE command with the ALL qualifier sometimes failed on the first substitution. This happened if the string used with the ALL qualifier was at or near the end of the line, and the length from the beginning to the end of the line was shorter than the length of the substitute search string.
- The message "must select full lines" was displayed while in NOKEYPAD mode, if a SELECT was done followed by a LINE mode command.
- EDT did not understand search lists correctly. If the user edited a file that did not exist yet (creating a new file), EDT created the new file where the LAST entry of the search list specified instead of where the FIRST entry of the search list specified.
- In the EDT initialization file, comments that included a CTRL/Z caused the command file to exit and not finish the initialization.
- A macro that cleared itself caused an access violation.
- If the SET ENTITY command caused EDT to backup to the previous line, EDT lost its place within the buffer.
- Doing a FILL command with SELECT going backwards caused an access violation.
- Multiple CTRL/C's in the journal file were not processed properly.
- The SET LINES command did not do proper lower bounds checking. It was possible to issue SET LINES 0 without any error reported.
- LINE mode INSERT could parse the command and data incorrectly, if the line was greater than 255 characters.
- In callable EDT, when a file was not found the return status was success instead of EDT$_INPFILNEX.

2.8 VMS Mail Utility — Folder Name Parameter Now Supports Mixed Cases

V5.1

The VMS Mail Utility folder name parameter now supports mixed cases when you enclose the name within double quotation marks. Starting with Version 5.0, when you specified a folder name, it was changed to all capitals even if you used quotation marks. Before Version 5.0, if you quoted the folder name, it was left in mixed cases. Version 5.1 restores the Version 4.X capability and supports mixed cases in quoted folder names.

A folder name can be 1 to 39 characters in length. Valid characters for folder names are A through Z, a through z, dollar sign ($), underscore (_), hyphen (-), and 0 through 9. To retain mixed cases, enclose the folder name within quotation marks.

2.9 Process Identification (PID) — All Significant Digits Must Be Specified

V5.2

All system services that control processes or obtain information about processes use **pidadr** as the first argument and **prcnam** as the second argument. These two arguments are also used to reference remote processes.

The process identification (PID) is unique for each process in the cluster. To reference a process on another node in the cluster, specify its PID as the **pidadr** argument. Prior to VMS Version 5.2, you could abbreviate a PID by omitting the high-order field. However, because the high-order field is now used to identify the node, you must specify all the significant digits of a PID; you can omit leading zeros.

For example, prior to VMS Version 5.2, you could specify the following to stop the process with the PID 48400136:

```
$
```

With VMS Version 5.2, you must specify the following, or you will receive a status code message warning you that this is a nonexistent process (SS$_NONEXPR):

```
$
```

The process name has also been extended to reflect clusterwide accessibility. To access information about a remote process, the node name must be prefixed to the process name. For example, to reference the process BATCH_69 on node ATHENS, use the name ATHENS::BATCH_69.

This change in process naming has the following implications:

- Process name strings can be up to 23 characters long.
  - — 6 characters for the node name
  - — 2 characters for the double colon (::) that follows the node name
  - — 15 characters for the process name
- A local process name can look like a remote process name. Therefore, if you specify ATHENS::SMITH, the system checks for a process named ATHENS::SMITH on the local node before checking node ATHENS for a process named SMITH.

2.10 Sort/Merge Utility

V5.2

The following are changes and enhancements to the Sort/Merge Utility:

- The specification file keyword /FIELD has been enhanced to allow you to define constants. In the following example, the value of the constant **zip_code** is defined as 03060.

  ```
  /FIELD=(NAME=zip_code,SIZE:5,VALUE="03060",CHARACTER)
  ```

- The performance of the Sort/Merge Utility has been improved by 26%. A major factor in this improvement is that the Sort/Merge Utility now allows asynchronous reads on scratch files and input files.

- SORT scratch files are entries in the SYS$SCRATCH directory, rather than temporary files as defined by the VMS Record Management Services (RMS).
- Specification file fields that are defined as floating now allow the use of a decimal point. For example:

```
/COND=(NAME=A,TEST=(B EQ 150.40))
```

2.11 Symbol Names — Caution

V5.0

When making symbol name assignments, Digital recommends you use caution when assigning a symbol name that is already a DCL command name. This can cause unnecessary confusion particularly for inexperienced users. Digital especially discourages the assignment of symbols such as IF, THEN, ELSE, and GOTO by which you may affect the interpretation of command procedures.

3 System Manager Release Notes

This chapter includes information about VMS Version 5.2 that is of interest to the system manager.

For information about the new features included in VMS Version 5.2, see the *VMS Version 5.2 New Features Manual*.

3.1 *VMS V5.2 Upgrade and Installation Procedures* — New Manual

V5.2

For VMS Version 5.2, the VMS document set contains a new manual entitled, *VMS V5.2 Upgrade and Installation Procedures*. This manual contains the following:

- The VMS Version 5.2 upgrade procedure
- Information about the VMS Version 5.2 installation procedure, including DECwindows software installation
- Release notes about various VAX computers

Read this manual before you install, upgrade, or use VMS Version 5.2.

3.2 Installation and Upgrade Information

The following sections contain additional information relating to your VMS installation and upgrade.

3.2.1 Layered Products Information

V5.2

Because of the way the VMS Version 5.2 upgrade procedure is designed, you should not have to reinstall most layered products after the upgrade. However, you must reinstall certain layered products because of product-specific installation procedures. For example, you must reinstall products that create directories synonymous with system directories and products that use VMS-defined data structures. If a product is available (see the *VMS Version 5.2 Upgrade and Installation Procedures*), yet exhibits unexpected behavior once Version 5.2 is running, check the current version of the VMS release notes for layered product cautions. If problems persist, contact your Digital support representative.

Table 3–1 lists the layered products that will be supported sometime after the release of VMS Version 5.2. Contact your Digital representative for specific release dates.

Table 3–1 Layered Products To Be Supported After VMS Version 5.2

Product Name	Version Number
ALL-IN-1 System for Sales and Marketing	1.2
VAX DY32	3.0
VAX PCL	2.0
VS11-VAX Driver	2.5

3.2.2 Upgrade Procedure Driver Version Mismatch Errors

V5.0

The device drivers for some hardware options are shipped with the software kits that support those options. For example, the drivers for VAX workstation hardware and certain communications devices are on separate kits, the drivers are not in the VMS operating system distribution kit.

When upgrading a workstation or other system with separately-shipped drivers, error messages about drivers are returned while the system boots. These messages report that the system version of the driver does not match the current version. For example, when you upgrade a VAXstation II/GPX, the following message may be returned:

```
%SYSGEN-E-SYSVERDIF, system version mismatch - reassemble and relink driver
-SYSGEN-I-DRIVENAM, driver name is VADRIVER
```

These messages disappear when you install a compatible version of the software, (in this example, the VMS Workstation Software (VWS)). Note that the workstation or other devices cannot be used until the correct drivers are available.

3.3 AUTOGEN Command Procedure — Notes

This section describes changes and problems with the AUTOGEN command procedure.

3.3.1 AUTOGEN Aborts During GENFILES Phase

V5.0

AUTOGEN may abort during the GENFILES phase displaying the following error message:

```
%SYSTEM -F-NOSUCHFILE, no such file
```

This problem occurs when an invalid file specification is passed to AUTOGEN by the DCL command SHOW MEMORY. This can occur in the following two ways:

- The logical name SYS$SYSROOT is defined improperly. You can determine this by entering the following DCL command:

```
$
   "SYS$SYSROOT" = "ddcu:[SYSE.SYS0.]" (LNM$SYSTEM_TABLE)
```

As a result, the file specification ddcu:[SYSE.SYS0.SYSEXE]SWAPFILE.SYS;1 is incorrectly produced. You can resolve this problem by defining SYS$SYSROOT to its proper value and then reinvoking AUTOGEN.

- The preexisting system page or swap file has an invalid back pointer. You can verify this by entering the command ANALYZE/DISK_STRUCTURE. Subsequently, SHOW MEMORY produces the file specification ddcu:[ ]SWAPFILE.SYS;1.

  You can correct this problem by repairing the file using the Analyze/Disk_Structure Utility command ANALYZE/DISK_STRUCTURE/REPAIR and then invoking AUTOGEN again.

3.3.2 Changing Window System with AUTOGEN

V5.2

In VMS Version 5.2, if you wish to switch from VWS to DECwindows (or vice versa), you must reboot your system twice in order for AUTOGEN to set system parameters appropriately.

Note: When switching window systems, first delete all AUTOGEN FEEDBACK data files. This is necessary because DECwindows and VWS FEEDBACK parameters are not compatible. Prior to executing AUTOGEN NOFEEDBACK, enter all layered product and third-party software parameter requirements into the file MODPARAMS.DAT.

The command sequence is as follows:

1 First delete all AUTOGEN FEEDBACK files:

```
$
$
SYSGEN>
SYSGEN>
SYSGEN>
```

2 Reboot the system using the following command:

```
$
```

3 After the reboot, execute AUTOGEN using the following command:

```
$
```

4 The system will automatically reboot using the newly generated DECwindows or VWS parameters.

Regular execution of AUTOGEN FEEDBACK will ensure that system parameters reflect user load. After the system has been running a typical load for at least 24 hours, invoke AUTOGEN FEEDBACK as follows:

```
$
```

Note: If you wish to change window system more than once, save copies of your system parameter files (SYS$SYSTEM:VAXVMSSYS.PAR) for both DECwindows and VWS. You can subsequently change window system by a conversational boot using the appropriate parameter file. These saved versions should be kept up to date

(that is, after running AUTOGEN through SETPARAMS, save the newly generated parameter file SYS$SYSTEM:VAXVMSSYS.PAR).

3.3.3 Executing from GENPARAMS Phase — Problem

V5.1

When executing AUTOGEN from the GENPARAMS phase using an old (pre-release) SYS$SYSTEM:PARAMS.DAT file, AUTOGEN reports two undefined symbols: DEBNA_CNT and TK_CNT. If this problem occurs, rerun AUTOGEN from GETDATA or SAVPARAMS, which will create a new version of SYS$SYSTEM:PARAMS.DAT.

This problem will be corrected in future releases of the VMS operating system.

3.3.4 Feedback Mechanism — Additional Data Files Used

V5.0

AUTOGEN now uses two new data files, SYS$SYSTEM:AGEN$ADDHISTORY.TMP and AGEN$ADDHISTORY.DAT. These files are used in conjunction with the feedback mechanism to avoid accumulating the values of ADD_ symbols found in MODPARAMS.DAT and VMSPARAMS.DAT from one invocation of AUTOGEN to the next. These files should not be deleted or modified.

3.3.5 Hexadecimal Values Processed Correctly in MODPARAMS.DAT

V5.0

Parameters set to hexadecimal values in MODPARAMS.DAT and that correspond to a negative decimal value are now processed correctly. For example, the MODPARAMS.DAT record "SMP_CPUS=%XFFFFFFFF" results in the parameter SMP_CPUS being set to -1.

3.3.6 Mechanism to Control MSCP Server Buffer Size Is Obsolete

V5.0

Prior to Version 5.0, a system manager could define internal AUTOGEN symbols of the form MSCP_* (for example, MSCP_BUFFER) in MODPARAMS.DAT to control the amount of nonpaged pool allocated to the MSCP server for buffer space. AUTOGEN recognized these symbols and set up the corresponding bit fields in the parameters VMS5 and VMS6. This was necessary because the MSCP server must be loaded early in the boot sequence on local area VAXcluster boot nodes.

Beginning with Version 5.0, the method used to load the MSCP server has changed. The early loading of the server on local area VAXcluster boot nodes is still required; it is accomplished using the new SYSGEN parameter MSCP_LOAD, which can have either of the following values:

0	Do not load the server (default)
1	Load the server

If MSCP_LOAD is 1, the amount of nonpaged pool allocated to the server's buffer is controlled by another new SYSGEN parameter, MSCP_BUFFER, which is expressed in pages (128 pages is the default).

If you have defined any of the old AUTOGEN internal symbols in MODPARAMS.DAT, they should be removed, because they may conflict with the new SYSGEN parameters.

For more information, refer to the *VMS System Generation Utility Manual*.

3.3.7 OLDSITE Mechanism Is Obsolete

V5.0

The OLDSITE mechanism for propagating parameter values is now obsolete. Make sure that SYS$SYSTEM:MODPARAMS.DAT contains a record for any SYSGEN parameter that AUTOGEN does not calculate that requires a value different from its default.

3.3.8 Prefix ADD_ Can No Longer Subtract Values for Parameters Generated by FEEDBACK

V5.2

For FEEDBACK generated parameters *only*, AUTOGEN now disallows subtracting parameter values by means of the prefix ADD_ in MODPARAMS.DAT. For example, the following command has no effect on the parameter GBLPAGES:

```
ADD_GBLPAGES = -20
```

Likewise, if you replace the value of a parameter with the ADD_ prefix with a smaller value, the net subtraction is also disallowed. For example, replacing:

```
ADD_GBLPAGES = 200
```

with the following:

```
ADD_GBLPAGES = 100
```

has no effect on the parameter GBLPAGES.

Note: AUTOGEN FEEDBACK has a mechanism in place to decrease parameter values based on the system load. Users wishing to decrease FEEDBACK generated parameters further may do so by entering a MAX_ prefix for that parameter in MODPARAMS.DAT or by explicitly setting the parameter value (parameter = parameter-value). Digital strongly recommends that these entries be removed from MODPARAMS.DAT after a single execution of AUTOGEN, so that FEEDBACK can resume calculating the parameter.

3.3.9 System Files Not Marked for NOBACKUP

V5.0

AUTOGEN warns you if it finds an existing page, swap or dump file that is *not* marked for NOBACKUP because it is probably unnecessarily increasing your disk backup time. If you receive this warning, do the following:

1. Make sure that at least one viable page file remains.
2. Rename the file in question so that it will not be installed during the next boot.

3 Shut down and reboot the system.

4 Use the SET FILE/NOBACKUP command to mark the file NOBACKUP.

5 Rename the file to its original name.

6 Shut down and reboot the system.

3.3.10 Swap File Size Changes

V5.0

The VMS memory management swap file allocation algorithm has been changed significantly. A swap slot is allocated only when a process is selected as an outswap candidate. The swap slot need not be virtually contiguous or contained in one file. This means that swap file requirements will be significantly less than for previous versions of the VMS operating system.

As a result, modifications have been made to the sizing algorithm for AUTOGEN's swap file. This allows AUTOGEN to produce much smaller swap files, typically only one quarter of what would have been produced under Version 4.n. If you have overridden AUTOGEN's swap file calculation by defining the symbol SWAPFILE=0 in MODPARAMS.DAT, remove this symbol definition in order to let AUTOGEN create a smaller swap file.

3.4 Backup Utility — Notes

This section contains release notes describing the VMS Backup Utility (BACKUP).

3.4.1 Data Overwrite Problem with MOUNT/NOUNLOAD

V5.0-1

In Version 5.0 of the VMS operating system, if you mounted a tape using the MOUNT/NOUNLOAD command and then used the tape in a backup operation that required multiple tapes, the Backup Utility could not unload the first tape. Instead of prompting you to load a new tape, the Backup Utility continued processing.

Note: In a save operation, this problem caused data on the first tape to be overwritten by data that was intended for the second tape.

Version 5.0-1 of the VMS operating system fixes this problem. The Backup Utility can now successfully unload the tape, regardless of how it was mounted.

3.4.2 Errors Using Degaussed Tapes on HSC Drives

V5.0-1

In VMS Version 5.0, a backup operation to a degaussed tape on an HSC drive failed and displayed the following error message:

```
BACKUP-F-LABELERR, error in tape label processing on 'device'
SYSTEM-?-NOMSG Message number 8026.
```

VMS Version 5.0-1 fixes this problem.

3.4.3 Errors Using TU81–Plus Tape Drives

V5.0-1

When the Backup Utility writes to a faulty magnetic tape, it recovers by positioning the tape after the bad block and rewriting the data on a good block. However, in VMS Version 5.0, a tape written in this manner could not be read by a TU81–Plus tape drive. When a TU81–Plus tape drive attempted to read the tape, the following error message resulted:

```
%BACKUP-E-FATALERR, fatal error on MUA0:[]TEST.;
-SYSTEM-F-TAPEPOSLOST, magnetic tape position lost
%BACKUP-I-SPECIFY, specify option (QUIT, CONTINUE or RESTART)
```

Choosing the CONTINUE or RESTART option caused the error to recur.

VMS Version 5.0-1 fixes this problem. When a backup operation encounters a bad block and recovers, a recoverable media error is reported in the error log. A tape with a bad block can be read by a TU81–Plus tape drive.

3.4.4 Forced Error Handling

V5.0

Most VMS utilities and DCL commands treat a forced error flag as a fatal error. For example, if you use the DCL command COPY to move a file that contains a block with the forced error flag, the resulting error causes the operation to terminate.

The Backup Utility, however, is designed to continue in the presence of almost all errors, including forced errors; BACKUP continues to process the file, creating a new copy of the file in the output save set. An error message indicating the forced error is displayed, but the forced error is not present in the new copy of the file that is being created. Subsequent use of the new file (for example, in a restore operation) will indicate no errors. Thus, data that was formerly marked as bad with the forced error flag may be accidentally propagated and now seem correct.

System managers (and other users of BACKUP) should assume that forced errors reported by BACKUP signal degradation of the data in question and should act accordingly. The safest procedure is to replace the file containing the forced error with a good copy of the file from a previous BACKUP operation.

For more information on Digital Storage Architecture (DSA) and forced errors, see the *VMS I/O User's Reference Manual: Part I*.

3.4.5 Magnetic-Tape Save Sets — Restriction

V5.2

Only magnetic-tape save sets created with the /INTERCHANGE qualifier can be copied successfully to disk using the DCL command COPY. In general, Digital does not recommend using the DCL command COPY to copy magnetic-tape save sets to disk. This is because BACKUP's default error correction methods will not be used, and the file created with the COPY command may contain inconsistent data.

Digital recommends that you process magnetic-tape save sets with the BACKUP command.

3.4.6 Reading Multiple Save Sets from a TU81–Plus Tape Drive

V5.2

Under some circumstances, when reading multiple backup save sets from a TU81 PLUS tape drive, the drive can become misaligned and return a "position lost" status, thus aborting the backup. This can occur while attempting to read a save set after having just read a preceding save set on the same volume. This is a timing problem which will be addressed in a future fix for the TU81 PLUS controller microcode. In the meantime, you can use the following temporary solutions:

- Before entering your BACKUP command, reposition the tape with the following DCL command:

  ```
  $
  ```

- Use $BACKUP/REWIND to read the second or subsequent save set on the tape. Note that this rewinds the tape to the beginning (BOT) and then searches forward for the specified save set.

3.4.7 Standalone BACKUP Notes

This section contains release notes describing standalone BACKUP.

3.4.7.1 Additional Privilege Required to Execute STABACKIT.COM

V5.0

The *VMS Backup Utility Manual* describes how to use the SYS$UPDATE:STABACKIT.COM command procedure to create a standalone BACKUP kit on a disk. This section states that the user privileges BYPASS, CMKRNL, CMEXEC, LOG_IO, SYSNAM, VOLPRO, and OPER (or the user privilege SETPRV) are required to execute STABACKIT.COM. The VMS operating system also requires that you have either the user privilege PHY_IO or SETPRV to execute STABACKIT.COM.

3.4.7.2 Booting Standalone BACKUP on a VAX 8820, 8830, 8840

V5.2

The *VMS Installation and Operations: VAX 8820, 8830, 8840* manual specifies the following two commands for booting standalone BACKUP from the console fixed disk:

```
PS-CIO-0>
PS-CIO-0>
```

Use the following commands instead:

```
PS-CIO-0>
PS-CIO-0>
```

3.4.7.3 Kit on RX33 Diskette Contains Three Volumes

V5.2

Prior to VMS Version 5.2, the command procedure STABACKIT.COM treated the RX33 diskette as a large disk, which meant that the complete standalone BACKUP kit was contained on one volume.

With VMS Version 5.2, the files required for standalone BACKUP no longer fit on a single RX33 diskette. The RX33 diskette is treated as a small console volume, much like the RX50 diskette. For reasons of consistency, future expansion, and the ability to build the system and application kits separately, you need three RX33 diskettes to build the full kit. You can build the application and system kits separately. The procedure STABACKIT.COM automatically initializes each diskette and you can use the data reliability options on each diskette.

3.4.8 Using BACKUP with Compound Document Files

V5.1

Normal usage of BACKUP correctly preserves all file attribute information for compound document (for example, DDIF) files. However, BACKUP/INTERCHANGE fails to preserve the semantics attribute. As a workaround, DDIF files restored from BACKUP/INTERCHANGE save sets can be relabeled as DDIF files using the command:

```
$
```

3.5 Batch/Print Facility — Notes

The following sections contain information pertinent to the Batch/Print facility of VMS.

3.5.1 Generic Queue Restriction Now Enforced

V5.0-2

The VMS Batch/Print Facility is designed to allow one execution queue to be the target of no more than eight generic queues. In Versions 5.0 and 5.0-1 of the VMS operating system, this restriction was not enforced, resulting in unpredictable behavior if more than eight generic queues directed jobs to one execution queue.

Beginning in Version 5.0-2 of the VMS operating system, this restriction is enforced.

3.5.2 Job Controller Now Deallocates Virtual Memory

V5.0-2

In Versions 5.0 and 5.0-1 of the VMS operating system, the job controller failed to deallocate all of the virtual memory it allocated to process a batch or a print job request. This sometimes caused the job controller process to abort with an "insufficient virtual memory" error after processing several thousand jobs.

Version 5.0-2 of the VMS operating system corrects this problem.

3.5.3 Print Symbiont Notes

The following sections pertain to the print symbiont.

3.5.3.1 Symbiont Working Set Purge Less Frequent

V5.2

In previous versions of VMS, when the print symbiont finished processing a file, it purged its working set if no other queue it supported contained a job that was printing. For example, the command PRINT FILE.TEXT/COPIES=5 caused a print symbiont supporting only one queue to purge its working set five times.

In VMS Version 5.2, the print symbiont process purges its working set only if none of the queues supported by that symbiont contains a non-pending job within a **purge-delay** time interval following completion of any file it processes. The purge-delay time interval is currently defined as approximately five minutes.

3.5.3.2 Print Symbiont Virtual Memory Deallocation

V5.0-1

In previous versions of the VMS operating system, the amount of virtual memory allocated to the print symbiont process increased in relation to the number of print jobs handled by that symbiont. This problem occurred because virtual memory was being allocated in several symbiont routines without subsequent deallocation.

In Version 5.0-1 of the VMS operating system, virtual memory is deallocated in these symbiont routines. This prevents the continual growth of the print symbiont's virtual page count under normal operation.

3.5.4 START/QUEUE/MANAGER/BUFFER_COUNT Command Enhancement

V5.2

In VMS Version V5.0, the /BUFFER_COUNT qualifier of the START/QUEUE/MANAGER command has been enhanced to accept a value greater than the previous limit of 127 local buffers. Increased use of local buffers for the queue file by each job controller in a cluster may improve overall batch/print performance by reducing the disk I/O at the expense of memory consumption, locking activity, and CPU cycles. This can be important for a system that supports many queues and whose workload is batch or print job intensive.

For VMS Version 5.2, the enqueue limit and working set quotas for the job controller process can support up to 1500 local buffers, provided that memory is available for the system to dynamically extend the job controller's working set size. If the job controller's working set cannot be extended sufficiently, excessive page faulting will occur. If free memory is limited, the local buffer value should not exceed 300.

3.5.5 START/QUEUE/TOP_OF_FILE Command Improvement

V5.2

In previous versions of VMS, an unwanted line feed was printed at the top of the first page of output following a START/QUEUE/TOP_OF_FILE command. VMS Version 5.2 corrects this problem.

3.5.6 Unsynchronized Cluster Time Affects SUBMIT/AFTER Command

V5.0

In a VAXcluster environment, a batch job submitted to execute at a specified time may begin execution a little before or after the requested time. This occurs when the clocks of the member systems in the VAXcluster environment are not synchronized. For example, a job submitted using the DCL command SUBMIT/AFTER=TOMORROW may execute at 23:58 relative to the host system's clock.

This problem can occur in a cluster even if a job is run on the same machine from which it was submitted, because the redundancy built into the batch/print system allows more than one job controller in the cluster to receive a timer asynchronous system trap (AST) for the job and, thus, to schedule it for execution. Moreover, this behavior is exacerbated if the batch job immediately resubmits itself to run the next day using the same SUBMIT command. This can result in having multiple instances of the job executing simultaneously because TOMORROW (after midnight) may be only a minute or two in the future.

A solution to this problem is to place the SUBMIT command in a command procedure that begins with a WAIT command, where the delta time specified in the WAIT command is greater than the maximum difference in time between any two systems in the cluster. Use the SHOW TIME command on each system to determine this difference in time.

The cluster time can be kept in synchronization by periodic execution of the DCL command SET TIME/CLUSTER. This will recalibrate the individual system times.

3.6 Bootstrapping Issues

This section contains release notes describing the bootstrapping process.

3.6.1 Guidelines for Configuring Large VAXcluster Systems

V5.2

With VMS Version 5.2, the number of CPUs supported in local area and Mixed-Interconnect VAXcluster systems is increased from 42 to 96.

This section provides guidelines for configuring large clusters (approximately 30 CPUs or more) and describes procedures that you may find helpful. Some recommendations may also be appropriate for clusters with fewer than 30 CPUs. Topics include the following:

- Configuring disk server Ethernet adapters and memory
- Configuring system disks
- Rebuilding cluster disks
- Defining the VAXcluster alias
- Running AUTOGEN with FEEDBACK
- Adding CPUs to an existing VAXcluster system
- Setting up a new large VAXcluster system

3.6.1.1 Configuring Disk Server Ethernet Adapters and Memory

V5.2

Because disk serving activity in a large local area or Mixed-Interconnect VAXcluster system can generate a substantial amount of I/O traffic on the Ethernet, boot and disk servers should use the highest-bandwidth Ethernet adapters that you have available. In addition, a large local area or Mixed-Interconnect cluster should include multiple boot and disk servers to enhance availability and to distribute I/O traffic over several Ethernet adapters.

Relatively little memory is required to serve disks. Even busy boot and disk servers probably require no more than one-quarter to one-half megabyte of physical memory for disk-serving activity. However, if boot and disk servers must also support timesharing users or run batch queues for the cluster, the servers should be configured with memory appropriate for those additional tasks.

3.6.1.2 Configuring System Disks

V5.2

Depending on the number of CPUs to be included in a large cluster, you must evaluate the tradeoffs involved in configuring a single system disk or multiple system disks.

While a single system disk is easier to manage, a large cluster may require more system disk I/O capacity than a single system disk can provide.[1] To achieve satisfactory performance levels, multiple system disks may be needed. However, you should recognize the increased system management efforts involved in maintaining multiple system disks.

Concurrent User Activity

In clusters with many workstation satellites, the amount and type of user activity[2] on those satellites influences system disk load and therefore the number of satellites that can be supported by a single system disk. For example, if many users are active or run multiple applications simultaneously, the load on the system disk can be significant. Conversely, in an environment where few users are simultaneously active, or where most users run a single application for extended periods, a single system disk might support a large number of satellites.[3]

This situation is similar to the traditional timesharing model, because the probability is low that most users are simultaneously active at any given time. Thus, while a VAXcluster system can be configured assuming that all users are constantly active, a smaller system can be configured for more typical working conditions. The tradeoff is between a larger VAXcluster system that handles rare peak loads without performance degradation, and a smaller one that handles all activity, but suffers some performance degradation during peak load periods.

[1] Consider using the optional VAX Volume Shadowing Software product to increase disk I/O capacity. For more information on VAX Volume Shadowing Software, see the *VAX Volume Shadowing Manual*.

[2] For example, any active batch job or other task created on the workstation by or for the user.

[3] These environments, however, often direct significant numbers of I/O requests to application data disks.

One difference from the traditional timesharing model should be noted. In a timesharing system, the most important shared resource is the processing power of the CPU. But because each workstation user in a VAXcluster system has a dedicated CPU, running large compute-bound jobs on a workstation does not significantly affect other clustered CPUs.

For clustered workstations, the critical shared resource is the disk server. Thus, if a workstation user runs even a small I/O-intensive job, its effect on other CPUs sharing the disk server may be noticeable.

Concurrent Booting Activity

One of the rare times when all clustered CPUs are simultaneously active is during a cluster reboot—for example, after a power failure. All satellites are waiting to reload, and as soon as a boot server is available, they begin to boot in parallel. This booting activity places a significant I/O load on the system disk or disks.

For example, Table 3–2 shows system disk I/O activity and elapsed time until login for a single satellite with minimal startup procedures when it is the only one booting.

Table 3–2 System Disk I/O Activity and Boot Time for Single Satellite

Total I/O Requests to System Disk	Average System Disk I/O Operations per Second	Elapsed Time until Login (Minutes)
4200	6	12

Table 3–3 shows system disk I/O activity and times elapsed between boot server response and login for various numbers of satellites booting from a single system disk.[4] The disk in these examples has a capacity of 40 I/O operations per second.

Table 3–3 System Disk I/O Activity and Boot Times for Multiple Satellites

Number of Satellites	I/Os per Second Requested	I/Os per Second Serviced	Elapsed Time until Login (Minutes)
1	6	6	12
2	12	12	12
4	24	24	12
6	36	36	12
8	48	40	14

(continued on next page)

[4] The numbers in the tables are fabricated and are meant to provide only a generalized picture of booting activity. Elapsed times until login on satellites in any particular cluster depend on the complexity of the site-specific system startup procedures. CPUs in clusters with many layered products or site-specific applications require more system disk I/O operations to complete the boot procedure.

Table 3–3 (Cont.) System Disk I/O Activity and Boot Times for Multiple Satellites

Number of Satellites	I/Os per Second Requested	I/Os per Second Serviced	Elapsed Time until Login (Minutes)
12	72	40	21
16	96	40	28
24	144	40	42
32	192	40	56
48	288	40	84
64	384	40	112
96	576	40	168

While the elapsed times shown in Table 3–3 do not include the time required for the boot server itself to reload, they illustrate that the I/O capacity of a single system disk can be the limiting factor for cluster reboot time.

Note that you can reduce overall cluster boot time by configuring multiple system disks and distributing system roots for clustered CPUs evenly across those disks. This technique has the advantage of increasing overall system disk I/O capacity but the disadvantage of requiring additional system management effort. For example, layered product installation or VMS operating system upgrades must be repeated once for each system disk.

In a Mixed-Interconnect VAXcluster system with HSC-connected disks, you can use VAX Volume Shadowing Software to increase the I/O capacity of a single system disk. Installations or updates need only be applied once to a volume-shadowed system disk. For clusters with substantial system disk I/O requirements, you can use multiple system disks, each configured as a shadow set.

Boot Time Considerations

When configuring a VAXcluster system for minimum boot times, consider the following:

- Consequences of having the workstations unavailable during a cluster reboot
- Hardware costs of additional disk drives
- Cost of VAX Volume Shadowing Software if needed
- System management effort required to maintain multiple system disks
- The probability of power interruptions

Note: Sites with stringent demands for system availability should investigate power conditioning options to reduce the probability of power interruptions.

Moving High-Activity Files off System Disks

To reduce I/O activity on system disks, you can move page and swap files for clustered CPUs off system disks, and you can set up page and swap files for satellites on the satellites' local disks, if such disks are available. You specify the sizes and locations of page and swap files when you run the command procedure CLUSTER_CONFIG.COM to add CPUs.

You should also move off the system disk such high-activity files as the following:

SYSUAF.DAT	NETPROXY.DAT	RIGHTSLIST.DAT
JBCSYSQUE.DAT	ACCOUNTNG.DAT	VMSMAIL_PROFILE.DATA

For instructions on specifying the location of the files, refer to the *VMS VAXcluster Manual*.

Controlling Dump File Size and Creation

Whether you intend to use a single system disk or multiple disks, you should plan a strategy to manage dump files. Dump file management is especially important for large clusters with a single system disk.

In the event of a software-detected system failure, each clustered CPU normally writes the contents of memory to a full dump file on its system disk for analysis. By default, this full dump file is the size of physical memory plus a small number of pages. If system disk space is limited (as is probably the case if a single system disk is used for a large cluster), you may want to specify that no dump file be created for satellites, or that AUTOGEN create a selective dump file. The selective dump file is typically 30% to 60% of the size of a full dump file.

You can control dump file size and creation for each clustered CPU by specifying appropriate values for the AUTOGEN symbols DUMPSTYLE and DUMPFILE in the CPU's MODPARAMS.DAT file. Dump files are specified as shown in Table 3–4.

Table 3–4 AUTOGEN Dump File Symbols

Value	Effect
DUMPSTYLE = 0	Full dump file created (default)
DUMPSTYLE = 1	Selective dump file created
DUMPFILE = 0	No dump file created

Caution: While it is possible to configure CPUs without dump files, the lack of a dump file can make it difficult or impossible to determine the cause of a system failure.

Sharing Dump Files

Another option for saving dump file space is to share a single dump file among multiple CPUs. This technique makes it possible to analyze isolated CPU failures. But dumps are lost if multiple CPUs fail at the same time, or if a second CPU fails before you can analyze the first failure. Because boot server failures have a greater impact on cluster operation

than failures of other clustered CPUs, you should configure full dump files on boot servers to help ensure speedy problem analysis.

The VMS operating system attempts to ensure that dump files are not unintentionally shared. However, if you want to share dump files, you can follow these steps:

1 Decide whether to use full or selective dump files.

2 Determine the size of the largest dump file needed by any satellite.

   a. Select a satellite whose memory configuration is the largest of any in the cluster.

   b. Set DUMPSTYLE = 0 (or DUMPSTYLE = 1) in that satellite's MODPARAMS.DAT file.

   c. Remove any DUMPFILE symbol from the satellite's MODPARAMS.DAT file.

   d. Run AUTOGEN on that satellite to create a dump file.

3 Rename the dump file to SYS$COMMON:[SYSEXE]SYSDUMP-COMMON.DMP, or create a new dump file named SYSDUMP-COMMON.DMP in the directory SYS$COMMON:[SYSEXE].

4 For each satellite that is to share the dump file, do the following:

   a. Create a file synonym entry for the dump file in the system-specific root. For example, to create a synonym for the satellite using root SYS1E, you could include a command like the following in the appropriate startup procedure:

```
$
```

   b. Add the following lines to the satellite's MODPARAMS.DAT file:

```
DUMPFILE = 0
DUMPSTYLE = 0 (or DUMPSTYLE = 1)
```

5 After a satellite has rebooted, you can delete any SYSDUMP.DMP file in its SYS$SPECIFIC directory.[5]

3.6.1.3 Rebuilding Cluster Disks

V5.2

To minimize disk I/O operations when files are created or extended (and thus improve performance), the VMS file system maintains a cache of pre-allocated file headers and disk blocks.

If a disk is improperly dismounted—for example, if a system crashes or if it is removed from a cluster without running the command procedure SYS$SYSTEM:SHUTDOWN.COM—this pre-allocated space becomes temporarily unavailable. When the disk is subsequently remounted, the MOUNT operation scans the disk to recover the space while rebuilding the disk.

[5] If the old dump file is deleted before the satellite reboots, the disk space is lost. You can recover it by entering the DCL command ANALYZE /DISK / REPAIR.

On a non-clustered VMS system, the scan operation merely prolongs the boot process. In a VAXcluster system, however, this operation can degrade response time for all user processes in the cluster. While the scan is in progress on a particular disk, most activity on that disk is blocked. User processes that attempt to read or write to files on the disk can experience delays of several minutes or longer, especially if the disk contains a large number of files or has many users.

Because the rebuild operation can delay access to disks during the startup of any clustered system, Digital recommends that procedures used to mount cluster disks should mount them with the /NOREBUILD qualifier. When MOUNT/NOREBUILD is specified, disks are not scanned to recover lost space, and users experience minimal delays while CPUs are mounting disks.

System disks are especially critical in this regard, because most system activity requires access to a system disk. When a system disk is being rebuilt, very little activity is possible on any clustered CPU that uses that disk. The system disk (unlike other disks) is automatically mounted early in the boot sequence. If a rebuild is necessary and if the SYSGEN parameter ACP_REBLDSYSD is 1, the system disk is rebuilt during the boot sequence. (The default setting of 1 for the SYSGEN parameter ACP_REBLDSYSD specifies that the system disk should be rebuilt.)

In local area and Mixed-Interconnect clusters, however, the ACP_REBLDSYSD parameter should normally be set to 0 on all satellites. This setting prevents them from rebuilding a system disk when it is mounted early in the boot sequence and eliminates delays caused by such a rebuild when satellites join the cluster.

In large clusters, a substantial amount of system disk space (some for each CPU) may be preallocated to caches, and if many CPUs abruptly leave the cluster (for example, during a power failure), this space could become temporarily unavailable. Thus, ACP_REBLDSYSD on boot servers in local area and Mixed-Interconnect clusters with many CPUs should be set to the default value of 1, and procedures that mount disks on the boot servers should use the /REBUILD qualifier. While these measures can make boot server rebooting more noticeable, they ensure that system disk space is available after an unexpected shutdown.

Once the cluster is up and running, you can submit one or more batch procedures that execute SET VOLUME/REBUILD commands to recover lost disk space.[6] Such procedures can run at a time when users would not be inconvenienced by the blocked access to disks (for example, between midnight and 6 A.M. each day). Because the SET VOLUME/REBUILD command itself determines whether a rebuild is needed, the procedures can execute the command for each disk that is usually mounted.

Caution: If either or both MOUNT/NOREBUILD and ACP_REBLDSYSD = 0 are used when mounting disks, it is essential to run a procedure with SET VOLUME/REBUILD commands on a regular basis to rebuild the disks. Failure to rebuild disk volumes can result in

[6] The procedures execute more quickly and cause less disk access delay if executed on powerful CPUs. Moreover, you can execute several such procedures, each of which rebuilds a different set of disks, simultaneously.

a loss of free space and to subsequent failures of applications to create or extend files.

3.6.1.4 Defining the VAXcluster Alias

V5.2

The VAXcluster alias acts as a single network node identifier that all participating clustered CPUs can use to communicate with other CPUs in a DECnet-VAX network. A maximum of 64 clustered CPUs can participate in a VAXcluster alias. If your cluster includes more than 64 CPUs, you must determine which 64 should participate in the alias and then define it on those CPUs. For detailed information on the VAXcluster alias, refer to the *VMS Networking Manual.*

3.6.1.5 Running AUTOGEN with FEEDBACK

V5.2

Both the number of clustered CPUs and their interactions determine the need to adjust SYSGEN parameters that control certain VMS system resources. The need for adjustment increases as VAXcluster systems grow larger.

To help make these adjustments, the AUTOGEN facility supplied with the VMS operating system has a FEEDBACK mechanism that can examine system resource usage and set parameters appropriately. This mechanism is possible because the VMS operating system maintains statistics on peak and current usage and on how often a CPU must wait for a resource to become available. For example, the system records each instance of a disk server waiting for buffer space to process a disk request. Based on this information, AUTOGEN can automatically size the disk server's buffer pool to ensure that sufficient space is allocated.

Digital strongly recommends that you routinely run AUTOGEN with FEEDBACK to keep all clustered CPUs adequately configured. It is important to repeat this operation regularly, so that each CPU can automatically adjust to changes in the cluster configuration. As CPUs and disks are added, and as user work patterns change, you can run AUTOGEN with FEEDBACK to readjust parameters for current needs.

3.6.1.6 Adding CPUs to an Existing VAXcluster System

V5.2

When a CPU is first added to a cluster, SYSGEN parameters that control the CPU's system resources are normally adjusted in several steps, as follows:

1. The command procedure CLUSTER_CONFIG.COM sets initial parameters that are adequate to boot the CPU in a minimum environment.
2. When the CPU boots, AUTOGEN runs automatically to size the static system (without using any dynamic FEEDBACK data), and the system reboots into the production environment.
3. After the newly added CPU has been subjected to typical use for a day or more, you should manually run AUTOGEN with FEEDBACK to adjust parameters for the production environment.

4 At regular intervals, and whenever a major change occurs in the cluster configuration or production environment, you should manually run AUTOGEN with FEEDBACK to readjust parameters for the changes.

Because, however, the first AUTOGEN run (initiated by CLUSTER_CONFIG.COM) is performed both in the minimum environment and without FEEDBACK, a newly added CPU may be inadequately configured to run in the production environment of some large clusters. For this reason, additional configuration measures might be needed. These are described in the following subsections.

Running AUTOGEN with FEEDBACK For Initial Configuration

To ensure that CPUs are adequately configured for production use when they first join the cluster, you can run AUTOGEN with FEEDBACK automatically as part of the initial boot sequence. While this step adds an additional reboot before the CPU can be used for production work, the CPU's performance can be substantially improved for the first few days of use.

When a CPU first boots into a large cluster, much of the CPU's resource utilization is determined by the current cluster configuration. Factors such as the number of CPUs, the number of disk servers, and the number of disks available or mounted, contribute to a fixed minimum resource requirement. Because this minimum does not change with continued use of the CPU, FEEDBACK information on the required resources is immediately valid.

Other FEEDBACK information, however, such as that influenced by normal user activity, is not immediately available, because the only "user" has been the system startup process. If AUTOGEN were run with FEEDBACK at this point, some system values might be set too low.

By running a simulated user load at the end of the first production boot, you can ensure that AUTOGEN has reasonable FEEDBACK information. The User Environment Test Package (UETP) supplied with the VMS operating system contains a test that simulates such a load. You can run this test (the UETP LOAD phase) as part of the initial production boot and then run AUTOGEN with FEEDBACK before a user is allowed to log in.

To implement this technique, you can create a command file like that in step 1 of the following procedure and submit the file to the CPU's local batch queue from the cluster common SYSTARTUP procedure. Your command file conditionally runs the UETP LOAD phase and then reboots the CPU with AUTOGEN FEEDBACK.

Creating a Command File To Run AUTOGEN with FEEDBACK

As shown in the following sample file, UETP lets you specify a typical user load to be run on the CPU when it first joins the cluster. The UETP run generates data that AUTOGEN uses to set appropriate SYSGEN values for the CPU when rebooting it with FEEDBACK. Note, however, that the default setting for the UETP user load assumes that the CPU is used as a timesharing system. This calculation can produce SYSGEN values

that might be excessive for a single-user workstation, especially if the workstation has large memory resources. Therefore, you might want to modify the default user load setting, as shown in the sample file.

Follow these steps:

1 Create a command file like the following:

```
$!
$!   ***** SYS$COMMON:[SYSMGR]UETP_AUTOGEN.COM *****
$!
$! For initial boot only, run UETP LOAD phase and
$! reboot with AUTOGEN FEEDBACK.
$!
$ SET NOON
$ SET PROCESS/PRIVILEGES=ALL
$!
$! Run UETP to simulate a user load for a satellite CPU
$! with 8 simultaneously active user processes. For a
$! CI-connected CPU, allow UETP to calculate the load.
$!
$ LOADS = "8"
$ IF F$GETDVI("PAA0:","EXISTS") THEN LOADS = ""
$ @SYS$TEST:UETP LOAD 1 'loads'
$!
$! Create a marker file to prevent resubmission of
$! UETP_AUTOGEN.COM at subsequent reboots.
$!
$ CREATE SYS$SPECIFIC:[SYSMGR]UETP_AUTOGEN.DONE
$!
$! Reboot with AUTOGEN to set SYSGEN values.
$!
$ @SYS$UPDATE:AUTOGEN SAVPARAMS REBOOT FEEDBACK
$!
$ EXIT
```

2 Edit the cluster common SYSTARTUP file and add commands like the following at the end of the file. Assume that queues have been started and that a batch queue is running on the newly added CPU. Submit the command procedure UETP_AUTOGEN.COM to the CPU's local batch queue:

```
$!
$ NODE = F$GETSYI ("NODE")
$ IF F$SEARCH ("SYS$SPECIFIC:[SYSMGR]UETP_AUTOGEN.DONE") .EQS. ""
$ THEN
$     SUBMIT /NOPRINT /NOTIFY /USERNAME=SYSTEST -
      /QUEUE='node'_BATCH SYS$MANAGER:UETP_AUTOGEN
$ WAIT_FOR_UETP:
$     WRITE SYS$OUTPUT "Waiting for UETP and AUTOGEN... ''F$TIME()'"
$     WAIT 00:05:00.00          ! Wait 5 minutes
$     GOTO WAIT_FOR_UETP
$ ENDIF
$!
```

Note that UETP must be run under the username SYSTEST.

3 Execute the command procedure CLUSTER_CONFIG.COM to add the CPU.

When you boot the CPU, it runs the command procedure UETP_AUTOGEN.COM to simulate the user load you have specified and then reboots with AUTOGEN FEEDBACK to set appropriate SYSGEN values.

3.6.1.7 Setting Up a New Large VAXcluster System

V5.2

When building a new large cluster, you must be prepared to run AUTOGEN and reboot the cluster several times during the installation. The parameters that AUTOGEN sets for the first CPUs added to the cluster will probably be inadequate when additional CPUs are added. Readjustment of parameters is especially critical for boot and disk servers.

One solution to this potential problem is to run the command procedure UETP_AUTOGEN.COM to reboot CPUs at regular intervals as new CPUs are added. You should run the procedure according to the percentage of growth. For example, each time there is a significant percentage increase in the number of CPUs (from 5 to 10, from 10 to 20, and so forth), you should run UETP_AUTOGEN.COM. For best results, the cluster environment should be as close as possible to the final production environment when you run the procedure.

To set up the cluster, you can follow these steps:

1. Configure boot and disk servers, using the CLUSTER_CONFIG.COM command procedure.
2. Install all layered products and site-specific applications required for the cluster production environment, or as many as possible.
3. Prepare the cluster startup procedures so that they are as close as possible to those to be used in the final production environment.
4. Add a small number of satellites (perhaps 2 or 3), using CLUSTER_CONFIG.COM.
5. Reboot the cluster to verify that the startup procedures work as expected.
6. After you have verified that startup procedures work, run UETP_AUTOGEN.COM on every CPU's local batch queue to reboot the cluster again and to set initial production environment values. When the cluster has rebooted, all CPUs should have reasonable parameter settings. However, check the settings to be sure.
7. Add additional satellites to double their number, and then rerun UETP_AUTOGEN.COM on each CPU's local batch queue to reboot the cluster and set values appropriate to accommodate the newly added satellites.
8. Repeat the previous step until all satellites have been added.
9. When all satellites have been added, run UETP_AUTOGEN.COM a final time on each CPU's local batch queue to reboot the cluster and to set new values for the production environment.

Note that for best performance, you might not want to run UETP_AUTOGEN.COM on every CPU simultaneously, because the procedure simulates a user load that is probably more demanding than that for the final production environment. A better method is to run UETP_AUTOGEN.COM on several satellites (those with the least recently adjusted parameters) while adding new CPUs. This technique increases efficiency, because little is gained when a satellite reruns AUTOGEN shortly after joining the cluster. For example, if the entire cluster is rebooted after 30 satellites have been added, few adjustments are made to system parameter values for the 28th satellite added—only two satellites have joined the cluster since that satellite ran UETP_AUTOGEN.COM as part of its initial configuration.

3.6.2 Handling of the Special Files

V5.0

In VMS Version 5.0, a set of special files that includes the page file, the swap file, the dump file, the cluster incarnation data file, and the Executive loaded images is opened during the system bootstrap to prevent these files from being accidentally deleted or being accidentally shared among member nodes in a cluster.

The implications of this change are as follows:

- The following SET FILE commands require exclusive access to a file. Because the page file, swap file, dump file, and Executive loaded images are open, the following SET FILE commands fail and issue the error message "ACCONFLICT, file access conflict" when you attempt to modify any of these open files.

```
SET FILE/[NO]BACKUP
SET FILE/DATACHECK
SET FILE/END_OF_FILE
SET FILE/ERASE
SET FILE/[NO]EXPIRATION_DATE
SET FILE/EXTENSION
SET FILE/GLOBAL_BUFFER
SET FILE/OWNER
SET FILE/TRUNCATE
SET FILE/VERSION_LIMIT
```

- With proper privileges, it is possible to mark a special file for deletion. That is, when the DCL command DELETE is entered to delete a special file, the special file is removed from the directory and the file is marked for deletion. However, the file body is not deleted. The only way to reclaim the disk blocks occupied by the special file marked for deletion is to reboot the system and enter the command ANALYZE/DISK_STRUCTURE/REPAIR on the system disk.
- The primary page file (PAGEFILE.SYS), the primary swap file (SWAPFILE.SYS), the dump file (SYSDUMP.DMP), and the cluster incarnation data file (SYS$INCARNATION.DAT) must reside in the system-specific rooted directory, SYS$SPECIFIC:[SYSEXE]. If any of these files reside in the common rooted directory, SYS$COMMON:[SYSEXE], the system bootstrap will not be able to find or use the files.

If you use shared dump files you may be affected by this change. A **shared dump file** is a system dump file that is used by two or more nodes in the VAXcluster environment. To allow dump files to be shared among nodes in a VAXcluster environment, do the following where *n* is the size of the system dump file:

1 Create the shared dump file SYS$COMMON:[SYSEXE]SYSDUMP-COMMON.DMP. You can do this by entering the following commands:

```
$
SYSGEN>
SYSGEN>
$
```

2 For each VAXcluster node sharing the file enter the following DCL command where *n* is the system-specific root for the node:

```
$
```

Note: Note that Digital does not recommend shared dump files, because they cannot be relied on to capture memory dumps from multiple failing systems.

3.6.3 Large System Bootstrap Failure

V5.0

VAX computer systems that contain memory in excess of 260 megabytes may fail to bootstrap and return the following error message:

```
%SYSBOOT-F-Unexpected exception  . . .
```

This error occurs when certain combinations of SYSGEN parameters cause the system address space boundaries to be exceeded.

You can solve this problem by performing a conversational boot. At the *SYSBOOT>* prompt you can examine and modify certain parameters that affect the size of the system address space. Digital recommends that you lower the value of the BALSETCNT parameter by half of its current value. You can do this as follows:

```
SYSBOOT>
SYSBOOT>
SYSBOOT>
```

The system should bootstrap successfully at this point.

Automatic detection and correction of these excessive combinations of SYSGEN parameters is expected to be improved for large memory systems in a future release of the VMS operating system.

3.6.4 Booting a Satellite Node (CLUSTER_CONFIG.COM ADD Phase)

V5.0

When a satellite node boots during the CLUSTER_CONFIG.COM procedure's ADD phase, another command procedure, SYS$MANAGER:NETCONFIG.COM, executes. NETCONFIG.COM invokes the Network Control Program (NCP) and Authorize Utilities, which display various informational and error messages. You can ignore these messages.

3.6.5 Rebooting a Satellite Node with an Operating System on a Local Disk

V5.0

In some circumstances, cluster software reboots satellite nodes automatically. Before booting a satellite node, the boot procedures check for the presence of an operating system on the node's local disk. If an operating system is found, the operating system on the satellite's local disk is booted.

If an operating system is installed on a satellite's local disk, one of the following measures should be taken before performing any operation that causes an automatic reboot—for example, executing SYS$SYSTEM:SHUTDOWN.COM with the REBOOT option or using CLUSTER_CONFIG.COM to add that node to the cluster:

- Rename the directory file ddcu:[000000]SYS0.DIR on the local disk to ddcu:[000000]SYSx.DIR (where SYSx is a root other than SYS0 or SYSE). In the following example, SYS0 is renamed SYS1:

  ```
  $
  ```

  Then enter the DCL command SET FILE/REMOVE to remove the old directory entry for the boot image SYSBOOT.EXE as follows:

  ```
  $
  ```

  For subsequent reboots of the system from the local disk, enter a command in the format B/x0000000 at the console-mode prompt (>>>), as in the following example:

  ```
  >>>
  ```

- Disable the local disk. To disable the local disk on MicroVAX II or VAXstation II machines, press the READY button so that the light is off. (This option is not available if the satellite's local disk is being used for paging and swapping.)

3.7 Cluster I/O — Subsystem Changes

The following sections describe changes in the VAXcluster disk–I/O subsystem.

3.7.1 Disk Failover

V5.2

In VMS Version 5.2, the disk class drivers trigger failover in response to certain attention messages generated by Digital Storage Architecture (DSA) controllers. For most configurations, this will not result in any operational changes other than a faster recovery from some classes of errors.

As a result of this change, the requirement that disks that are dual pathed between local controllers be mounted on both nodes has been lifted.

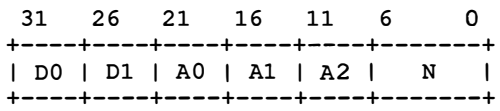
3.7.2 MSCP Serving Third Party Disks

V5.2

Problems that prevented the mass storage control protocol (MSCP) from serving third party disks to other members of a VAXcluster have been corrected in VMS Version 5.2.

For the disk to be served correctly, its device driver must use a device type in the range DT$_FD1 through DT$_FD7, the device class must be DC$_DISK and it must correctly initialize the parameter UCB$L_MEDIA_ID.

The parameter UCB$L_MEDIA_ID contains a bit-encoded media identification that is used by the class drivers to form a local device name for a disk served by a remote VAXcluster member. It consists of five 5-bit fields and one 7-bit field. The fields are defined as follows:



Fields D0 and D1 contain the device type name and are the only fields required by the disk class drivers to form a device name. They are encoded such that A is a 1, B is a 2, and so forth. The remaining fields are optionally used to contain a media name, such as ZZ01 with ZZ encoded in the field A0-A2 in the same fashion as D0 and D1, and the 01 encoded in N as a binary number.

The parameter UCB$L_MEDIA_ID can be initialized by DPT_STORE macros or by driver initialization routines. Refer to the *VMS Device Support Manual* for more details.

Note that this device name *must* match the local device name provided to SYSGEN on the serving node. For example, if the disk name is ZBA0, UCB$L_MEDIA_ID should contain %XD0800000. The SYSGEN command to connect the device on the serving node is:

```
SYSGEN>
```

To serve this device to the other members of the VAXcluster, use the following DCL command:

```
$
```

Note that the MSCP server must have been loaded by setting the SYSGEN parameter MSCP_LOAD (refer to the *VMS VAXcluster Manual*).

If UCB$L_MEDIA_ID is not correctly initialized, no error message is generated. If UCB$L_MEDIA_ID is blank, the disk is not visible to the other VAXcluster members. If UCB$L_MEDIA_ID contains a different name from that used locally, the disk appears to the other VAXcluster members as a different device, which may result in uncoordinated access to the disk by multiple members.

3.8 Debugger — System Management Considerations

V5.2

In previous versions, the debugger and the program being debugged ran in the same process.

Starting with VMS Version 5.2, the debugger consists of two parts (main and kernel) to accommodate the debugging of multiprocess programs (see Section 3.8.2).

Note the following changes affecting system management:

- For a program that runs in one process, a debugging session now requires two processes instead of one.
- For a mutiprocess program, a debugging session requires as many processes as are used by the program, plus an additional process for the main debugger.

Under these conditions, multiple users who are simultaneously debugging programs can place an additional load on a system. This section describes the resources used by the debugger, so that you can tune your system for this activity.

Note that the following discussion covers only the resources used by the debugger. In the case of multiprocess programs, you may also need to tune the system to support the programs themselves.

3.8.1 User Quotas

V5.2

Each user needs a PRCLM quota sufficient to create an additional subprocess for the debugger, beyond the number of processes needed by the program.

BYTLM, ENQLM, FILLM, and PGFLQUOTA are pooled quotas. You may need to increase these quotas to account for the debugger subprocess as follows:

- You should increase each user's ENQLM quota by at least the number of processes being debugged.
- You may need to increase each user's PGFLQUOTA. If a user has an insufficient PGFLQUOTA, the debugger may fail to activate, or produce "virtual memory exceeded" errors during execution.
- You may need to increase each user's BYTLM and FILLM quotas. The debugger requires BYTLM and FILLM quotas sufficient to open each image file being debugged, the corresponding source files, and the debugger input, output, and log files. The debugger command SET

MAX_SOURCE_FILES can be used to limit the number of source files kept open by the debugger at any one time.

3.8.2 System Resources

V5.2

The kernel and main debugger communicate through global sections. The main debugger communicates with up to eight kernel debuggers through a 65-page global section. Therefore, you may need to increase the SYSGEN global-page and global-section parameters (GBLPAGES and GBLSECTIONS, respectively). For example, if 10 users are using the debugger simultaneously, 10 global sections using a total of 650 global pages are required by the debugger.

3.9 DECnet-VAX Notes

This section contains release notes pertaining to DECnet-VAX.

3.9.1 Constraints on Passive Maintenance Functions Relaxed

V5.0

In prior versions of the VMS operating system, the passive functions upline-dump and downline-load (without Software ID) were performed only when the node was present in the volatile node database and the SERVICE CIRCUIT parameter matched the circuit over which the function was requested. For VMS Version 5.0, the SERVICE CIRCUIT parameter is ignored for these functions.

3.9.2 Downline Loading Correction

V5.1-1

VMS Version 5.1-1 incorporates a fix that prevents problems in downline loading system images due to improper assignment of the MOM$LOAD logical name. The problem was observed when loadable images were placed in directories other than SYS$SYSROOT:[MOM$SYSTEM]. The corrected STARTNET.COM checks the system logical name table for the logical name MOM$LOAD and does not assign that logical name if it has been previously defined.

3.9.3 Executor Parameters

The following sections describe changes in DECnet EXECUTOR parameters.

3.9.3.1 MAXIMUM LINKS — Maximum Value Increased

V5.2

The maximum value for the EXECUTOR parameter MAXIMUM LINKS has been increased to 3885. Also, the value for MAXIMUM LINKS is no longer reduced by the value set for the EXECUTOR parameter ALIAS MAXIMUM LINKS.

3.9.3.2 MAXIMUM PATH SPLITS Default Value

V5.0

The EXECUTOR parameter MAXIMUM PATH SPLITS default value is 1. As a result, DECnet-VAX does not path split over equal cost paths by default.

3.9.3.3 PIPELINE QUOTA — Default Value Changed

V5.2

Prior to VMS Version 5.2, the DECnet EXECUTOR parameter PIPELINE QUOTA had a default value of 3000 bytes. Increasing this value to 10,000 can result in a significant improvement in DECnet-VAX performance.

Therefore, in VMS Version 5.2. the default value for PIPELINE QUOTA has been increased to 10,000. This is a change in the default value only; systems that have PIPELINE QUOTA explicitly set in the DECnet database are not affected, even if its value is less than 10,000.

3.9.4 Proxy Access Parameters — Changes

V5.0

The proxy access parameters used by the executive node to determine what kind of access is allowed have changed. By default, incoming and outgoing access are both enabled. The NCP commands to modify the parameters are as follows:

```
NCP>                              ! Enable incoming proxy access
NCP>                              ! Disable incoming proxy access
NCP>                              ! Enable outgoing proxy access
NCP>                              ! Disable outgoing proxy access
```

The proxy database is now built into the network ancillary control process (NETACP) as a volatile database at network startup time. Use the following NCP command to accomplish this:

```
NCP>
```

The STARTNET.COM command procedure supplied by Digital has been modified to issue this command to build the volatile proxy database. If a private network startup procedure is used, and proxy access is desired, then the appropriate commands should be added to load the volatile database.

Changes made to NETPROXY.DAT using the Authorize Utility after the volatile proxy database has been created are also automatically changed in the volatile database.

3.9.5 Session Incompatibility with Phase IV Implementations

V5.0-1

Incompatibilities have been found between the Phase IV Session Control Architecture and the Phase IV+ Session Control Architecture. DECnet-VAX V5.0 implements Phase IV+ of the DNA architecture, which means it is affected by the incompatibilities. One of the following two problems can occur when attempting a connection from a Phase IV+ node to a Phase IV node.

- The Phase IV session control architecture defines the invokeProxy bit in the connect initiate message as being reserved. It also states that any bit defined as reserved must be set to zero unless otherwise

specified. Some Phase IV implementations expect the invokeProxy bit to be zero and will reject the connection with a protocol error if it is nonzero. Others do not check this field because it is not used in Phase IV. Because proxy is architected as part of Phase IV+, the invokeProxy bit is now nonzero. This causes connections initiated from Phase IV+ implementations to be rejected by Phase IV implementations with a protocol error.

- Phase IV implementations expect the session version number in the connect initiate message to be 0, because this is the value for Session V1.0. The session version is now 1 to reflect Session V2.0. The connection will be rejected because of version skew.

Table 3–5 lists the DECnet implementations and the version/update required to resolve the compatibility problems.

Table 3–5 Patches for DECnet Compatibility Problems

DECnet Implementation	Version/Update that Contains the Patch
DECnet-10	Fixed in Autopatch Tape 14[1]
DECnet-20	Fixed in Autopatch Tape 14[1]
DECnet-VAX	All versions work correctly
DECnet/E	Fixed in Version 2.1
PRO/DECnet	Fixed in Version 2.1, patches also available in POS V3.1[1]
DECnet/RT	Patch supplied in RT-11 update January, 1987[1]
DECnet-IAS	No plans to correct the problem
DECnet-DOS	All versions work correctly
DECnet/RSX	Patch supplied with M+ Update C or later[1]
	Patch supplied with M/S Update C or later[1]
DECnet/MicroRSX	Fixed in Version 1.1
DECnet-ULTRIX	All versions work correctly
DECnet Router	Fixed in Version 1.2

[1] Installation of these patches were optional. Some customers may have elected not to install the patches provided and may be running the correct version without the patches.

3.9.6 Support for X.25 Virtual Circuits Requirement

V5.0

In order for DECnet-VAX to support 128 X.25 virtual circuits for Data Link Mapping, the parameter /FILE_LIMIT in the file SYS$MANAGER:LOADNET.COM should be changed from 10 to 128.

3.10 DECwindows Notes for System Managers

The following release notes pertain to DECwindows.

3.10.1 DECwindows Startup for Upgrade Only

V5.2

In VMS Version 5.1, system managers were instructed to execute the DECwindows startup file, DECW$STARTUP.COM, in the site-specific startup file, SYSTARTUP_V5.COM. With VMS Version 5.2, the command procedure DECW$STARTUP.COM is performed as part of VMS startup after SYSTARTUP_V5.COM has completed. If you are upgrading from a Version 5.1, Version 5.1-1 or Version 5.1-B system, you need to remove the @SYS$MANAGER:DECW$STARTUP command from your SYSTARTUP_V5.COM file.

If you did not place a call to DECW$STARTUP.COM in your system startup file, you need to take no additional action after upgrading to VMS Version 5.2.

If you invoked DECW$STARTUP.COM outside of SYSTARTUP_V5.COM, do one of the following:

- Remove the call to DECW$STARTUP.COM that you inserted in the VMS Version 5.1, Version 5.1-1 or Version 5.1-B system.
- Signal VMS to not execute DECW$STARTUP.COM, by defining the logical name DECW$IGNORE_DECWINDOWS in SYSTARTUP_V5.COM as follows:

```
$ DEFINE DECW$IGNORE_DECWINDOWS TRUE   ! Delay DECwindows startup
```

Note: You should only define this logical name if your site-specific startup is going to invoke DECW$STARTUP.COM at a later time. DECW$STARTUP.COM should be invoked on all machines, including non-workstations and workstations that are not using a DECwindows display.

3.10.2 DECwindows Startup

V5.2

The following notes pertain to starting DECwindows:

- Part of the DECwindows startup procedure now executes after your site-specific startup procedure SYS$MANAGER:SYSTARTUP_V5.COM has completed. Therefore, should you exit the startup process directly from this procedure, you will terminate the DECwindows startup prematurely.

- As part of its startup process, DECwindows waits for up to 10 minutes for DECnet startup to complete. If you do not intend to start up DECnet in your site-specific startup procedure, make sure that the command DEFINE DECW$IGNORE_DECNET TRUE is included in that procedure. If you do plan to start up DECnet, make sure that this command is commented out.

  If you start the window system before DECnet is running, the window system will not be able to accept remote DECnet display requests.

- If the DECwindows startup command procedure DECW$STARTUP.COM determines that it is necessary to run the command procedure AUTOGEN.COM, and you elect to have this done automatically, AUTOGEN.COM is now run with feedback if valid feedback data exists.
- If your system does not have valid VAX/VMS or VAXCLUSTER licenses installed, appropriate messages are displayed on your console terminal and DECwindows is not automatically started. If this happens, log in to the console terminal, install the valid licenses, and reboot the system.

3.10.3 Shutting Down Systems Running DECwindows

V5.1-1

The orderly system shutdown command procedure, SYS$SYSTEM:SHUTDOWN.COM, interacts improperly with DECwindows under certain circumstances. This problem will be fixed in a future release of VMS. In the interim, you can avoid the problems by:

- Using SHUTDOWN only from a session that is logged into the system manager's account.
- Using SHUTDOWN after logging in with a SET HOST command from a system other than the one that is being shutdown.

3.10.4 Tailoring DECwindows

V5.2

If you have a small system disk (RD53) and you tailor off the DECwindows files, you may find that you end up with less free space than is indicated by the tailoring-off process. This is most likely related to AUTOGEN creating larger page, swap, and dump files. AUTOGEN is run when you tailor off device support.

When tailoring *on*, please check that you have proper free space. Tailoring does not check that there is sufficient space for the selected files.

3.11 New Devices Supported

The following sections describe new devices supported by VMS.

3.11.1 Magnetic Tape Devices

V5.2

The following sections describe new magnetic tape storage devices supported in VMS Version 5.2.

3.11.1.1 TA90 Magnetic Tape Subsystem

V5.2

The TA90 is a 5- by 4-inch, 200-megabyte (Mb) cartridge tape, fully read- and write-compatible with the IBM® 3480 format. The TA90 includes a master controller and a dual transport unit. As many as three additional dual transport slave units can be connected to a single TA90 master controller for a total of eight drives. The controller connects to the HSC 5X-DA high-speed channel card in the Hierarchical Storage Controller (HSC).

TA90 tape drives can be equipped with optional stack loaders for unattended backup operations. Each TA90 master has two dual-port STI connections to the HSC. Such dual pathing allows each control unit to service two HSC controllers, which significantly increases tape drive availability. The TA90 subsystem includes a 2-Mb cache which allows the controller to prefetch upcoming commands and store them while completing current data transfers. This behavior helps optimize performance. The TA90 is a tape mass storage control protocol (TMSCP) device.

3.11.1.2 RV20 Write-Once Optical Drive

V5.2

The RV20, a 2-gigabyte, double-sided, write-once optical (WORM) disk drive is accessed sequentially similar to a tape. A 100-bit error correction code (ECC) protects user data. The controller performs bad block replacement. Three RV20 slaves can be daisy-chained to the subsystem controller in the RV20 master for a total of four drives.

RV02 cartridges can be used on any Digital RV20 optical subsystem.

The average access time is 212.5 milliseconds (ms) with an average seek rate of 150 ms. The maximum data transfer rate is 262 Kb per second (formatted and sustained) with a burst rate of 1.33 Mb per second.

3.11.1.3 TK70 Cartridge Tape System

V5.2

The TK70 is a 295-Mb, 5.25-inch, streaming cartridge tape system. (See the *VMS I/O User's Reference Manual: Part I* for information about streaming-tape technology). The TK70 tape drive records data serially on 48 tracks using serpentine recording, rather than separate (parallel) tracks. Data written to the tape is automatically read as it is written. A CRC check is performed and the controller is notified immediately if an error occurs on the tape.

The TQK70 is a dual-height, Q-bus controller for the TK70 magnetic tape drive. The TK70 subsystem includes a 38-Kb cache to optimize performance. The TBK70 is a VAXBI-bus controller for the same drive. Section 3.11.1.5 describes compatibility between the TK50 and TK70 magnetic tape cartridge systems.

3.11.1.4 TZ30 Cartridge Tape System

V5.2

The TZ30 is a 95-Mb, 5.25-inch, half-height cartridge streaming-tape drive with an embedded SCSI controller. The TZ30 uses TK50 cartridge tapes. It records data serially on 22 tracks using serpentine recording. Section 3.11.1.5 describes compatibility between the TK50, TK70, and TZ30 magnetic tape cartridge systems.

3.11.1.5 Read and Write Compatibility Among Cartridge Tape Systems

V5.2

When you insert a cartridge tape into the TZ30, TK50, and TK70 tape drives, the hardware initializes the media to a device-specific recording density automatically.

Depending on the type of cartridge and the type of drive on which it is formatted (inserted and initialized), full read and write access to tape cartridges may not be permitted.

Formatting a Blank TK50 Cartridge Tape

A blank, unformatted TK50 cartridge can be formatted on the TK50, TK70, and TZ30 cartridge systems. For example, a TK70 tape drive has full read and write access to a TK50 cartridge formatted on a TK70 drive. Once the cartridge tape is formatted on a particular tape drive, the tape drive has full read and write access to the cartridge tape.

Formatting a Previously Initialized TK50 Cartridge Tape

If a TK50 cartridge tape is formatted on a TZ30 or TK50 cartridge tape drive, the TZ30 and TK50 drives initialize the TK50 cartridge to TK50 density. The following table summarizes the types of access available:

	TK50	
Controller	**Read**	**Write**
TZ30[1]	Yes	Yes
TQK50	Yes	Yes
TQK70	Yes	No

[1] Has an internal controller.

The TK70 tape drive can read data on a TK50 cartridge formatted on a TK50 or TZ30 tape drive.

Formatting a TK50 Cartridge Tape on a TK70 Tape Drive

If a TK50 or TK52 cartridge tape is formatted on a TK70 tape drive, the TK70 cartridge tape drive initializes the TK50 or TK52 cartridge tape to TK70 density. The following table summarizes the types of access available:

	TK50		TK52	
Controller	**Read**	**Write**	**Read**	**Write**
TZ30[1]	No	No	No	No
TQK50	No	No	No	No
TQK70	Yes	Yes	Yes	Yes

[1] Has an internal controller.

The TK50 and TZ30 tape drives cannot read or write data on a TK50 cartridge tape formatted on a TK70 drive.

3.11.1.6 Magnetic Tape Driver Features

V5.2

The following sections describes driver features for magnetic tape devices.

Dual Path Tape Drives

A **dual-path** HSC tape drive is a drive that connects to two HSCs, both of which have the same nonzero tape allocation class. The VMS operating system recognizes the dual-pathed capability of such a tape drive under the following circumstances:

1 The VMS operating system has access to both HSCs

2 Select buttons for both ports are depressed on the tape drive

If one port fails, the VMS operating system switches access to the operational port automatically, provided that the allocation class information has been defined correctly.

Dynamic Failover and Mount Verification

Dynamic failover occurs on dual-pathed tape drives if mount verification is unable to recover on the current path and an alternate path is available. The failover occurs automatically and transparently, and then mount verification proceeds.

A device enters mount verification when I/O request fails because the device has become inoperative. This might occur in the following instances:

- The device is placed off line accidentally.
- The active port of an HSC-connected drive fails.
- A hardware error occurs.
- The device is set to write protected during a write operation.

When the device comes back on line, either through automatic failover or operator intervention, the VMS operating system validates the volume, restores the tape to the postition when the I/O failure occurred, and retries the failed request.

Tape Caching

The RV20, TA90, TK70, and TU81-Plus contain **write-back volatile** caches. The host enables write-back volatile caches explicitly, either on a per-unit basis or on a per-command basis. To enable caching on a per-unit basis, the user can enter the DCL MOUNT command specifying the qualifier /CACHE=TAPE_DATA.

The VMS Backup Utility enables caching on a per-command basis. The user can implement caching on a per-command basis at the QIO level by using the IO$M_NOWAIT function modifiers on commands where it is legal. (See the *VMS I/O User's Reference Manual: Part I*.) In the unlikely event that cached data is lost, the system returns a fatal error and the device accepts no further I/O requests. The IO$M_FLUSH function code can be used to ensure that all write-back-cached data has been written out to the specified tape unit. The IO$_PACKACK, IO$_UNLOAD, IO$_REWINDOFF, and IO$_AVAILABLE function codes also flush the cache.

3.11.2 New Disk Support

The following sections describe new disk storage devices supported in VMS Version 5.2.

3.11.2.1 HSC-Series Controllers

V5.2

The HSC series of intelligent disk controllers consists of the HSC40, HSC50, and the HSC70. HSC controllers are high-speed, high-availability controllers for mass storage devices that implement the DIGITAL Storage Architecture (DSA). An HSC controller is connected to a processor by a Computer Interconnect (CI). The VMS operating system supports the use of the HSC40, HSC50, HSC70 in controlling the RA family of disks.

The HSC40 can support up to 12 Standard Disk Interconnect (SDI) disks from the SA or RA families of disk drives or a combination of up to 12 SDI disk drives and TA-series tape drives.

The HSC70 can support up to 32 SDI disks from the SA or RA families of disk drives or a combination of SDI disk drives and TA-series tape drives.

HSC controllers, in implementing DSA, take over the control of the physical disk unit. VMS operating system processes request virtual or logical I/O on disks controlled by the HSC controller. The VMS operating system maps virtual block addresses into logical block addresses. The HSC controller then resolves logical block addresses into physical block addresses on the disk.

HSC controllers correct bad blocks on the disk by revectoring a failing physical block to another, error-free physical block on the disk; the logical block number is not changed. The VMS operating system, which performs logical or virtual I/O to such a disk, does not recognize that any bad blocks might exist on a disk attached to an HSC controller. HSC controllers also correct most data errors.

The HSC series of controllers provides access to disks despite most hardware failures. Use of an HSC controller permits two or more processors to access files on the same disk.

Note: Only one system should have write access to a Files-11 On-Disk Structure Level 1 disk or to a foreign-mounted disk; all other systems should only have read access to the disk. For Files-11 On-Disk Structure Level 2 volumes, the VMS operating system enables read/write access to all nodes that are members of the same VAXcluster.

HSC-series controllers allow you to add or subtract disks from the device configuration without rebooting the system.

3.11.2.2 SII Integral Adapter

V5.2

The SII integral adapter on the MicroVAX 3300/3400 provides access through the Digital Small Storage Interconnect (DSSI) bus to a maximum of seven storage devices.

The term **dual-host** refers to pairs of CPUs connected to a bus. In dual-host configurations of pairs of MicroVAX 3300/3400 CPUs, the DSSI bus must be connected between the SII integral adapters present on both CPUs.

A maximum of six devices can be connected to the SII integral adapter in dual-host configurations.

3.11.2.3 KFQSA Adapter

V5.2

The KFQSA adapter allows a maximum of seven storage devices for use on Q-bus systems.

In dual-host configurations of pairs of MicroVAX 3800/3900 CPUs, the DSSI bus must be connected between KFQSA adapters present on both CPUs.

A maximum of six devices can be connected to the KFQSA adapter in dual-host configurations.

3.11.2.4 RQDX3 Controller

V5.2

The RQDX3 is a Q-bus controller used with the RD series of Winchester-type disk drives and the RX33 and RX50 flexible diskette drives.

3.11.2.5 RA70 and RA90 Disk Drives

V5.2

The RA70 is a 5.25-inch 280-Mb, high-performance DSA disk drive that uses thin-film media. It has an average access time of 27.0 ms and an average seek time of 19.5 ms. The RA70 uses the Standard Disk Interconnect (SDI), uses the KDA50 controller, and can be dual-ported.

The RA90 is a 1.2-gigabyte disk drive designed with thin-film heads and 9-inch thin-film media with an average seek time of 18.5 ms. The RA90 conforms to DSA and uses the SDI. Both the RA70 and RA90 disk drives can be connected to a medium-sized system with the HSC-series controllers, KDB50, or UDA50 controllers.

3.11.2.6 RD-Series Disks

V5.2

The RD53 and RD54 are 5.25-inch, full-height, Winchester-type drives with average access time of 38 ms and a data transfer rate of 0.625 Mb per second. The RD53 and RD54 have a formatted capacity of 71 Mb and 159 Mb, respectively. When used with the RQDX3 controller, the RD53 and RD54 are DSA disks.

See the *VMS I/O User's Reference Manual: Part I* for information about using RD series disks on the VAXstation 2000.

3.11.2.7 RF-Series Disks

V5.2

The RF series of Winchester-type disk drives consists of the RF30 and the RF71. The RF30 is a 150-Mb, 5.25-inch, half-height disk drive while the RF71 is a 400-Mb full-height disk drive. The RF30 and RF71 include an embedded controller for multihost access and a Mass Storage Communications Protocol (MSCP) server. The RF71 has a peak data transfer rate of 1.5 Mb per second with average seek and access time of 21 ms and 29 ms, respectively.

Both the RF30 and RF71 disks use Digital Storage System Interconnect (DSSI) bus and host adapters.

3.11.2.8 RX-Series Drives

V5.2

The following sections describe RX family of flexible diskette drives.

RX23

The RX23 device is a one-inch high, flexible diskette drive that uses 3.5-inch microfloppy diskettes. The RX23 drive can access standard- and high-density media. The following table summarizes capacities for standard- and high-density media.

Density	Unformatted	Formatted
Standard	1.0 Mb	700 Kb
High	2.0 Mb	1.4 Mb

The RX23 is downwardly compatible in that it can read 1-Mb media. It can also read and write 2.0-Mb double-sided, high-density (135 tracks per inch) media.

The RX23 communicates with the controller using the ST506 fixed disk interconnect (FDI).

RX33

The RX33 is a 1.2 Mb, 5.25-inch, half-height diskette drive. The RX33 can record in either standard- or high-density mode. High-density mode provides 1.2 Mb of storage using 96 tracks per inch using double-sided, high-density diskettes.

In standard-density mode, the RX33 drive is read- and write-compatible with single-sided, standard-density RX50 diskettes.

RX50

The RX50 dual diskette drives stores data in fixed-length blocks on 5.25-inch 0.8-Mb, flexible diskettes using preformatted headers. The RX50 can accommodate two diskettes simultaneously.

3.11.2.9 RZ-Series Disks

V5.2

The RZ series of Winchester-type disk drives consists of the RZ22, RZ23, and the RZ55. The RZ22 and RZ23 are 3.5 inch, half-height Small Computer System Interconnect (SCSI) drives with an average seek rate of 33 ms and an average data transfer rate of 1.25 Mb per second. The RZ22 and RZ23 have a capacity of 52 Mb and 104 Mb, respectively.

The RZ55 is a 332-Mb, 5.25 inch, full-height SCSI drive with an average access rate of 24 ms.

3.11.2.10 RRD40 and RRD50 Read-Only Memory (CDROM)

V5.2

The RRD40 and RRD50 are Compact Disc Read-Only Memory (CDROM) devices that use replicated media with a formatted capacity of approximately 600 Mb.

The RRD40 is a 5.25-inch half-height, front-loading table-top or embedded device that attaches to the system using either the Small Computer System Interface (SCSI) or Q-bus interface.

The RRD50 is a 5.25-inch, top-loading table-top device that attaches to the system using a Q-bus interface.

The RRD40 has an average access time of 0.5 seconds while the average access time for the RRD50 is 1.5 seconds. Both the RRD40 and RRD50 have a data transfer rate of 150 Kb per second.

The media for the RRD40 and the RRD50 are removable 4.7-inch (120 mm) compact disks. However, the media for the RRD40 are enclosed in protective self-loading carriers. The RRD40 with a SCSI interface is also available as an embedded unit. The RRD40 and RRD50 Q-bus subsystems are standard disk MSCP devices.

3.11.2.11 Disk Driver Features

V5.2

The following subsections describe driver features for disk storage devices.

Dual-Pathed DSA Disks

A dual-ported DSA disk can be failed over between the two CPUs that serve it to the VAXcluster under the following conditions:

1. The same disk controller letter and allocation class are specified on both CPUs
2. Both CPUs are running the MSCP server

Note: Failure to observe these requirements can endanger data integrity.

However, because a DSA disk can be on line to only one controller at a time, only one of the CPUs can use its local connection to the disk. The second CPU accesses the disk through the MSCP server. If the CPU that is currently serving the disk fails, the other CPU detects the failure and fails the disk over to its local connection. The disk is thereby made available to the VAXcluster once more.

Note: A dual-ported DSA disk may not be used as a system disk.

Dual-Porting HSC Disks

By design, HSC disks are cluster accessible. Therefore, if they are dual ported, they are automatically dual pathed. CI-connected CPUs can access a dual-pathed HSC disk by way of a path through either HSC-connected device.

For each dual-ported HSC disk, you can control failover to a specific port using the port select buttons on the front of each drive. By pressing either port select button (A or B) on a particular drive, you can cause the device fail over to the specified port.

With the port select button, you can select alternate ports to balance the disk controller workload between two HSC subsystems. For example, you could set half of your disks to use port A and set the other half to use port B.

The port select buttons also allow you to fail over all the disks to an alternate port manually when you anticipate the shutdown of one of the HSC subsystems.

Dual-Pathed RF-Series Disks

In a dual-path configuration of pairs of MicroVAX 3300/3400 CPUs or pairs of MicroVAX 3800/3900 CPUs using RF-series disks, CPUs have concurrent access to any disk on the DSSI bus. A single disk is accessed through two paths and can be served to all satellites by either CPU.

If either CPU fails, satellites can access their disks through the remaining CPU. Note that failover occurs in the following situations:

- When the DSSI bus is connected between SII integral adapters on both MicroVAX 3300/3400 CPUs
- When the DSSI bus is connected between the KFQSA adapters on pairs of MicroVAX 3300/3400s or pairs of MicroVAX 3800/3900s

Note: The DSSI bus should not be connected between a KFQSA adapter on one CPU and an SII integral adapter on another.

SCSI Disk Class Driver

The VAXstation 3100, 3520, and 3540 contain a SCSI bus that provides access to as many as seven SCSI disks. The SCSI disk class driver controls SCSI disks on all of the above systems. Although, SCSI disks do not conform to DSA, they do support the following error recovery features:

- Static and dynamic bad block replacement (BBR)
- Error correcting code (ECC)
- Reexecution of read or write operations within the SCSI drive
- Reexecution of read or write operations by the SCSI disk class driver

All SCSI disks supplied by Digital implement the REASSIGN BLOCKS command which relocates data for a specific logical block to a different physical location on the disk. The SCSI disk class driver reassigns the block in the following instances: (1) when the retry threshold is exceeded during an attempt to read or write a block of data on the disk or (2) when an irrecoverable error occurs during a write operation.

Unlike DSA, there is no forced error flag in SCSI. Blocks that produce irrecoverable errors during read operations are not reassigned in order to prevent undetected loss of user data. Instead, the SCSI disk class driver returns the SS$_PARITY status whenever a read operation results in an irrecoverable error.

3.12 Dialup Support on Serial Lines

V5.2

The VMS terminal driver default modem protocol meets the requirements of the United States and of European countries. The protocol also functions in a subset mode for multiplexers that do not support all modem signals. The following signals are the minimum requirement for correct operation of VMS modem protocol:

- DTR—data terminal ready
- RTS—request to send

- CTS—clear to send
- DSR—data set ready
- CARRIER—data channel received line signal detector

When these signals are present, the terminal driver can perform all necessary modem functions, including hangup and process termination when a disconnect is detected.

Some systems, such as the VAXstation 3100, provide built in serial lines via 6-pin modular jacks. These lines do not provide the minimun required modem signals. Although the hardware may allow a dial out connection to be established, the integrity of that connection cannot be guaranteed by the terminal driver default modem protocol.

The chapter on terminal drivers of the *VMS I/O User's Reference Manual: Part I* provides additional information on which devices provide full modem signals.

3.13 DIGITAL Command Language (DCL) Commands

The following sections pertain to DIGITAL Command Language commands.

3.13.1 DISMOUNT Command — Changes Regarding Open Files

V5.2

In the past, the DCL command DISMOUNT performed a set of relatively simple tests before attempting to dismount a Files-11 volume. These tests verified that the volume was in fact mounted, that it was not the system disk, and that the user had the necessary privileges to dismount the volume. If these tests ran successfully, the volume was marked for dismount and the DISMOUNT command returned a success status.

The dismount of a Files-11 volume actually completed when the volume became idle, that is, when the file system determined that all files in the volume had been closed. In many instances the user might not have been aware of open files on the volume until it was discovered that the volume had remained in the marked-for-dismount state for an extended period of time. At this point, however, the volume was committed to being dismounted regardless of the consequences brought about by closing the open files, if in fact they could be closed (see Section 3.13.1.3).

In VMS Version 5.2, the DISMOUNT command checks for conditions that will prevent the dismount from completing. The conditions are categorized as follows:

- Installed swap and page files
- Installed images
- Devices spooled to the volume
- Open user files (any files not falling into one of the first three groups)

If none of the above conditions is found, the volume is marked for dismount as usual, and the volume changes quickly from the marked-for-dismount state to the dismounted state. If any of the above conditions exists, the DISMOUNT command does not mark the volume for dismount, but instead displays messages indicating that the volume cannot be dismounted, the conditions that exist, and the number of instances of each condition. For example:

```
$

%DISM-W-CANNOTDMT, $10$DJA100: cannot be dismounted
%DISM-W-INSWPGFIL, 4 swap or page files installed on volume
%DISM-W-SPOOLEDEV, 3 devices spooled to volume
%DISM-W-INSTIMAGE, 7 images installed on volume
%DISM-W-USERFILES, 6 user files open on volume
```

As shown in the example, the conditions are displayed in order of decreasing severity, where severity refers to the level of difficulty you would expect to have rectifying these conditions.

The return status from the DISMOUNT command reflects the most severe of the four possible conditions found. You can use this return status to construct a command procedure or image that calls routines to handle the individual conditions. Once one condition has been addressed, the procedure should loop back to attempt the DISMOUNT command again to determine if other conditions exist. The symbol names and values for the four conditions are:

```
DISM$_INSWPGFIL = %X739018
DISM$_SPOOLEDEV = %X739020
DISM$_INSTIMAGE = %X739028
DISM$_USERFILES = %X739030
```

3.13.1.1 Cluster-Wide Support

V5.2

You can use the DISMOUNT command cluster wide if you specify DISMOUNT/CLUSTER. This command first checks for conditions that will prevent the volume from dismounting on the local node. If none is found, it then checks for such conditions on all of the other nodes in the cluster. If the command DISMOUNT/CLUSTER finds one of the conditions on any node, it sends an error message identifying the device and the node on which the error occurred, followed by an error message indicating that there are open files on the volume. For example:

```
$

%DISM-W-RMTDMTFAIL, $10$DJA100: failed to dismount on node SALT
%DISM-W-FILESOPEN, volume has files open on remote node
%DISM-W-RMTDMTFAIL, $10$DJA100: failed to dismount on node PEPPER
%DISM-W-FILESOPEN, volume has files open on remote node
%DISM-W-CANNOTDMT, $10$DJA100: cannot be dismounted
```

In this example, the final return status is DISM-W-CANNOTDMT. Note that while this message is also displayed when one of the error conditions is found on the local node, it is a return status only if the conditions are found on a remote node. Thus, it can be used in a command procedure or an image to distinguish the location of the error condition. The symbol and value for this status are:

```
DISM$_CANNOTDMT = %X739010
```

3.13.1.2 Restoring the Previous Behavior of the DISMOUNT Command

V5.2

There are cases where you want to mark a volume for dismount even though there are files open on the volume. Marking the volume for dismount prevents users from opening any new files, thereby allowing activity to wind down. Also, file-system caches are flushed at the time the volume is marked for dismount, which is especially important when the system is shutting down and the file-system caches must be written to the disk. For these reasons, the qualifier /OVERRIDE=CHECKS has been provided for the DCL command DISMOUNT to override the new V5.2 behavior and allow the volume to be marked for dismount despite the fact that there are files open.

If you specify the qualifier /OVERRIDE=CHECKS, the DISMOUNT command reverts to the earlier behavior with the following exception. Informational messages are displayed to inform you of conditions that will prevent the volume from dismounting, immediately followed by an informational message indicating that the volume has been marked for dismount. The final status is success with a severity of informational (DISM$_MARKEDDMT). For example:

```
$
%DISM-I-INSWPGFIL, 2 swap or page files installed on volume
%DISM-I-SPOOLEDEV, 1 device spooled to volume
%DISM-I-INSTIMAGE, 5 images installed on volume
%DISM-I-OPENFILES, 3 user files open on volume
%DISM-I-MARKEDDMT, $10$DJA100: has been marked for dismount
```

You can specify the equivalent of the qualifier /OVERRIDE=CHECKS when using the $DISMOU system service by using the new **DMT$M_OVR_CHECKS** flag. You should specify this flag in the **flags** argument to the $DISMOU system service if you desire the behavior of previous versions of VMS.

The command procedure SYS$SYSTEM:SHUTDOWN.COM has been modified in VMS Version 5.2 to specify the /OVERRIDE=CHECKS qualifier when dismounting volumes.

You must dismount VAX Distributed File Service (DFS) client pseudo-devices (DFSCn:) using the command DISMOUNT/OVERRIDE=CHECKS DFSCn:. For example:

```
$
```

The following informational message will be displayed, and the device will be dismounted:

```
%DISM-I-USERFILES, 1 user file open on volume
%DISM-I-MARKEDDMT, DFSC1001 has been marked for dismount
```

3.13.1.3 Closing Open Files

V5.2

With VMS Version 5.2, you can address all the conditions that prevent a volume from being dismounted, if you have the appropriate privileges. In previous versions of VMS, you could not dismount disks with installed secondary swap and page files. Disks with secondary swap and page files were considered an extension of the system disk, which cannot be dismounted. VMS Version 5.2 allows you to cancel the installed status of these files, thereby allowing you to dismount the volume. For additional information, see the *VMS Version 5.2 New Features Manual.*

Some knowledge of the files specific to your environment may be required to eliminate the conditions preventing a volume from being dismounted.

First you must determine the names of the files open on the device and the process that owns each file. Each file can then be addressed as shown in the following sections. This information can be displayed using the following command:

```
$
```

where:

ddcu: is the name of the device you are attempting to dismount.

System-Owned Files (Process ID = 0) with the Extension .SYS

The files INDEXF.SYS and QUOTA.SYS can remain open. INDEXF.SYS is normally open on any mounted volume. QUOTA.SYS is normally open if quotas are enabled on the volume. Neither of these open files prevents the volume from being dismounted.

Any remaining files with the extension .SYS are most likely installed secondary swap and page files. You can verify this by examining the site-specific system startup file SYS$MANAGER:SYPAGSWPFILES.COM and by using the DCL command SHOW MEMORY/FILES/FULL. To cancel the installed status of these files, use either of the following commands:

```
$
$
```

For further information see the *VMS Version 5.2 New Features Manual* and SYSGEN's online HELP.

System-Owned Files (Process ID = 0) with the Extension .EXE

System-owned files with the extension .EXE are most likely installed images. You should verify this by examining the installed-image list using the Install Utility's command LIST. You can then cancel the installed status of the files by using the Install Utility's command DELETE. For further information see *VMS Install Utility Manual.*

Process-Owned Files

Process-owned files are normally closed when the processes accessing the files finish with them. Contact the users who own the processes and ask them to complete their work and close the files or log out. If this cannot be done, you can force the processes to exit using the DCL command STOP PROCESS/ID=process-id.

Spooled Devices

You can locate spooled devices using the DCL command SHOW DEVICE. The SHOW DEVICE command displays "spooled" in the device status field if the device is spooled. You can examine the system startup command procedure SYS$MANAGER:SYSTARTUP_V5.COM to determine if the device is spooled to the volume that is being dismounted and to get the names of the queues used by the spooled device. Once you have done this, you should first prevent any queued files from being lost by setting the queue to retain jobs on error as follows:

```
$
```

Next, stop the queue while requeuing the current job, but placing it on hold as follows:

```
$
```

The device can then be set to no-spooled:

```
$
```

You can now restart the queue without losing any jobs in the queue or any files that have been spooled to the volume. If you do not want to wait until the volume is remounted to restart the queue, you can set the device to be spooled to a different volume and restart the queue immediately.

3.13.2 DISMOUNT Command — Explicit /UNLOAD overrides MOUNT/NOUNLOAD

V5.2

In the past, if a MOUNT/NOUNLOAD command was followed by a DISMOUNT or DISMOUNT/UNLOAD command, the volume was not physically unloaded.

Beginning in VMS Version 5.2, there is a difference between an explicit DISMOUNT/UNLOAD command and a DISMOUNT command where the qualifier /UNLOAD is taken as the default value. This distinction allows an explicit DISMOUNT/UNLOAD command to override any /NOUNLOAD qualifier used with the previous MOUNT command. The behavior of a DISMOUNT command without a specified /UNLOAD qualifier is unchanged.

VMS Version 5.2 also provides this capability through the $DISMOU system service. A new **DMT$M_UNLOAD** flag has been added to the $DISMOU system service. You should specify this flag in the **flags** argument to the $DISMOU system service if it is desirable for the volume to be physically unloaded after being dismounted. As with the /UNLOAD qualifier, the DMT$M_UNLOAD flag overrides any /NOUNLOAD qualifier used when the volume was mounted. Note that when you specify the DMT$M_UNLOAD flag, you should not specify the DMT$M_NOUNLOAD flag; however, if you do specify the DMT$M_NOUNLOAD flag, it will be ignored.

The $DMTDEF macro contains the definitions for the DMT$M_UNLOAD flag as well as other flags used by the $DISMOU system service.

Note: The equivalent of a MOUNT/NOUNLOAD command can be invoked through the $MOUNT system service by specifying the MNT$M_NOUNLOAD flag. This flag was previously undocumented. You should specify this flag in the flags argument as input to the $MOUNT system service if you want the volume to remain physically loaded after the volume is dismounted (except when the volume is explicitly unloaded as described above).

The $MNTDEF macro contains the definitions for the MNT$M_NOUNLOAD flag as well as other flags used by the $MOUNT system service.

3.13.3 SET TIME Command

V5.0

When you enter the DCL command SET TIME, the date and time are stored internally. On VAX 8530, 8550, 8700, 8800, 8820, 8830, and 8840 computers, the date and time are sometimes stored incorrectly because of a protocol error in the console interface. If this happens, the system asks you for the date and time the next time you boot.

3.13.4 SET TIME/CLUSTER Command

V5.0

If you have a VAXcluster configuration that includes a VAX 8530, 8550, 8700, 8800, 8820, 8830, or 8840, be careful when you enter the DCL command SET TIME/CLUSTER. Make sure the consoles for these systems are connected and running the console program before you enter the SET TIME/CLUSTER command. If they are not running when you enter the command, the system crashes.

3.14 Distributed File Server (DFS) Error Logging

V5.2

With the Distributed File Server (DFS) Version 1.1, client-device mount and dismount operations are logged to the VMS error log. An incompatibility between the error log format used by DFS and the error log format used by VMS Version 5.2 causes these entries to become corrupted. The corruption can be recognized from the unusual output from the ANALYZE/ERROR command for these entries, such as incorrect device unit numbers and unusually high operation and error counts (often ten or more digits).

You can ignore these entries. The corruption is confined to DFS entries on VMS Version 5.2 systems. No other entries in the error log are affected. DFS mount and dismount operations are correctly logged under earlier versions of VMS. This will be fixed in the next release of DFS. Also, a patch to DFS Version 1.1 that prevents DFS from logging mount and dismount operations is available from your customer support representative.

3.15 Distributed Queuing Service (DQS) — Creating the Account DQS$SERVER

V5.2

With the more security-conscious VMS Version 5.2, creating a default DECNET account is not the default choice of the command procedure NETCONFIG.COM. The Distributed Queuing Service (DQS) Version 1.1 uses the default DECNET account on the DQS client to receive notification from the DQS server that a job has completed printing, when a user has used the PRINT/NOTIFY command. If your DQS client-only VAX computer has no default DECNET account, users on your system cannot receive notification of job completion.

A VAX configured as a DQS server (and also client) uses the DQS$SERVER account to receive notification of job completion when printing to other DQS servers. To receive notification of job completion on a DQS Version 1.1 client-only machine, create a DQS$SERVER account. Using the Authorize Utility with a unique User Identification Code (UIC) and a not easily guessed password, do the following:

```
UAF>
_UAF>
_UAF>
```

Don't forget to create the directory SYS$COMMON:[DQS$SERVER]. You then must do the following using the Network Control Program (NCP), using the same password you specified in AUTHORIZE:

```
NCP>

_
NCP>
```

3.16 DMB32 Product Software Required for DMB32 Communications Controller

V5.0

VAX 8200, 8250, 8300, 8350 and VAX 8530, 8550, 8700, and 8800 systems that include the DMB32 communications controller must install the DMB32 optional software product in order to use the controller's synchronous port. The VMS operating system kit does not contain the DMB32 software.

3.17 $DNS System Service Not Supported

V5.2

A number of files in the VMS Version 5.2 kit refer to a system service $DNS, a feature that may be included in a future release of VMS. Use of these files under VMS Version 5.2 is not supported. Under VMS Version 5.2, the $DNS system service will return SS$_UNSUPPORTED to all calls.

Note: The Remote System Manager and VAX Distributed File Service products contain files with similar names. The files mentioned above will not interfere with the operation of these products.

3.18 DUDRIVER and DSDRIVER Correction

V5.1

Changes have been made to the disk class drivers (DUDRIVER and DSDRIVER) that improve failover on dual path DSA disks (for example, RA60, RA81) connected to local controllers. A node with a local controller that is accessing a disk through the MSCP server on the other node now discovers its local, secondary path soon after boot and switches to that path if it becomes unreachable through the remote server.

3.19 Dump Files

The following sections pertain to dump files.

3.19.1 Interaction Between the SAVEDUMP Parameter and PAGEFILE.SYS Size — Caution

V5.0

When a system dump is saved in the page file, the SYSGEN parameter SAVEDUMP specifies whether the space used by the dump should be reserved until the dump is analyzed. If your system is configured to write a complete physical memory dump (DUMPSTYLE set to 0, the default) you can calculate the size of the dump and make your page file large enough to hold both the dump and the required additional paging space.

However, a more difficult situation arises when you are dumping to the page file and your system is configured to write a selective dump file (DUMPSTYLE set to 1). Selective dumps write out processes until they are all dumped or dump file space is exhausted. If you reduce your paging file to a size smaller than what is required for a full dump in an attempt to obtain the disk space savings of selective dumping, selective dumps may use up the required additional paging space and the dump will be discarded on reboot.

While Digital does not recommend enabling SAVEDUMP, it does recognize that the parameter allows some configurations with very restricted disk space to save crash dumps. Systems that enable SAVEDUMP must recognize the need for a substantially larger primary page file in order to preserve system dumps.

3.19.2 Selective Crash Dump Files — Caution

V5.0

One of AUTOGEN's functions is to recommend a dump file size based upon your system's physical memory and other system parameters. By default, this procedure has not changed, and continues to work correctly. However, if you enable selective dumps, and NETACP uses a large (over a thousand pages) address space on your system, AUTOGEN's algorithm may not recommend a page file large enough to hold as many processes as may make a particular crash dump useful for analysis.

Selective dumps represent a tradeoff between the usefulness of a crash dump and the disk space required to hold it. There is no formula about how many processes are useful to dump. However, if you notice that only a small fraction of memory resident processes are dumped

during a bugcheck, it will probably be in your interest to increase the size of your system dump file according to the approximate size of the NETACP working set. You can do this by noting NETACP's working set size as given by the DCL command SHOW SYSTEM and in SYS$SYSTEM:MODPARAMS.DAT, either increasing the value of DUMPFILE by that amount or adding a line to have DUMPFILE increased beyond AUTOGEN's calculated value. You then need to invoke AUTOGEN as follows:

3.19.3 Dump File Size Changes

V5.0

The system bugcheck mechanism has been rewritten to allow selective dumps, and the System Dump Analyzer (SDA) has been enhanced to analyze both complete and selective dumps. These changes have been included so that dump files for large memory systems will not consume large amounts of disk space. Selection of complete or selective dumps is accomplished by using the new SYSGEN parameter DUMPSTYLE. The value of 0 enables the traditional complete memory dump. Setting DUMPSTYLE to 1 enables the selective dump.

As a result of this enhancement, modifications were made to the sizing algorithm for AUTOGEN's dump file. This allows AUTOGEN to produce smaller dump files. To enable the smaller selective dumps and AUTOGEN's new dump file sizing algorithm, set the parameter DUMPSTYLE=1 in MODPARAMS.DAT. If you have overridden AUTOGEN's dump file calculation by defining a symbol DUMPFILE=0 in MODPARAMS.DAT, remove this symbol definition to let AUTOGEN create a smaller dump file.

3.19.4 System Dump File — Calculating Minimum Size

V5.2

The new SYSGEN parameter ERLBUFFERPAGES specifies the number of pages of memory to allocate for each buffer requested by the ERRORLOGBUFFERS parameter. The ERLBUFFERPAGES parameter has a default value of 2 pages and a maximum value of 32 pages.

The new size of the error log buffer in memory requires that system managers increase the minimum dump file (SYS$SYSTEM:SYSDUMP.DMP) size. Use the following formula to calculate the size (in blocks) of the system dump file:

```
[(value of SYSGEN parameter ERRORLOGBUFFERS) x (value of SYSGEN parameter
ERLBUFFERPAGES)] + (number of pages of physical memory) + 1
```

To determine the number of pages of physical memory, use the SHOW MEMORY command.

Note: AUTOGEN automatically sizes the dumpfile correctly unless overridden in the file MODPARAMS.DAT. See Section 3.19.3 for details.

3.20 Ethernet Notes

The following notes pertain to the Ethernet.

3.20.1 DEBNA VAXBI Ethernet Controller — Tuning the VMS Operating System

Tune the VMS operating system for DEBNA controllers by adjusting the SYSGEN parameters and network parameters, as described in the following sections.

3.20.1.1 SYSGEN Parameters

V5.2

For systems with multiple DEBNA controllers, AUTOGEN automatically adds the value 50*n to the SYSGEN parameter SCSBUFFCNT, where n is the number of DEBNA controllers in your system. If you included the following line in the file SYS$SYSTEM:MODPARAMS.DAT under a previous version of VMS, delete it before running VMS Version 5.2:

```
ADD_SCSBUFFCNT=50*(n)
```

Otherwise the SYSGEN parameter SCSBUFFCNT will be set to a higher value than necessary.

3.20.1.2 Network Parameters

V5.0

Check and adjust the network parameters in the following list:

- LINE BUFFER size must be 1498

  Check the LINE BUFFER size. Make sure that it is set to 1498.

- LINE RECEIVE BUFFERS must be at least 8

  The LINE RECEIVE BUFFERS parameter should not be set to a value of less than 8. If it is set to less than 8, it may cause an excessive loss of packets in the controller (DEBNA).

- HELLO TIMER

  Adjust this parameter if Adjacent Node Listener Receive timeouts occur by increasing the Hello Timer value on the adjacent node.

- BUFFER_LIMIT in LOADNET.COM

  The BUFFER_LIMIT should be increased for each additional DECnet line. Increase it from the default of 65K. The typical value for four lines is 131K. Line Open errors occur if BUFFER_LIMIT is too small.

3.20.2 DEBNI Ethernet/802 Controller

The following sections pertain to the DEBNI Ethernet controller.

3.20.2.1 I/O Interface

V5.2

VMS Version 5.2 supports a new Ethernet/802 controller called DEBNI that connects to the VAXBI bus. The QIO interface to the DEBNI controller is the same as that described for the DEBNA device driver in the *VMS I/O User's Reference Manual: Part II*, except that the device type of the DEBNI controller is DT$_ET_DEBNI.

The DEBNI controller is supported by ETDRIVER. Its device name is:

```
ETcu
```

where:

c is the controller and u is the unit number (for example, ETA0).

The NCP LINE and CIRCUIT name for the DEBNI controller is:

```
BNA-<controller number>  (for example, BNA-0 for ETAn, BNA-1 for ETBn)
```

3.20.2.2 Node ID

V5.2

If a DEBNI controller is configured at a lower node id than the BI disk adapters, then the disk adapter controller letter is incorrectly incremented. For example, instead of DJA0 the disk would be DJB0.

This problem will be fixed in a future release of VMS.

3.20.3 DEQNA Ethernet Adapter May Receive Corrupt Data

V5.0

Under certain rare circumstances, the DEQNA Ethernet adapter in large and complex Ethernet configurations may receive corrupted data. The VMS operating system automatically enables a data integrity feature that reduces the risk to VAXcluster users. Digital recommends that this feature remain enabled on all VAXcluster members that use DEQNA devices.

The DECnet command COPY provides data integrity checking. User-written applications performing data transfers to systems using DEQNA adapters must provide their own data integrity checking.

3.21 Halting a VAX 8530, 8550, 8700, or 8800 System

V5.0

After you halt a VAX 8530, 8550, 8700, or 8800 system, you must enter the CLEAR RESTART_FLAGS command to clear the WARM_RESTART and COLD_RESTART flags. For example:

```
>>>
>>>
```

Clearing these flags prevents the automatic boot and restart procedures from looping indefinitely when you enter the next BOOT command. Keep this in mind when you halt the system during the VMS installation procedure.

3.22 Hierarchical Storage Controller Notes

The following release notes are specific to the Hierarchial Storage Controller (HSC):

3.22.1 Incorrect CIRCUITS Display for HSC40 and HSC70 Fixed

V5.2

In VMS Version 5.1-1, HSC40s and HSC70s were displayed as "HSC50" in the RP_TYPE field of the CIRCUITS class. This has been corrected.

3.22.2 Shadow-Set Failover Problem

V5.1-1

If an HSC fails while a shadow set is performing a copy operation, the shadow set fails over and is rebuilt on the other HSC. This procedure can sometimes alter the copy mode.

This will be corrected in a future release.

3.23 INITIALIZE Command — Defining Volume Serial Numbers

V5.0

Since the introduction of Digital Storage Architecture (DSA) disks, the VMS operating system has not properly initialized the Files-11 On-Disk Structure Level for a DSA disk to include the hardware serial number stored in the DSA volume's factory formatting data. This causes the home block SERIALNUM field to be zero which, in turn, causes the $GETDVI system service and the F$GETDVI lexical function to return zero for the SERIALNUM item, instead of a unique serial number.

In VMS Version 5.0, the Files-11 initialization process has been corrected. The correction has been made both in the INITIALIZE command and in the BACKUP/IMAGE command. This means that all output volumes processed by these two commands will have properly defined serial numbers and return something other than zero from the $GETDVI item code SERIALNUM.

Volumes transported to Version 5.0 and not processed with one of these two commands will continue to report zero for the SERIALNUM item, because the home blocks on such volumes will continue to contain zero in the SERIALNUM field. The most convenient way to overcome this problem is to perform a BACKUP/IMAGE on volumes where a correct SERIALNUM value is deemed important.

In addition, there is a restriction on SERIALNUM initialization by BACKUP/IMAGE. The process executing BACKUP must have LOG_IO privilege, or BACKUP must be installed with LOG_IO privilege. This is because the I/O function used to obtain the SERIALNUM information for inclusion in the home block is a physical I/O function. Digital expects to remove this restriction in a future release of the VMS operating system.

Finally, all these services are restricted to directly-accessed DSA disks. DSA disks accessed through the MSCP server will continue to initialize with the SERIALNUM field set to zero. This restriction results from limitations in the MSCP server with respect to serving DSA disks. Digital

will remove this restriction in a future release of the VMS operating system.

3.24 Local Area Terminal (LAT) Notes

The following notes pertain to Local Area Terminal (LAT) software.

3.24.1 Delay in Process Disconnect

V5.0

If virtual terminals are enabled on your system and you enter the LAT command DISCONNECT, the process is not deleted immediately. The process is deleted when the timeout period expires. This is normal and should be expected.

3.24.2 Dynamic Service Rating Algorithm

V5.2

VMS Version 5.2 uses a new Local Area Terminal (LAT) dynamic service rating algorithm. The LAT dynamic service rating is a value ranging from 0 (meaning this VAX system is very busy) to 255 (this VAX system is not busy). This value is used by terminal servers to balance the load on different nodes of a VAXcluster. The algorithm used in VMS Version 5.0 often produced artificially low rating values. The new algorithm calculates values that provide much better load balancing. In addition, you can adjust the parameter CPU_RATING in this algorithm to suit the needs of a particular VAXcluster.

At every MULTICAST timer tick, the routine LTDRIVER calculates a new LAT service rating. The MULTICAST timer is normally set to 60 seconds. The new algorithm calculates the LAT rating using scaled integer arithmetic with the following formula:

$$\text{RATING} = \frac{20 * (\text{IJOBLIM}-\text{IJOBCNT})}{\text{IJOBLIM}} + \frac{\min(235, \text{CPU\_RATING}) * 100}{(100 + \text{LOAD\_AVERAGE})}$$

The first term is known as the **availability** term, and represents the proportion of free interactive job slots left on the system. If the availability term reaches 0, then the rating is set to 0 to indicate that there are no free job slots.

The second term is the **load** term. As system load increases, the load term causes the LAT rating to decrease.

The new algorithm uses a quantity called LOAD_AVERAGE. This is a moving average of the number of computable processes waiting in the VMS scheduler queues. Processes with a priority less than DEFPRI are not counted; thus, background batch jobs no longer have any effect on the LAT rating.

The factor CPU_RATING in the load term adjusts the LAT rating according to the power of the CPU, as you judge it. A higher value for CPU_RATING produces a higher LAT rating for the CPU. The default value for CPU_RATING is IJOBLIM, since it is assumed that you set a higher IJOBLIM on CPUs that have a greater capacity.

You can change the value of CPU_RATING by using the new qualifier /CPU_RATING for the LATCP command SET NODE. This allows you to adjust the LAT rating algorithm independently of the IJOBLIM setting. This is desirable if IJOBLIM is used only to restrict the number of interactive jobs, and you want to adjust the LAT rating in a CPU-dependent fashion, while maintaining a dynamic rating algorithm. The command syntax is as follows:

```
SET NODE/CPU_RATING=nnn
```

where:

nnn can range from 0 (which means use IJOBLIM) to 100.

The value 100 was chosen as the upper limit so that relative CPU weights can be described as percentages. For example, one CPU in a cluster might be rated as 100, and another less powerful one as 50. This means that one is 50% as powerful as the other. Internally, this value is scaled up to span the range from 0 to 255 to match the LAT rating maximum of 255.

3.24.3 LAT Error Message Translation

V5.2

When examining LAT printer queue status to determine the cause of a print job failure, you may notice a LAT error message that begins with the string *%LAT-F-NOMSG* followed by a hexadecimal number. To translate this into a text error message, you must use the following command before issuing a SHOW QUEUE command:

```
$
```

As an example,

```
%LAT-F-NOMSG 01769F54
```

translates into:

```
%LAT-F-LRJRESOURCE, insufficient resources at server
```

3.24.4 LAT Control Program (LATCP) — Changes and Restrictions

V5.0

Following is a list of changes and restrictions to the LAT Control Program (LATCP):

- LATCP no longer restricts the service node to a single Ethernet. LATCP now supports a configuration that allows terminal connections from two separate Ethernets.
- LATCP now allows dedicated ports to be established that are associated with application services.
- The commands START NODE, CREATE LINK, and SET LINK accept the /DECNET qualifier. This qualifier directs the LAT protocol to use the DECnet Ethernet address (/DECNET) or the hardware address (/NODECNET) when starting the Ethernet controller. The default is /DECNET.

The qualifier /NODECNET can help improve performance when you have a VAX processor with two Ethernet controllers connected to the same Ethernet backbone. You can restrict LAT traffic to one Ethernet controller and DECnet traffic to the other. Note that once you start the LAT protocol using the /NODECNET qualifier, you cannot start DECnet on the same Ethernet link without stopping the LAT port driver and restarting it.

- When a SET NODE command is executed before a START NODE command, LATCP only parses the START NODE qualifiers /LINK and /DECNET. LATCP incorrectly assumes that the START NODE qualifiers /ENABLE, /DISABLE, /IDENTIFICATION, and /MULITCAST_TIMER were correctly set at the same time as the node name. Although this is usually the manner in which characteristics are set, it should not be a requirement. To avoid this, issue a SET NODE command that specifies the correct qualifiers before issuing a START NODE command. This restriction will be removed in a future release of the VMS operating system.
- LATCP no longer allows remapping an application port while the port is spooled. In order to change the /NODE, /PORT, and/or /SERVICE qualifier values, you must stop the associated queue, set the device /NOSPOOL, and then run LATCP to change the assignments.

3.24.5 LATCP Version Check

V5.2

In order to start the LAT procotol on your system, you must use the version of LATCP supplied with VMS Version 5.2. If you attempt to use an older version of LATCP, you will see the following fatal error message and the LAT protocol will not start:

```
%LAT-F-VERMISMATCH, version of LATCP does not match driver
```

or:

```
%LAT-F-NOMSG, Message number 01769FD4
```

3.24.6 LAT PASSALL Session

V5.0

When using a host-initiated connection with the terminal that has the PASSALL characteristic set, the terminal server's input flow control for the port is disabled. This is normal behavior.

3.24.7 LAT Symbiont (LATSYM) Correction

V5.1

Prior to VMS Version 5.1, the LAT symbiont (LATSYM) would allocate an output device, print the requested file, and then deallocate the device. This would occur even if another print job was in the print queue.

Beginning with VMS Version 5.1, this behavior has been modified. Now, every time a print queue that has a LAT printer as a target device is started, LATSYM allocates the LTAx: output device when it begins to print the first file in queue and keeps it allocated across multiple print jobs, until the output queue is stopped. The result is that other

applications that assign an output channel to that LTAx: write their output to a temporary disk file, which gets spooled to the LATSYM print queue when the application closes its output channel.

3.24.8 LAT Ratings Display

V5.2

If dynamic LAT service ratings are used, the LAT Control Program (LATCP) command SHOW CHARACTERISTICS now displays the current dynamic LAT service rating as follows:

```
Dynamic rating: nnn
```

(smallskip) instead of:

```
Rating: <auto>
```

3.24.9 SETMODE — Implementation of More Terminal Functions

V5.2

In previous versions of VMS, using the terminal driver SETMODE/SETCHAR $QIOs or the $SET TERMINAL command to modify a LAT terminal's speed, parity, and frame size characteristics had no effect on the terminal server's physical port characteristics.

With VMS Version 5.2, you can use either function to change the terminal server's physical port characteristics. If the terminal server port has more than one session running, the other sessions are notified of the port characteristic changes.

Note: The use of this feature requires a corresponding change in terminal server software. This functionality may not be immediately available with all terminal servers.

3.24.10 Solicit Connection QIO

V5.0

Do not enter a QIO connection request if LATCP has not yet started the LAT protocol. The QIO request may not complete and will not return an error.

3.25 MA780 (Multiport Shared Memory)

V5.1

All processors connected to the multiport shared memory (MA780) must be running the same version of VMS, either Version 4.x or Version 5.x. Running one processor at Version 4.x and another at Version 5.x does *not* work, due to changes in the global section data structure for VMS Version 5.0.

3.26 Manually Configuring Q-bus Devices on MicroVAX 3400 Series Systems

V5.1

When using the SYSGEN command CONNECT to manually configure Q-bus devices, note that on MicroVAX/VAXserver 3400 series systems, the Q-bus adapter is not adapter nexus 0. The SYSGEN SHOW/ADAPTER command shows the following:

```
SYSGEN>

CPU Type:  MicroVAX 3400 Series

Nexus     Generic Name or Description
  0       KA640
  1       UB0
```

The following CONNECT command connects a Q-bus device to the KA640 adapter and may result in a system crash:

```
SYSGEN>
SYSGEN>
```

Use /ADAPTER=UB0 to specify the first Q-bus adapter.

3.27 Mass Storage Control Protocol (MSCP) Server

The following sections contain information concerning the mass storage control protocol (MSCP).

3.27.1 Buffer Segmentation Algorithm Problem

V5.0-1

In Version 5.0 of the VMS operating system, the buffer segmentation algorithm used by the MSCP Server did not correctly account for transfers exhausting the local controller's maximum byte count. Under certain circumstances, this resulted in a port driver deadlock.

Version 5.0-1 of the VMS operating system fixes this problem.

3.27.2 Controller Letters — Restriction

V5.1-1

The MSCP server only serves disks with controller letters A through G and shadow set virtual units that have the controller letter S. This restriction will be lifted in a future release.

3.27.3 Diskette Devices — Some Not Allowed

V5.0

The mass storage control protocol (MSCP) does not allow all the functions associated with certain diskette devices. Therefore, the MSCP server (which is based upon MSCP) does not automatically serve diskette devices such as the RX01, RX02, and RX33.

3.28 Modem Signal Requirements now Enforced

V5.0

The VMS operating system now enforces the modem signal requirements described in the *VMS I/O User's Reference Manual: Part I*. System managers must ensure that their host system modems are wired properly and meet the signal requirements.

If a modem is wired incorrectly, the following error message is displayed:

```
VAX/VMS host system modem wired incorrectly - contact your system manager
```

3.29 Modified-Page Writer — Flushing of Modified_Page List Eliminated

V5.2

Prior to VMS Version 5.2, the modified-page list was completely written or "flushed" under the following circumstances:

- Deletion of a global section with file backing store
- Balance-slot cleanup for a deleted process
- A process dead-page-table scan
- An operator-induced crash using the OPCCRASH procedure

The frequency of flushing depended on workload and configuration, and on some systems, there was a significant negative effect on system performance.

In VMS Version 5.2, all flushing of the modified-page list has been eliminated and replaced with a selective modified-page writing mechanism. Assuming that a system is configured with adequate main memory for its workload, there are two major benefits of the new approach:

- The size of the average modified-page list is significantly higher, increasing its effectiveness as a page cache.
- The average free-page file space is significantly greater, decreasing the frequency of process or system hangups due to insufficient page-file space.

3.30 Modular Executive — Notes

The following sections describe the Modular Executive in VMS Version 5.0.

3.30.1 Introduction to the Modular Executive

V5.0

All of the code contained previously in the image SYS.EXE has now been separated into approximately 20 images, called Executive loaded images. All executable code in the modular executive is contained in these images.

The partitioning of SYS.EXE is meant to group together modules that logically belong together in terms of the functions they perform. For example, modules that deal with image activation and image rundown were moved to an image called IMAGE_MANAGEMENT.EXE, while modules related to system security were moved into an image called SECURITY.EXE.

In the modular executive, SYS.EXE remains as one of the many executive images. However, SYS.EXE, now called the base image, has some unique functions:

- It provides a transfer vector area in system (S0) space for routines of the Executive loaded images.
- It includes an area for universal data cells, which are cells that all code in both the Executive and other privileged images can access.

All transfer vectors and global data cells within the base image are guaranteed to be fixed for all time. The base image is the unchanging pathway by which routines and data in executive loaded images can be accessed.

3.30.2 Effects on Privileged Code

V5.0

This reorganization of the Executive affects only privileged code. All privileged code must be relinked with the Version 5.0 linker against the new Version 5.0 SYS.STB, the system symbol table.

In earlier versions of the VMS operating system, several data structures were statically declared in the SYS.EXE image. In VMS Version 5.0, some of these data structures have been moved into one of the Executive loaded images. All other Executive loaded images and privileged images must reference the structure through a pointer stored in the base image. When data was moved out of the base image, the name of the cell was changed so that any code referencing the cell would not link with undefined symbols. Any privileged image that fails to link in such a way requires source-code changes to reference these data structures through pointers to these structures.

When you are debugging privileged code or device drivers, it may be necessary to disable system paging. The special SYSGEN parameter SYSPAGING was provided for this purpose. For VMS Version 5.0, the SYSPAGING parameter has been replaced with another special parameter, S0_PAGING, which is a mask with a "1" bit to disable paging. If bit 0 (low-order bit) of S0_PAGING is set, then paging of the Executive is disabled. If bit 1 of S0_PAGING is set, then paging of RMS is disabled. Note that the S0_PAGING parameter is a special SYSGEN parameter and should be used only by your Digital Field Service representative.

3.30.3 Effects on SYSGEN

V5.0

The SYSGEN CONNECT/DRIVERNAME command specifies the name of the driver as recorded in the prologue table. If the driver has not been loaded, the system assumes that the driver name is also the name of an executable image (file type of EXE) in the SYS$LOADABLE_IMAGES or the SYS$SYSTEM directory, and loads the driver. The default for the driver name is the first two characters of the device name plus DRIVER.

3.30.4 Effects on System Management

The following sections describe the effect of the modular executive upon system images.

3.30.4.1 SYS$LOADABLE_IMAGES Directory on the System Disk

V5.0

The SYS$LOADABLE_IMAGES logical name points to a special directory on the system disk. This directory contains the set of images that are loaded during the bootstrap of the system. The Executive loaded images, device drivers, and other images loaded into system space (for example, SYSLOA780.EXE) reside in this directory. Images in this directory are special in that they are not executable in the conventional sense; that is, they cannot be executed with RUN or other DCL commands.

3.30.4.2 System Failure

V5.0

If the system fails, or if the system is forced to fail in an emergency shutdown with CRASH, the list of Executive loaded images in the system is printed on the console terminal. The bugcheck message and information about the stack are printed, followed by the list of Executive loaded images with the starting and the ending address of each image. Then a dump of memory is written to the system dump file on disk.

3.30.5 Version Numbers and Version Checking

V5.0

Prior to VMS Version 5.0, a system version number was used to control several different ways that the system could change. These included the following:

- Location of routines or data in SYS.EXE
- Layout of data structures
- Details of the interface to a routine

Even though the modular executive guarantees that all transfer vectors and global data cells within the base image are fixed, privileged code still needs relinking when the system changes in one of the three ways previously mentioned.

In addition to the overall system version number, the modular executive has several version numbers, one for each functional component of the Executive. Each symbol in the base image has a small set of version numbers associated with it. When a privileged image is linked against the system symbol table (SYS.STB), version numbers associated with all routines referenced by the image are recorded in the image header. Version numbers associated with routines not referenced by this image are not recorded. Thus, the version numbers recorded in the image header provide a complete description of dependencies of this image on the set of routines in the base image.

For major releases of the VMS operating system after Version 5.0, privileged images need not be relinked for every major release of the VMS operating system. A privileged image needs to be relinked against the new system symbol table only if a functional component on which the

image is dependent contains an incompatible change (in data structures or routine interfaces) from the previous release.

For example, a user-written device driver contains references to various I/O routines; therefore, the version number of the I/O component in the system symbol table against which this driver is linked is recorded in the image header of the driver. Changes to other functional components of the modular executive for example, memory management will not likely affect this device driver. Therefore, this driver need not be relinked if a subsequent release of the VMS operating system contains extensive changes to the memory management component and no changes to the I/O component.

During the system bootstrap, the secondary bootstrap program (SYSBOOT) and the system initialization code perform checks on the version of the Executive loaded images and other images loaded into the system space against the base image. If an incompatibility in the version numbers is detected, the image is rejected and the bootstrap fails.

When loading a device driver, SYSGEN checks the version numbers of the driver against the base image. If an incompatibility in the version numbers is detected, the driver is not loaded.

The image activator and the Install Utility also perform version checks. When a mismatch between the set of version numbers of a privileged image with the base image is detected, the CMKRNL and CMEXEC privileges are removed, but the activation (or making the image into a known file, in the case of INSTALL) continues.

3.31 Monitor Utility — Notes

The following sections provide information about the Monitor Utility.

3.31.1 Error in Display

V5.0

If the number of free packets on the SRP, IRP, or LRP lists exceeds 500, the Monitor Utility will display asterisks in those fields rather than the actual data. This affects both the POOL and the DECNET classes.

MONITOR must hold exclusive access to each list while counting free packets. Holding exclusive access for a significant length of time can have serious side effects. This change reduces the amount of time that MONITOR holds exclusive access to these lists.

3.31.2 RMS Bucket and Multibucket Split Rates Invalid

V5.0

When you enter the MONITOR command, RMS/ITEM=LOCKING, MONITOR displays RMS bucket and multibucket split rates. However, because the counters are not maintained properly in RMS, MONITOR always displays a rate of zero for these items. This will be corrected in a future release of the VMS operating system.

3.32 Mount Utility — Notes

The following sections discuss changes to the Mount Utility.

3.32.1 Automatic Tape-Labeling — Problem

V5.1

The /AUTOMATIC qualifier to the MOUNT command does not work as intended. This qualifier controls the automatic tape-labeling feature (which is enabled by default) and causes the system to generate labels for multiple magnetic tape reels. For example, to mount a magnetic tape, you could enter the following command:

```
$
```

If you use more than one magnetic tape reel, the system expects the next tape to be labeled FIRS02 (the label generated by the automatic tape labeling feature).

Although this feature is intended to allow users to mount a magnetic tape and specify their own volume labels for multiple magnetic tape reels, it does not work correctly. For example, suppose you mount a tape using the following command:

```
$
```

The first magnetic tape must be labeled FIRST, but the system ignores the user-specified label for the second reel (NEXT) and expects the label of the second tape to be FIRS02.

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

3.32.2 Incorrect Device Reference Count Prevents Volume From Being Mounted

V5.1

In previous versions of VMS, beginning with Version 4.2 and prior to 5.1, a problem existed that caused the device reference count to be incorrect.

In a cluster environment, a node shutting down after an incorrect reference count would not be able to mount the device when the node rejoined the cluster. MOUNT would return a "VOLALRMNT, another volume of same label already mounted" error.

This problem mainly occurred on systems that made extensive use of global sections—in particular, systems running ALL-IN-1.

This problem is corrected in VMS Version 5.1.

3.33 Network Control Program (NCP) — SET/DEFINE CIRCUIT Command

V5.2

In VMS Version 5.0, the COST parameter for the SET/DEFINE CIRCUIT command of the Network Control Program (NCP) accepted decimal values in the range 1 to 25. In VMS Version 5.2, the maximum value has been increased from 25 to 63 to accommodate an increased diversity in DECnet routing specifications. The default value in VMS Version 5.2 is still 10.

3.34 OPCOM Changes

V5.2

Effective with VMS Version 5.2, there are several changes in the way the operator communications manager (OPCOM) works. These changes are:

- Operator request numbers in a cluster begin at 1 and increase by 1 each time a request is queued. They are not reset to 1 unless the entire cluster is shut down.
- The operator log file can be placed on any disk.
- The operator log file can be enabled or disabled for specific operator classes.
- Operator log files are no longer created on satellite nodes in a cluster by default.
- By defining logical names in the command procedure SYS$MANAGER:SYLOGICALS.COM, the system manager can override the defaults for:
  - Whether an operator is enabled on OPA0:
  - Which operator classes the operator on OPA0: controls
  - Whether an operator log file is opened
  - Which operator classes are recorded in the log file
  - Location and name of the operator log file
- The number of messages sent by OPCOM has been greatly reduced, and the overhead of the OPCOM process is similarly reduced.

3.34.1 Log file Operator Classes

V5.2

The DCL command REPLY /LOG has been enhanced to allow individual operator classes to be selected for inclusion in the log file. When the command REPLY /LOG /ENABLE is used, the classes listed on the /ENABLE qualifier are added to the set currently saved in the log file. If no log file is open when the REPLY /LOG /ENABLE command is used, the log file is opened with the specified classes.

Similarly, when the command REPLY /LOG /DISABLE is used, the classes listed on the /DISABLE qualifier are removed from the set currently saved in the log file. If the REPLY /LOG /DISABLE command removes all operator classes, the log file is closed.

The new feature to allow specification of classes for the log file is controlled by the commands REPLY/LOG/ENABLE=(list-of-classes) or REPLY/LOG/DISABLE= (list-of-classes). When /LOG is added to /ENABLE or /DISABLE, the classes refer to the log file rather than the current terminal. For more information, see the commands REPLY/LOG and REPLY/ENABLE and REPLY/DISABLE in the *VMS DCL Dictionary*.

3.34.2 OPCOM Default States

V5.2

OPCOM has the following default states:

- For all systems except workstations in a VAXcluster:
  - — OPA0: is enabled for all classes.
  - — The log file SYS$MANAGER:OPERATOR.LOG is opened for all classes.
- For workstations in a VAXcluster:
  - — OPA0: is not enabled.
  - — No log file is opened.

3.34.3 Overriding the OPCOM Default States

V5.2

To override the default enabled classes, define the following system logical names in the command procedure SYS$MANAGER:SYLOGICALS.COM :

- OPC$OPA0_ENABLE
  If defined to be true, then OPA0: is enabled as an operator. If defined to be false, then OPA0: is not enabled as an operator. DCL considers any string beginning with *T* or *Y* or any odd integer to be true, all other values are false.
- OPC$OPA0_CLASSES
  This logical defines the operator classes to be enabled on OPA0:. The logical can be a search list of the allowed classes, a list of classes, or a combination of the two, for example:

  ```
  $
  $
  $
  ```

  Note that OPC$OPA0_CLASSES can be defined even if OPC$OPA0_ ENABLE is not defined. In that case, the classes are used for any operators that are enabled, but the default is used to determine whether or not to enable the operator.
- OPC$LOGFILE_ENABLE
  If defined to be true, then an operator log file is opened. If defined to be false, then no log file is opened.
- OPC$LOGFILE_CLASSES
  This logical defines the operator classes to be enabled for the log file. The logical can be a search list of the allowed classes, a comma-separated list, or a combination of the two.

  Note that OPC$LOGFILE_CLASSES can be defined even if OPC$LOGFILE_ENABLE is not defined. In that case, the classes are used for any log files that are opened, but the default is used to determine whether or not to open the log file.

- OPC$LOGFILE_NAME
  This logical supplies information to be used in conjunction with the default name SYS$MANAGER:OPERATOR.LOG to define the name of the log file. If the log file is directed to a disk other than the system disk, commands to mount that disk should be included in the command procedure SYLOGICALS.COM.

Example SYLOGICALS.COM

The following example shows how to disable the SECURITY class messages from being displayed on OPA0:, and also how to disable SECURITY class messages from being saved in the operator log file.

```
$
$
```

Note that since OPC$OPA0_ENABLE or OPC$LOGFILE_ENABLE was not defined, the defaults will determine whether the OPA0 operator is enabled and whether the logfile is opened.

3.34.4 Removing Old Reply Commands — Requirement

V5.2

In previous versions of VMS, to override the default OPCOM states it was necessary to place REPLY commands in system startup files like SYS$MANAGER:SYSTARTUP_V5.COM. This was done by defining SYS$COMMAND to an enabled operator terminal, and then issuing the REPLY command. For example, to disable the SECURITY operator class on OPA0: you could enter:

```
$
$
```

While this technique is still permitted, Digital recommends that all commands of this nature be removed from system startup files, and that the logical names in SYLOGICALS.COM be used to define the desired operator state.

3.34.5 Mixed-Version Cluster Operation with VMS Version 5.1 or Version 5.1-1

V5.2

Some of the new features in Section 3.34 are not available when running in mixed-version clusters with VMS Version 5.1 or Version 5.1-1 systems:

- Mixed-version clusters assign operator request numbers in the old format.

- Since VMS Version 5.1 and Version 5.1-1 systems do not support enabling or disabling individual operator classes for log files, this is not available in mixed-version clusters.
  - The command to enable additional operator classes REPLY /LOG /ENABLE is treated as REPLY /LOG, and causes the current logfile to be closed and a new logfile to be opened.
  - The command to disable operator classes REPLY /LOG /DISABLE is rejected as an illegal operator request.
- The changes to reduce message and CPU overhead are not used in a mixed-version cluster.

It is possible to force OPCOM to run in Version 5.1 mode by setting the bit corresponding to the decimal value 256 in the SYSGEN parameter VMSD1. This causes OPCOM to stay in Version 5.1 mode even when there are no Version 5.1 or Version 5.1-1 nodes in the cluster. You can set this bit on any Version 5.2 node in the cluster to force all Version 5.2 nodes to run in Version 5.1 mode.

OPCOM will, however, automatically sense when a VMS Version 5.1 system is present in the cluster, and automatically enter a mode compatible with Version 5.1. When in V5.1 compatibility mode, newly assigned request numbers will use the old format, and any open operator log files will have all operator classes enabled.

NOTE: Digital recommends that VMSD1 value 256 be set until all Version 5.1 or Version 5.1-1 nodes have been updated to Version 5.2. Although OPCOM will automatically enter V5.1 compatibility mode whenever Version 5.1 or Version 5.1-1 nodes are present, there is a small possibility that two different OPCOM requests could be assigned the same request number due to the different methods of assigning request numbers.

Forcing V5.1 Compatibility Mode Operation

To set the bit to force V5.1 compatibility mode, place the following line in SYS$SYSTEM:MODPARAMS.DAT:

```
VMSD1 = 256
```

When you run AUTOGEN and reboot, this will take effect. Since the SYSGEN parameter is VMSD1 dynamic, you can also set the bit manually with SYSGEN, so that a reboot is unnecessary:

```
$
SYSGEN>
SYSGEN>
SYSGEN>
```

This change will be noticed within five minutes of the change of the parameter setting, and OPCOM will display messages saying that V5.1 compatibility mode is in effect. These messages will be sent to CENTRAL operators, as well as being written to OPA0: directly.

In a large cluster, it is not necessary to set VMSD1 on all nodes in the cluster. It is sufficient to set it to 256 on enough voting nodes so that if quorum is present, at least one voting node has VMSD1 set to 256. It is never necessary to set VMSD1 on non-voting nodes.

3.35 Pseudo Terminal Driver

V5.1

VMS Version 5.1 includes a Pseudo Terminal driver. The components of this driver are PYDRIVER and TWDRIVER. This Pseudo Terminal driver is intended for the exclusive use of DECwindows; any other use of these devices is unsupported.

A future release of DECwindows will discontinue the use of this driver, at which point these images will no longer be shipped as part of VMS.

3.36 Quorum Disks — Notes

The following sections describe changes to the quorum disk environment.

3.36.1 Disk Behavior

V5.0

A quorum disk **watcher** has been created for VMS Version 5.0. The quorum disk watcher accesses the quorum disk and verifies its status to the other nodes in the VAXcluster system. Nodes that have the SYSGEN parameter DISK_QUORUM set to the name of a disk and that are directly connected to that disk (that is, not accessing the disk through an MSCP server) are quorum disk watchers.

When a node that is a quorum disk watcher is removed from a VAXcluster environment, the votes that would be contributed by the quorum disk are not counted towards cluster quorum for up to 4*QDSKINTERVAL. To offset this change, the default value for QDSKINTERVAL has been reduced from 20 to 10 seconds.

Nodes that do not specify the name of a disk for DISK_QUORUM never access the quorum disk. Rather, they rely on a watcher that verifies the status of the quorum disk. When a node that is not a quorum disk watcher is removed from a VAXcluster, the votes that are contributed by the quorum disk continue to count towards the cluster quorum without interruption.

More information on guidelines for configuring and using quorum disks can be found in the *VMS VAXcluster Manual*.

3.36.2 New Configuration Support

V5.0

In addition to the configurations previously supported, quorum disks are now supported in Local Area VAXcluster configurations and mixed interconnect clusters.

3.37 VAX RMS Journaling — Notes

V5.0

The following sections describe information applicable to VAX RMS Journaling Version 1.0. Note that before you can use the VAX RMS Journaling features, you must register the authorization key for journaling on your system. To do this, refer to Section 1.1. You should also read Section 1.10 for additional information.

3.37.1 Accessing Indexed Files on Systems Without VAX RMS Journaling Installed

V5.0

The VMS operating system does not support local (nonnetwork) access to RMS indexed files that were marked for recovery unit journaling, have been modified within a recovery unit, and are unmarked for recovery unit journaling, unless VAX RMS Journaling is installed on that system. This same restriction applies to local access to such files from VAXcluster members on which VAX RMS Journaling has not been installed.

Before you can access an RMS indexed file that has been modified in a recovery unit on a system on which VAX RMS Journaling is not installed, you must make a new copy of the file using the Convert Utility on the system where RMS Journaling is installed. You can then transfer the converted copy of the file to a VMS system where VAX RMS Journaling is not installed and access the file on that system.

Note that this restriction does not apply to network access to such a file from a system on which VAX RMS Journaling is not installed, provided that VAX RMS Journaling is installed on the remote system.

Digital will remove this restriction in a future release of the VMS operating system.

3.37.2 Backup Utility Errors

V5.0

The Backup Utility (BACKUP) cannot save or copy a file marked for recovery unit journaling if the file has active recovery units. If you encounter this problem during a BACKUP operation, you should attempt to access the file using another utility. For example, you can access the file with the DCL command TYPE. This attempt to type the file causes detached recovery to restore records modified during the recovery unit to their states before the recovery unit began. If detached recovery succeeds, the TYPE command succeeds and you can proceed with the BACKUP procedure. If detached recovery fails, the TYPE command fails and detached recovery outputs error messages to the terminal and to the operator communication manager (OPCOM). See the *VAX RMS Journaling Manual* if detached recovery fails.

3.37.3 The Backup Utility and the SET FILE Command

V5.0

Saving and restoring a file marked for recovery unit journaling has a different result from saving and restoring a file marked for after-image or before-image journaling. When you use the Backup Utility to save a file marked for recovery unit journaling, both the BACKUP copy of the file and the restored copy of the file are marked for recovery unit journaling. You do not need to re-mark the restored file for recovery unit journaling with the SET FILE/RU_JOURNAL command.

When you use the Backup Utility to save a file marked for either after-image or before-image journaling, both the BACKUP copy of the file and the restored copy of the file are marked for after-image or before-image journaling, respectively. In both cases, after-image or before-image journaling is disabled by BACKUP. Therefore, you must issue the appropriate SET FILE command to re-mark the restored file for after-image or before-image journaling.

If you use the COPY or CONVERT commands instead of the Backup Utility to copy a file marked for journaling, the destination file will not have the journaling attributes of the source file.

3.37.4 Detached Recovery Improvement

V5.1

Prior to VMS Version 5.1, opening a file marked for recovery unit journaling initiated detached recovery. This was a problem for many sites that opened and closed many files per day.

VMS Version 5.1 improves this situation by initiating detached recovery only when RMS determines that recovery is truly required. This has greatly decreased the time needed to open a file marked for recovery unit journaling.

3.37.5 Errors During the Execution of Recovery Unit Service

V5.0

This section describes the sequence of events that occurs when the $COMMIT_RU, $END_RU, $PREPARE_RU, or $ABORT_RU recovery unit service does not complete successfully. An application program that uses recovery unit journaling may call the $COMMIT_RU recovery unit service explicitly, or it may use the $END_RU recovery unit service, which calls the $PREPARE_RU and $COMMIT_RU recovery unit services. Similarly, an application program can either call the $ABORT_RU recovery unit service explicitly, or the $PREPARE_RU service calls the $ABORT_RU service automatically if the RMS recovery unit handler returns an error during a prepare operation.

The following sequence of events takes place if an error occurs when the $COMMIT_RU or $ABORT_RU recovery unit service is executing:

1 VAX RMS Journaling terminates the process with Bugcheck, and sets the process's final exit status to the following:

```
%RMS-F-BUG_RU_COMMIT_FAIL, recovery unit commit failed
                        or
%RMS-F-BUG_RU_ABORT_FAIL, recovery unit abort failed
```

The *VAX RMS Journaling Manual* discusses error messages output to OPCOM by recovery unit journaling.

2 If accounting is enabled on your system, the final status is also written to the accounting log.

3 VAX RMS Journaling deletes the user process to prevent the process from accessing inconsistent data.

3.37.6 Exclusive Access to Recovery Unit Journaled Files — Restriction

V5.0

You may receive the following unexpected error:

```
%RMS-F-DUP, duplicate key detected (DUP not set)
```

You may receive this message if you attempt to insert or update a record in an indexed file and all of the following conditions are true:

- The file is marked for recovery unit journaling.
- The file has a secondary key that disallows duplicate secondary keys.
- The file is opened for exclusive access.

To prevent this problem, open the file for shared access.

Digital expects to correct this problem in a future release of the VMS operating system.

3.37.7 New File Access Block Field for VAX RMS Journaling (FAB$B_JOURNAL)

V5.0

VAX RMS Journaling has supplied a new field in the file access block (FAB). The FAB defines file characteristics, file access, and certain run-time options. It also indicates whether other control blocks are associated with the file. For more information about the FAB, see the *VMS Record Management Services Manual.*

The FAB$B_JOURNAL field is set by the RMS services $OPEN and $DISPLAY. This field indicates if the opened file is a journal file or whether it is marked for after-image, before-image, or recovery unit journaling.

Table 3–6 lists the bits that RMS may set in this field and their meaning.

Table 3–6 FAB$B_JOURNAL Bit Settings

Bit Offset	Description
FAB$V_AI	The file is marked for after-image journaling.
FAB$V_BI	The file is marked for before-image journaling.
FAB$V_RU	The file is marked for recovery unit journaling.
FAB$V_JOURNAL_FILE	The file is a journal file.

3.37.8 Files Not to Mark for Journaling

V5.0

Digital recommends that you do not mark the following files for journaling:

- SYSUAF.DAT
- JBCSYSQUE.DAT
- MAIL.MAI

If you mark the system authorization file SYSUAF.DAT for journaling and the journal disk becomes full, all further logins will be disallowed.

If you mark JBCSYSQUE.DAT for journaling and the journal disk becomes full, all queue operations will fail.

If you mark the MAIL.MAI file for journaling, you cannot recover the file correctly. This is because the MAIL.MAI file contains the texts of brief mail messages and pointers to files containing longer mail messages. When you recover the MAIL.MAI file, the Recovery Utility will not recover the longer mail messages because they are contained in separate files.

3.37.9 Handling RMS I/O Errors when Journaling

V5.0

RMS operations can fail, issuing unexpected I/O error messages such as RMS$_WER, "file write error," or RMS$_WBE, "error on write behind." If the file is marked for after-image journaling, these error messages mean that the I/O operation to the data file failed. The I/O operation, however, was journaled.

Caution: If you are using both after-image and recovery unit journaling, and an RMS I/O operation fails with an unexpected I/O error message, abort the recovery unit immediately. This restores the data file to a consistent state, and the after-image journal file will be consistent with the data file.

If you are using only after-image journaling, it is not possible to make the data file consistent with the journal file. You can choose to retry the I/O operation at a later time. If the I/O operation is successful, the journal file will contain two copies of the I/O operation, and a recovery operation will result in a consistent data file. The safest procedure, however, is to recover a file marked for only after-image journaling immediately after an I/O error occurs.

3.37.10 Installation Verification Procedure (IVP)

V5.0

After registering the VAX RMS Journaling authorization key, run the VAX RMS Journaling installation verification procedure (IVP) to check whether VAX RMS Journaling is running successfully on your system. The VAX RMS Journaling IVP also serves as an example program and command procedure that uses RMS Journaling. To run the IVP, log in to the system manager's account and enter the following command:

```
$
```

The IVP displays the following information on your terminal screen:

```
$ !+
$ ! RUFEXAMPLE.COM -- Command file to show how to run
$ ! the example program that uses the Recovery Unit
$ ! Services.  This command file is also good for
$ ! verifying the installation of VAX RMS Journaling.
$ !
$ ! NOTE: All file names have dollar signs in them to
$ ! prevent any possible conflict with user file names.
$ ! Of course, any legal file name can be used instead.
$ !-
$ !
$ ! First, delete any old files that may be lying around.
$ ! Ignore error messages here.
$ !
$ SET NOON
$ SET FILE RUF$*.*;*/NOAI/NOBI/NORU_J/RU_F=1/RU_A=0/PROT=OWNER=RWED
%SET-F-SEARCHFAIL, error searching for SYS$SYSROOT:[SYSMGR]RUF$*.*;*
-RMS-E-FNF, file not found
$ DELETE RUF$*.*;*
%DELETE-W-SEARCHFAIL, error searching for SYS$SYSROOT:[SYSMGR]RUF$*.*;*
-RMS-E-FNF, file not found
$ SET ON
$ !
$ ! Initialize RUF$CHECKING.DAT and RUF$SAVINGS.DAT.
$ !
$ CREATE/FDL=SYS$INPUT RUF$CHECKING.DAT
FILE
        ORGANIZATION            indexed
RECORD
        FORMAT                  fixed
        SIZE                    18
KEY 0
        SEG0_LENGTH             9
        SEG0_POSITION           0
$ COPY RUF$CHECKING.DAT RUF$SAVINGS.DAT
$ !
$ ! Mark the files for all sorts of journaling.  We only
$ ! need RU journaling for the program, but AI and BI
$ ! journaling are useful for installation verification.
$ ! We expect the most popular choice to be AI plus RU.
$ !
$ ! NOTE: Since the two files will be participating in
$ ! the same recovery unit, they must both journal to
$ ! the same long term journals.  However, it is OK to
$ ! AI journal to one file and BI journal to another file.
$ !
$ ! NOTE: We will get a warning that our AI journal is
$ ! on the same device as our data file.  Normally one
$ ! should put the AI journal on a different device in
$ ! case the disk is wiped out, but for the purposes
$ ! of installation verification this is acceptable.
$ !
$ SET FILE RUF$CHECKING.DAT/AI=(FILE=RUF$AI.RMS$JOURNAL,CREATE)-
                            /BI=(FILE=RUF$BI.RMS$JOURNAL,CREATE)/RU_J
%SET-W-INVAIJDEV, after-image journal SYS$SYSROOT:[SYSMGR]RUF$AI.RMS$JOURNAL;1
is on same device as data file SYS$SYSROOT:[SYSMGR]RUF$CHECKING.DAT;1
$ SET FILE RUF$SAVINGS.DAT /AI=FILE=RUF$AI.RMS$JOURNAL-
                            /BI=FILE=RUF$BI.RMS$JOURNAL/RU_J
%SET-W-INVAIJDEV, after-image journal SYS$SYSROOT:[SYSMGR]RUF$AI.RMS$JOURNAL;1
is on same device as data file SYS$SYSROOT:[SYSMGR]RUF$SAVINGS.DAT;1
$ !
```

```
$ ! Back up the files.  Only done after marking for journaling.
$ !
$ BACKUP/RECORD RUF$CHECKING.DAT RUF$CHECKING.BCK
%BACKUP-I-STARTRECORD, starting backup date recording pass
%BACKUP-I-MODOUTAI, RMS after-image journaling disabled on saved copy of
_DUA0:[SYS0.SYSMGR]RUF$CHECKING.DAT;1
%BACKUP-I-MODOUTBI, RMS before-image journaling disabled on saved copy of
_DUA0:[SYS0.SYSMGR]RUF$CHECKING.DAT;1
$ BACKUP/RECORD RUF$SAVINGS.DAT  RUF$SAVINGS.BCK
%BACKUP-I-STARTRECORD, starting backup date recording pass
%BACKUP-I-MODOUTAI, RMS after-image journaling disabled on saved copy of
_DUA0:[SYS0.SYSMGR]RUF$SAVINGS.DAT;1
%BACKUP-I-MODOUTBI, RMS before-image journaling disabled on saved copy of
_DUA0:[SYS0.SYSMGR]RUF$SAVINGS.DAT;1
$ !
$ ! Test RU journaling by running the program.
$ ! The checking balance should be $90 and the savings $110.
$ ! If the program were interrupted, the balances would be restored.
$ !
$ RUN SYS$EXAMPLES:RUFEXAMPLE
Pausing for five seconds.
Checking account balance is        $90.00
Savings account balance is        $110.00
$ !
$ ! Test AI journaling:
$ ! Roll the RUF$CHECKING backup file forward and check that
$ ! it matches the current state of the RUF$CHECKING data file.
$ ! (There should be 0 differences encountered.)
$ !
$ RECOVER/FORWARD RUF$CHECKING.BCK
$ DIFFERENCES RUF$CHECKING.BCK RUF$CHECKING.DAT
Number of difference sections found: 0
Number of difference records found: 0
DIFFERENCES /IGNORE=()/MERGED=1-
    SYS$SYSROOT:[SYSMGR]RUF$CHECKING.BCK;1-
    SYS$SYSROOT:[SYSMGR]RUF$CHECKING.DAT;1
$ !
$ ! Test BI journaling:
$ ! Roll the RUF$SAVINGS data file backward and check that it
$ ! matches the original state of the RUF$SAVINGS data file.
$ ! (There should be 0 differences encountered.)
$ !
$ RECOVER/BACKWARD RUF$SAVINGS.DAT
$ DIFFERENCES RUF$SAVINGS.BCK RUF$SAVINGS.DAT
Number of difference sections found: 0
Number of difference records found: 0
DIFFERENCES /IGNORE=()/MERGED=1-
    SYS$SYSROOT:[SYSMGR]RUF$SAVINGS.BCK;1-
    SYS$SYSROOT:[SYSMGR]RUF$SAVINGS.DAT;1
$ !
$ ! Cleanup.  Ignore error messages here.
$ !
$ SET NOON
$ SET FILE RUF$*.*;*/NOAI/NOBI/NORU_J/RU_F=1/RU_A=0/PROT=OWNER=RWED
$ DELETE RUF$*.*;*
$ SET ON
$ IF V .EQ. 0 THEN $ SET NOVERIFY
```

If VAX RMS Journaling is enabled on a common system disk where SYS$MANAGER is defined to be a search list of directories, the Backup Utility issues a warning message immediately after each BACKUP/RECORD command as follows:

```
$BACKUP/RECORD RUF$CHECKING.DAT RUF$CHECKING.BCK
%BACKUP-W-NOFILES, no files selected from SYS$COMMON:RUF$CHECKING.DAT;*
   .
   .
   .
$BACKUP/RECORD RUF$CHECKING.DAT RUF$SAVINGS.BCK
%BACKUP-W-NOFILES, no files selected from SYS$COMMON:RUF$SAVINGS.DAT;*
```

These BACKUP commands succeed even though you receive warning messages. These warning messages indicate that BACKUP did not find a copy of RUF$CHECKING.DAT and RUF$SAVINGS.DAT in each directory to which the search list of directories points.

3.37.11 Mount Utility Creates the Logical Name DISK$volume_label

V5.0

A volume label is the only device-independent identifier for a VMS volume or volume set. VAX RMS Journaling uses a volume label and a file ID as the forward pointer from a data file to its journal file. In order for RMS to obtain the device name from the volume label, it is necessary that an executive-mode concealed logical name, DISK$volume_label, be defined for the device in which the volume is mounted.

In Version 1.0 of VAX RMS Journaling, if a logical name for the disk was specified when the disk was mounted with the /SYSTEM or /CLUSTER qualifier, the executive-mode concealed logical name DISK$volume_label was not created. VMS Version 5.0 creates the executive-mode concealed logical name DISK$volume_label when a disk is mounted with either the /SYSTEM or /CLUSTER qualifier, even if a logical name for the disk is specified. For example, the executive-mode concealed logical name DISK$FINANCE_DISK is created if you mount the disk with any of the following MOUNT commands:

```
$
$
$
$
```

If you mount a disk with the /GROUP qualifier, or as a private volume MOUNT creates a supervisor-mode concealed logical name. For VAX RMS Journaling to work properly on a volume mounted with the /GROUP qualifier or as a private volume, you must define an executive-mode concealed logical name as follows:

```
$
_Log name:
_Equ name:
```

3.37.12 SET FILE/AI_JOURNAL or SET FILE/BI_JOURNAL Command

V5.0

If you use the SET FILE/AI_JOURNAL or the SET FILE/BI_JOURNAL command without the CREATE keyword and you specify a journal file that is already being used, the SET command cannot open the journal file. The SET command issues the FLK error message (file currently locked by another user). The SET FILE command does not allow you to re-mark a file for journaling using the same journal file specification without the CREATE keyword.

If you want to create a journal file with the same name as a previously created journal file, use the CREATE keyword with the SET FILE/AI_JOURNAL or the SET FILE/BI_JOURNAL command.

The following example illustrates how to create a journal file with the same name as a previously created journal file:

```
$

$
$
```

3.37.13 VFC Format Sequential Files Partially Supported for Before-Image or Recovery Unit Journaling

V5.0

You cannot execute an $UPDATE on variable fixed-length control (VFC) sequential files when using before-image or recovery unit journaling. The VFC sequential file format is indicated by the symbolic value FAB$C_VFC in the FAB$B_RFM field of the FAB. The following error condition results if you attempt to execute an $UPDATE on a VFC format sequential file marked for before-image journaling, or on a VFC format sequential file modified within a recovery unit.

```
JNS, operation not supported by RMS journaling
```

For more information about this error message see the *VAX RMS Journaling Manual*.

Digital expects to remove this restriction in a future release of the VMS operating system.

3.37.14 WRTJNL_BIJ Error Message

V5.0

The WRTJNL_BIJ error message may return a zero completion status value (STV) rather than the message DEVICEFULL if the device on which the before-image journal file resides becomes full when RMS is trying to write to the before-image journal file. If you receive a zero STV in this situation, submit a Software Performance Report (SPR). The *VAX RMS Journaling Manual* lists the information you need to include with RMS Journaling-specific SPRs.

3.38 RQDX3 Controller

The following notes pertain to the RQDX3 controller.

3.38.1 Device Unit Number Changed with RQDX3 Controllers

V5.0

An error involving served satellite disks that change the device unit number of a disk by setting the high bit has been discovered. When this occurs, the disk cannot be accessed using the original device unit number. However, you can access the disk using the new unit number (old unit number + 128).

This problem occurs when the following conditions are present:

- Disks are accessed through the MSCP server
- Very large files are created (for example, creating paging and swapping files that are larger than 20,000 blocks.)
- Highwater marking is present

You can correct this problem with either of the following solutions:

1 Do not create large files from a remote node when highwater marking is present. Instead, you should do this on the local node. If you are creating paging and swappping files, create them using their minimal sizes and boot them on your local node. Then, you can make them larger on the local node.

2 Do not use highwater marking when creating large files over a served network path. Turn off the highwater marking and then create the file. This will prevent sending of the MSCP command ERASE. Thus, the file will be allocated but not zeroed.

3.38.2 Restriction for RQDX3 Controllers

V5.0

If you are using RQDX3 controllers on a system that serves disks in a Local Area VAXcluster or a Mixed-Interconnect cluster, and the RQDX3 controller does not contain a microcode revision level of 3.0 or later, you may see frequent controller resets. If your error log shows frequent controller resets during satellite booting, you should contact your local Digital Field Service representative to obtain the latest microcode.

You can determine the controller type and microcode revision level by entering the command ANALYZE/ERROR_LOG.

3.39 Security Review Recommended

V5.2

Digital strongly recommends that all site security administrators take the following steps to improve security on existing VMS systems:

- Disable or remove all of the Digital default accounts, except for SYSTEM, in any active SYSUAF.DAT files, unless you have an explicit need for these accounts. The default accounts are FIELD, SYSTEST, and SYSTEST_CLIG. Previous versions of VMS also included accounts named USER and USERP, which you should remove if they are present on your system. This is critically important if you have used the VMSKITBLD.COM command procedure to build alternate system disks, which will have a SYSUAF.DAT file containing the default Digital username/password combinations.
- Seriously consider using the password generator for all privileged accounts at sites with medium to high security needs.

Network security managers should also consider the following additional steps:

- Review all network proxies and eliminate, if at all possible, any proxies into privileged accounts.
- Remove the default DECnet account and substitute separate accounts for each network object required for a particular node. Use generated passwords for these accounts.
- Review the *Guide to VMS System Security*. This manual has been significantly enhanced to describe all of the new features present in VMS Version 5.2.

3.40 Security Features — Changes

The following sections describe new or changed system security features.

3.40.1 Department of Defense (DoD) Erase Pattern Fixed

V5.2

VMS Version 4.0 introduced the $ERAPAT system service to allow sites to generate their own erase patterns. DIGITAL supplies an example MACRO source file that contains the Department of Defense (DoD) erase patterns for memory, disk, and tape. Included in this file are instructions on how to assemble and link the module and how to install the resulting ERAPATLOA.EXE system loadable image in the directory SYS$LOADABLE_IMAGES.

VMS Version 5.2 contains several fixes that correct all known problems that relate to using a loadable, non-zero, erase pattern. Submit a new Software Performance Report (SPR) should you encounter any further problems with this service.

3.40.2 NETCONFIG.COM Security Enhancements

V5.2

In VMS Version 5.2, the DECnet network configuration command procedure, NETCONFIG.COM, has been enhanced to provide several options for limiting default access to your system. A new command procedure for existing networked systems, NETCONFIG_UPDATE.COM, has been created for the same purpose.

Previously, NETCONFIG.COM created one default account named DECNET. That account provided default access to all Digital-supplied objects and user-written applications that were not restricted by other forms of access control, such as proxy accounts and access control strings. That type of default access is appropriate only for systems with very low security requirements.

Now, in place of the default account DECNET, individual accounts for the following Digital-supplied objects can be created:

- MAIL
- File access listener (FAL)

- PHONE
- Network management listener (NML)
- Loopback mirror (MIRROR)
- VMS Performance Monitor (VPM)

These accounts restrict default access to their respective objects. Therefore, a system manager can enable default access for those objects that are appropriate for the system and the network. In addition, logs can be produced for these accounts so that the usage of these objects can be monitored.

Previously, to create these accounts, it was necessary to specify several commands for each. Now, using NETCONFIG.COM (or NETCONFIG_UPDATE.COM), you can create an account for a Digital-supplied object by simply responding YES to the respective prompt.

For more information about the new options, refer to the *VMS Version 5.2 New Features Manual*. For more information about network security, refer to the *Guide to VMS System Security*.

3.40.3 OPCOM Started by Default

V5.2

Prior to VMS Version 5.2, the OPCOM process was not started by default for nonclustered MicroVAX systems. Beginning with VMS Version 5.2, OPCOM is started for all VMS systems regardless of hardware configuration. If your installation does not require security auditing, OPCOM (and the AUDIT_SERVER) can be disabled through the SYSMAN utility. See the *VMS Version 5.2 New Features Manual* for complete instructions.

Note: Sites that desire security auditing should not disable OPCOM, since the current implementation of security auditing requires that OPCOM be present. Digital intends to remove this requirement in a future release of VMS.

3.40.4 Re-establishing Security Environment

V5.2

The VMS Version 5.2 upgrade provides new files and directories under [VMS$COMMON...]. If you had any special protections and ACLs before the upgrade, you need to reapply them to re-establish your previous security environment.

3.40.5 Security Audit Alarm Settings Preserved Between System Boots

V5.2

In prior versions of VMS, the classes of security-auditing alarm events had to be set each time a system was booted (typically in the site-specific startup command procedure SYSTARTUP_V5). Beginning with VMS Version 5.2, the security alarm settings are preserved between boots in the permanent audit server database SYS$MANAGER:AUDIT_SERVER.DAT.

Following the VMS Version 5.2 upgrade, you should remove any SET AUDIT/ALARM commands from your site-specific startup command procedure.

Note: The SET AUDIT/FAILURE_MODE setting is currently not saved in the audit-server database and must be reset after each system boot. Digital intends to remove this restriction in a future release of VMS.

3.40.6 Security Auditing — Enhancements

V5.2

There have been a number of enhancements to the security-auditing subsystem for VMS Version 5.2, including the following:

- Introduction of a separate security-auditing server process (AUDIT_SERVER)
- Creation of a cluster-wide, public-format, binary audit log
- New Audit Analysis Utility
- Addition of integrated resource monitoring of disk space and virtual memory associated with the system security-audit log file
- Transparent remote archive capability
- Real-time audit analysis
- Improved overall auditing performance

In addition, as part of the VMS Version 5.2 upgrade, authorization and break-in security alarms are enabled on all systems. Also, the AUDIT alarm event class is permanently enabled. Any use of the SET AUDIT/ALARM command results in the generation of a system security alarm that cannot be overridden. If you do not desire security auditing for your installation, refer to the *VMS Audit Analysis Utility Manual* for information on how to disable these events.

Resource monitoring of free disk space associated with the system security audit journal file (normally, the system volume) also will be enabled as part of the VMS Version 5.2 upgrade. Refer to the *Guide to VMS System Security* for further information on the various modes of resource monitoring.

3.40.7 SYSECURITY.COM Command Procedure — New Site-Specific Configuration File

V5.2

The VMS Version 5.2 installation and upgrade procedures create an empty SYSECURITY.COM command procedure, which is run prior to starting up the security-auditing server process. If you wish to direct your system security-audit journal file SECURITY_AUDIT.AUDIT$JOURNAL or audit-server database AUDIT_SERVER.DAT in the SYS$MANAGER directory to a disk other than the system disk, specify the command to mount the alternate disk in this file. This ensures that the alternate disk is mounted before the audit server process is started.

In addition to using the procedure SYSECURITY.COM to mount disks, you can use it to define the system logical name AUDIT_SERVER (to relocate the audit-server database file) or to define system logical names needed to resolve either the system security audit journal or system archive file's destination. For example, the following lines in the procedure SYSECURITY.COM mount the disk 254DUA118 and redirect the audit server permanent database to the alternate volume:

```
$ if .not. f$getdvi("$254$dua118","mnt") -
  then mount/system $254$dua118 audit audit$ /norebuild
$ define/system/exec audit_server audit$:[audit]audit_server.dat
```

3.40.8 SYSUAF Template File — Change

V5.2

The file SYS$SYSTEM:SYSUAF.TEMPLATE is used by the VMS installation procedure to initially create the System User Authorization File SYS$SYSTEM:SYSUAF.DAT. (You can also use the template file to create a new User Authorization File (UAF) file on your system, using the DCL command COPY SYS$SYSTEM:SYSUAF.TEMPLATE SYS$SYSTEM:SYSUAF.DAT.) Beginning with VMS Version 5.2, all accounts in the SYSUAF template file, except for the SYSTEM account, are disabled when shipped, by setting the flag /FLAG=DISUSER.

Additionally, all of the account passwords in the template file are set as expired, and the password lifetime on the DEFAULT account has been lowered from 180 days to 90 days. Digital recommends a maximum password lifetime of 90 days for unprivileged accounts, and a maximum lifetime of 30 days for privileged accounts.

3.40.9 Suppressing Duplicate Logging of Security Alarms by OPCOM

V5.2

VMS Version 5.2 differentiates between a security alarm and a security audit in the security auditing software.

A **security alarm** is a real-time event that is broadcast to security operator terminals by the operator communication manager (OPCOM). Security alarms are not necessarily recorded onto permanent media (e.g., disk or tape), although the site-security administrator may choose to do so.

A **security audit** is a security event that is logged by the audit server process (AUDIT_SERVER) directly to the system security audit log file (SYS$MANAGER:SECURITY_AUDIT.AUDIT$JOURNAL). Security audits are never displayed on security operator terminals.

In a future release of VMS, the site-security administrator will be able to control whether a given system-security event generates an alarm, an audit, or both.

For compatibility with previous releases, VMS Version 5.2 currently propagates all system-security events as both a system alarm and a system audit. Alarms are broadcast to all SECURITY class operators, and audits are logged in the system security audit journal file.

By default, OPCOM logs all SECURITY class messages in the operator log file, as in earlier releases. Because these entries now duplicate the entries in the system security audit log, to conserve disk space you might want to disable the SECURITY class in the operator log file by issuing the command:

```
$
```

See Section 3.34.3 for the recommended method of disabling an operator class each time the system is initialized.

Note: Because the audit server process directs important messages to both the CENTRAL and SECURITY class operators, disabling the SECURITY class will not prevent the receipt of critical information (as in previous releases).

3.40.10 User Authorization File (UAF) Notes

The following sections pertain to the User Authorization File (UAF).

3.40.10.1 Changes to the CAPTIVE Flag

V5.2

The User Authorization File (UAF) flag CAPTIVE has always been intended for accounts that perform individual (often privileged) functions (like BACKUP/RESTORE) or for accounts tied to a menu system (often referred to as *turn-key* accounts). Typically, these accounts operate in a restricted environment and do not allow the user complete access to the command language interpreter (DCL).

However, the security of a captive account—an account with the CAPTIVE flag set—has depended on the captive command procedure under which the account runs. Prior to VMS Version 5.2, a captive command procedure was only truly captive if the author of the command procedure exactly followed the guidelines in the *Guide to VMS System Security*.

UAF Flag CAPTIVE — New Interpretation

Beginning with VMS Version 5.2, the UAF flag CAPTIVE has been enhanced to make writing captive command procedures easier and to increase the security of systems using captive command procedures that do not follow the guidelines in the *Guide to VMS System Security* exactly. The enhancements include the following:

- Accounts with the CAPTIVE flag set no longer have direct access to DCL. Command procedures that terminate to DCL (for example, as a result of an unhandled error or of pressing <CTRL/Y>) now result in the error message CAPTINT and deletion of the process under which the procedure runs.
- Additionally, the INQUIRE command has been disabled for accounts with the CAPTIVE flag set. You must use the READ/PROMPT command instead. Using the INQUIRE command in a captive command procedure produces the error CAPTINQ which, if unhandled by a previous ON declaration, results in the error CAPTINT and deletion of the process.

For a complete list of restrictions imposed by the CAPTIVE flag, see the *Guide to VMS System Security*.

New UAF Flag RESTRICTED

Digital recognizes that these changes in the CAPTIVE flag may harm existing captive command procedures that depend on the behavior prior to Version 5.2 (see the subsection "Possible Incompatibilities with New Interpretation of CAPTIVE Flag"). Digital also recognizes that the previous behavior does have value in some situations – namely, to force the execution of a set of command procedures, after which the user is allowed normal access to DCL.

Therefore, the security restrictions formerly denoted by the CAPTIVE flag have been moved to a new UAF flag called RESTRICTED. Accounts in which the RESTRICTED flag is set obey all of the restrictions that were formerly implied by CAPTIVE. Future security enhancements in the area of captive accounts will most likely be tied to the CAPTIVE flag only.

Note: In a future release of VMS, Digital also intends to remove the SPAWN restrictions from the RESTRICTED flag. At such time, VMS will not treat RESTRICTED accounts differently than normal accounts once the login sequence has been completed. Customer written software should not use the RESTRICTED flag to prevent access to either the SPAWN command or the LIB$SPAWN RTL routine.

STARLET Symbol UAI$V_CAPTIVE — Value Change

VMS Version 4.4 introduced the system services $GETUAI and $SETUAI to provide a system service interface to the System User Authorization File (SYSUAF). These services allow privileged routines to retrieve and to modify information contained in any user's authorization record. The interface to these services uses the $UAIDEF symbols defined in the file STARLET.MLB.

Because of the change in interpretation of the UAF flag CAPTIVE in VMS Version 5.2, it has been necessary to change the value of the public symbol UAI$V_CAPTIVE, for two reasons:

- To allow Version 5.2 nodes to coexist securely with Version 5.1 nodes in the rolling upgrade environment
- To ensure that under Version 5.2, UAF records marked CAPTIVE will take on the additional restrictions by default

UAI$V_RESTRICTED is the new name for the bit that was formerly UAI$V_CAPTIVE. The symbol UAI$V_CAPTIVE still exists, but now defines a previously unused bit in the UAF flags longword. The following table shows the new values (in decimal radix) for these two symbols:

Symbol	New Value
UAI$V_RESTRICTED	3
UAI$M_RESTRICTED	8
UAI$V_CAPTIVE	16
UAI$M_CAPTIVE	65536

The new UAI$V_RESTRICTED flag functions exactly as the old CAPTIVE flag did prior to VMS Version 5.2. The new CAPTIVE flag has the following additional features:

- Returning direct command to DCL is not allowed
- Use of the DCL verb INQUIRE is not allowed

Captive command procedures that violate either of these new restrictions cause the process to be deleted with an appropriate error message. Please see the *VMS Version 5.2 New Features Manual*, the *Guide to VMS System Security*, and Section 3.40.10.1 of this document for a complete description of this change and for an exact description of the effect of these two flags.

Images that were linked prior to VMS Version 5.2 will continue to function normally; however, they will be manipulating the RESTRICTED bit (as viewed from AUTHORIZE, for example).

Programmers who wish to take advantage of the new behavior of CAPTIVE must recompile and relink from source.

Many Digital language products make the STARLET library definitions available to programmers. After installing VMS Version 5.2, you will need to reinstall these language products so that they reflect the changes to the STARLET definitions. For further details, refer to the installation guides for the language products that you have installed.

Some Digital language products include their own support for STARLET in a separate environment file that is not based on the STARLET library that is shipped with VMS. As a result, these language products will not automatically recognize the change in the definition of UAI$V_CAPTIVE, even if they are reinstalled.

For these layered products, you will have to override the definition of UAI$V_CAPTIVE (using the value shown in the preceding table) until the next release of the language product that includes support for the VMS Version 5.2 STARLET environment.

Possible Incompatibilities with New Interpretation of CAPTIVE Flag

During the initial stages of the V5.2 upgrade, all user accounts which were previously marked CAPTIVE will be modified to use the new UAF flag RESTRICTED instead.

The upgrade procedure also scans the UAF and marks all accounts which are then set RESTRICTED to also be set CAPTIVE. This ensures that all accounts which were previously marked CAPTIVE are fully secured under VMS V5.2.

However, as a result of this change, those accounts that specifically rely on the old behavior of CAPTIVE will no longer function correctly. The most common type of account likely to be affected by this change are network server accounts created by Digital and third-party layered products.

Note: Network server accounts that are defined in the permanent network object database (NETOBJECT.DAT) prior to the VMS V5.2 upgrade will not be modified by the upgrade and should continue to work without modification.

Problems specifically related to the new interpretation of CAPTIVE can be diagnosed by enabling the PROCESS accounting class. These problems typically manifest themselves as a "Network Partner Exited" message on the node that initiates a DECnet connection and as either a CAPTINT or CAPTINQ error on the remote node.

You can use the following command to locate all process termination records due to either a CAPTINT or CAPTINQ error:

```
$
```

Affected accounts can be modified to restore the old behavior by executing the following commands from a suitably privileged account:

```
$
$
UAF>
UAF>
```

Digital strongly recommends that the site-security administrator carefully review the relevant sections in the *VMS Version 5.2 New Features Manual* and the *Guide to VMS System Security* before clearing the CAPTIVE flag on any account. Indiscriminantly clearing the CAPTIVE flag could compromise the security of the captive account.

Table 3–7 lists VMS layered products that supply CAPTIVE accounts. Following a new installation of any of these layered products, you may need to use AUTHORIZE to modify these accounts to use the RESTRICTED flag.

Table 3–7 VMS Layered Product Captive Accounts

Product Name	Account Name
DECnet/SNA Data Transfer Facility	SNADTF
DECnet/SNA Gateway for Channel Transport	SNA$CSV
DECnet/SNA Gateway for Synchronous Transport	SNA$CSV
DECnet/SNA Gateway	SNACSV
Ethernet Teminal Server	PLUTO
VAX DEC/MAP	FTAMNET
VAX DEC/MAP	OSI
VAX FTSIE	FTSIE

(continued on next page)

Table 3–7 (Cont.) VMS Layered Product Captive Accounts

Product Name	Account Name
VAX Message Router	DDSNET
VAX Message Router	MRNET
VAX Notes	NOTES$SERVER
VAX OSI Transport Service	OSIT$DEFAULT
VAX RDB/VMS	RDB$REMOTE
VAX/VMS Service for MS-DOS	PCFS$ACCOUNT

Future releases of each of these layered products will correct this problem and alleviate the need for you to modify any of these accounts following the layered product installation.

3.40.10.2 DISIMAGE Flag

V5.2

Many sites effect their security by constructing alternate DCL command tables. Often these tables are used in conjunction with captive accounts to restrict the verbs (and images) you can access.

The User Authorization File (UAF) flag DISIMAGE prevents the execution of arbitrary user-written images by disabling the RUN and MCR verbs and the foreign command mechanism in DCL. Accounts in which this flag is set can only execute commands defined in their command table CLITABLES.

The DISIMAGE flag is intended to be used in combination with the RESTRICTED or DEFCLI UAF flags, which prevent the user from selecting an alternate CLI or CLITABLES using the /CLI or /TABLES qualifiers at login.

Note: If you, as a restricted user, retain access to the SET COMMAND verb, you can still run arbitrary images by using the Command Definition Utility to create your own verbs. Site security administrators must take this into account when creating captive environments. See the *Guide to VMS System Security* for more information on the DISIMAGE flag.

3.40.10.3 UAF Record Length Enforcement

V5.2

Beginning with VMS Version 5.2, checks have been added to the $GETUAI and $SETUAI system services to detect UAF records that are shorter than the UAF minimum record length (UAF$K_FIXED). If the requested record is less than the minimum value, the services return an SS$_ACCVIO error code.

System programmers should ensure that any user-written software that accesses the UAF directly correctly maintains the minimum UAF record length.

Note: The format of the UAF record and the way in which the system modifies it is subject to change in future versions of VMS. Digital does not support direct access to the UAF.

3.40.10.4 UAF Template File Changes

V5.2

The following changes have been made to the UAF template file (SYS$SYSTEM:SYSUAF.TEMPLATE) for VMS Version 5.2:

- The UAF parameter PWDLIFETIME for the DEFAULT account has been lowered from 180 days to 90 days
- The UAF parameter PWDLIFETIME for the accounts FIELD, SYSTEM, SYSTEST, and SYSTEST_CLIG has been lowered from 90 days to 30 days
- The DISUSER flag has been set for the accounts DEFAULT, FIELD, SYSTEST, and SYSTEST_CLIG

Because the DEFAULT account serves as a template for the ADD command in AUTHORIZE, site-security administrators may wish to clear the DISUSER flag on the DEFAULT account using the following commands:

```
$
$
UAF>
```

If you do not clear the DISUSER flag on the DEFAULT account, accounts created by some layered product installations may not function correctly. Depending upon the layered product installation procedures, this may in turn cause the Installation Verification Procedure (IVP) for the product to fail. If this occurs, clear the DISUSER flag on the DEFAULT account and reinstall the layered product.

3.41 SET HOST (CTDRIVER/REMACP/RTPAD) — Notes

The following section describes some of the behaviors of the SET HOST facility prior to VMS Version 5.2 and some additional features that are provided by VMS Version 5.2.

3.41.1 $CANCEL is Asynchronous

V5.2

The $CANCEL system service performs an asynchronous cancel operation. This means that the application *must* wait for each I/O operation issued to the driver to complete prior to checking the status for that operation.

Prior to VMS Version 5.0, the VMS terminal driver performed synchronous cancel operations. This was a result of the way VMS was scheduled on asymmetric multiprocessing (ASMP) machines and uniprocessors, not a design feature of the VMS terminal driver. Starting with VMS Version 5.0, in an SMP environment the VMS terminal driver completes these cancel operations asynchronously.

CTDRIVER has always completed the cancel operations asynchronously. If a read request is pending and CTDRIVER receives a cancel request, CTDRIVER queues a message to be sent to the remote system, telling it to cancel the read request. Control then returns to the application calling $CANCEL. Eventually, CTDRIVER sends the message to the remote system. The remote system then sends any data received prior to the

cancel request back to CTDRIVER. Once CTDRIVER receives this data, the pending read request can be canceled. For slow networks, this could take *seconds* before the pending read request is actually canceled.

3.41.2 CTDRIVER Enforces SETMODE/SENSEMODE Buffer Size

V5.2

CTDRIVER now enforces the SETMODE/SENSEMODE buffer size. The valid buffer sizes are either 8 or 12 bytes with the default being 8 bytes. If any other buffer values are specified then CTDRIVER returns the status code SS$_BADPARAM.

3.41.3 CTDRIVER's Output Buffering

V5.2

The SET HOST facility behaves differently from the VMS terminal driver in that it buffers output data from the program that is executing. This occasionally causes a perception problem for the user when the program is aborted with a CTRL/C, a CTRL/Y, or an out-of-band abort character. The user expects the program to abort immediately and the display to stop immediately. The components of the SET HOST facility try to preserve the feel of the VMS terminal driver by stopping the display as soon as the abort character is entered, which can discard a significant amount of program output.

When running between two VMS systems, the SET HOST facility is made up primarily of two elements: RTPAD (local VAX node), and CTDRIVER (remote VAX node). Both elements perform output buffering to enhance performance when using wide area networks. CTDRIVER performs the initial buffering, queues the buffers to be output later, and returns a successful write status. The user sees the results of the executing program only as the buffers of output data are displayed. This buffering causes a perception problem for the user when large sections of output are displayed on the local terminal, since the output displayed on the user's terminal is typically far behind the output generated by the application. The user is led to believe that the application is executing at the exact point that is producing the current display, when in reality the remote program can have completed execution.

The delay between executing an application and displaying its output leads to several anomalies in the effects of CTRL/C, CTRL/Y, and out-of-band abort characters.

3.41.3.1 Output Line Not In Sequence Following CTRL/C, CTRL/Y, or an Out-of-band Abort Character

V5.2

After you enter a control character that causes input and output to be aborted, it is possible to get one more line of output data. This occurs when the application program calls $QIO (directly or indirectly through RMS or language support routines) to output data to a buffer at the same time that the control character is entered.

When CTDRIVER receives the abort character (CTRL/C, CTRL/Y, or an out-of-band abort character) from the network, it flushes the current output buffers and aborts any pending read operations. However, if the application program has just called $QIO with a write operation, that $QIO write data is still buffered, and then displayed. That data may not be the next output in sequence from the user's point of view, since all the previous output buffers in CTDRIVER were flushed and the data in them was not displayed.

When not using the SET HOST facility, the effect of an abort character on the display is different, because the VMS terminal driver does not buffer output from the application that is executing. If the application program has just called $QIO with a write operation when the abort character is entered, then this $QIO write data is displayed. Because all write operations are sequential and do not complete until the output is actually displayed, the additional line displayed is in sequence. Since there is no break in the data, the user normally will not notice that there is an additional line.

3.41.3.2 Extra Input Prompt Displayed Following CTRL/C, CTRL/Y, or an Out-of-band Abort Character

V5.2

After you enter a control character that causes input and output to be aborted, the system may display more than one input prompt. This occurs when you are connected to a system running a VMS version prior to VMS Version 5.2, because the older CTERM protocol does not synchronize RTPAD and CTDRIVER after the abort occurs.

VMS Version 5.2 extends the CTERM protocol for VMS-to-VMS connections to allow CTDRIVER to synchronize with RTPAD before displaying any more data on the terminal. The extra read prompt is not displayed, a behavior that better emulates the VMS terminal driver.

Note: This problem is fixed only between VMS Version 5.2 systems. If SET HOST is used between VMS Version 5.2 and older versions of VMS, the extra read prompt is still displayed.

3.41.3.3 CTRL/C, CTRL/Y, and Out-of-band Abort Character Processing

V5.2

In VMS versions prior to VMS Version 5.2, if an application had a read operation pending and had queued a CTRL/C, CTRL/Y, or out-of-band abort character AST, it was possible for the application to queue additional read requests unknowingly after receiving the abort character AST, but before the read operation was aborted. This could occur because the CTERM protocol delivered two separate messages to CTDRIVER, the first describing the abort character and the second describing the aborted read operation. These events were sent to the application as soon as they were received by CTDRIVER, but they could be separated by several seconds when they reached the application, due to network delays in the CTERM protocol. If the application assumed that the read operation had completed when it received the abort character AST and responded by requesting an additional read, then multiple read requests would be outstanding, creating confusing displays and responses for the user.

Note: Such an assumption is really an error in the application program.

In VMS Version 5.2, this behavior has been changed. CTDRIVER does not deliver the abort character AST until it receives the second message; CTDRIVER then delivers both events, the abort character AST followed by the read completion event. The read status should be set very shortly after the abort-character AST is delivered to the application, and before the application has time to issue a new read request. Note, however, that these are still two asynchronous events, and that the application must still synchronize with the completing read operation.

3.41.3.4 Captive Command Procedures and CTRL/Y

V5.2

A side effect of buffering by CTDRIVER is the perception of how CTRL/Y works when using the SET HOST facility. Since CTDRIVER and RTPAD are trying to emulate the VMS terminal driver, the current read operation and all pending write operations are aborted when the user enters CTRL/Y. However, the pending write operations also include all the buffered output that should have been displayed before the CTRL/Y was entered, but due to the buffering was not.

The effect of the buffering can be especially confusing if a CTRL/Y is entered when a captive command procedure is executing. During execution of captive command procedures, DCL has a CTRL/Y pending. When this AST is delivered, DCL only re-enables it; no other action is performed. In that case, if the program being executed only performs output, it appears that the program was aborted by the CTRL/Y. Actually, the program completed execution before the CTRL/Y was entered, and the CTRL/Y merely discarded all the buffered output.

3.41.4 CTRL/C Processing

V5.2

Application programs that use both the CTRL/C asynchronous system trap (AST) and a CTRL/C out-of-band AST may receive CTRL/Y ASTs as well when using the SET HOST facility. This occurs when the first CTRL/C character is entered from the keyboard, following the declaration of both AST routines.

The problem arises because the command terminal protocol CTERM only supports a single function per control character. The CTERM function code for CTRL/C can be encoded to represent either a CTRL/C out-of-band AST or a CTRL/C AST, but not both. If the CTRL/C AST is declared prior to the CTRL/C out-of-band AST, then RTPAD removes the CTRL/C AST.

A feature in RTPAD allows you to work around the CTERM protocol limitation by always declaring the CTRL/C out-of-band AST (IO$M_ INCLUDE bit set) before declaring the CTRL/C AST. This places RTPAD in the correct state to handle both ASTs, thus allowing correct input to the application program.

3.41.5 New REMACP Image and RTTLOAD Command File

The remote I/O ancillary control process image REMACP and the command file RTTLOAD.COM have been rewritten to provide more flexibility.

3.41.5.1 Setting the Maximum Number of Remote Users

V5.2

The maximum number of remote users can be set at boot time or dynamically on a running system.

Defining the Range for the SYSGEN Parameter RJOBLIM

The allowable number of remote users of a system is determined by the SYSGEN parameter RJOBLIM. If the number of remote users exceeds the value of the parameter RJOBLIM, the image REMACP stops accepting additional incoming connections. The parameter RJOBLIM can take any value within a range defined at startup by the command file RTTLOAD.COM.

In VMS Version 5.2, the command file RTTLOAD.COM has been modified to provide an extended range for the value of the parameter RJOBLIM (the allowable number of remote users). The minimum value for RJOBLIM is 0 and the default maximum value is 255. You can raise the maximum above 255 by changing the value of RJOBLIM in SYSGEN's parameter list or by defining the logical name REM$MAX_TERMINALS. The largest possible value for REM$MAX_TERMINALS is 32767. The command procedure RTTLOAD.COM uses the larger of the values of RJOBLIM and either REM$MAX_TERMINALS (if it is defined) or the default 255 to compute the required process quotas for the REMACP process.

You can adjust the maximum number of remote users on a running system with the following procedure *after* all remote users have logged out.

1. Run the program SYS$SYSTEM:STOPREM to properly shutdown REMACP. This disables future remote logins.
2. Have all remote users log out.
3. Increase the value of RJOBLIM in SYSGEN's **active** list or define the logical name REM$MAX_TERMINALS to the newly desired maximum number of remote users.
4. Execute the command file SYS$MANAGER:RTTLOAD.COM. This provides REMACP with the correct quotas to support the specified number of remote users, and enables remote logins.

The SYSGEN Parameter RJOBLIM is Now Truly Dynamic

You can modify the value of the parameter RJOBLIM (allowable number of remote users) dynamically by changing the value in SYSGEN's **active** parameter list within the range set at boot time. Possible values for RJOBLIM now range from 0 (no remote users) to 32767.

Increasing the value of RJOBLIM allows additional remote users to log in to the system. However, if you increase the value of RJOBLIM above the range set at boot time, the image REMACP stops accepting additional incoming connections because of process resource limitations.

Decreasing the value of RJOBLIM below the current number of remote users does not log out any of the remote users. However, once a user logs out, the image REMACP will no longer accept connections until the number of remote users drops below the value of RJOBLIM.

3.41.5.2 Running Without RTTDRIVER

V5.2

The image REMACP now determines which driver is present in the system prior to accepting the network connection. Both drivers, CTDRIVER and RTTDRIVER, are now dynamically connected to the RTA device at the time the device is created. This allows you to load only the required remote terminal drivers, which is a benefit for small memory systems.

The logical names REM$NO_CTDRIVER and REM$NO_RTTDRIVER allow you to specify which drivers should not be loaded into the system. Defining either of these logical names (with any value) causes the appropriate driver *not* to be loaded during the startup of the REMACP process. If both logical names are defined, neither driver is loaded and the REMACP process is not started. The default if the logical names are not defined is still the previous behavior of loading both drivers.

If for some reason the other driver is needed after the system is already running, you must deassign the corresponding logical name and then execute the RTTLOAD command file.

Additionally, the image REMACP has been modified to ensure that the driver support is present prior to accepting the network connection. This allows RTPAD (SET HOST) to failover dynamically to the older protocol if CTDRIVER is not loaded.

3.42 Starting the Queue Manager in SYSTARTUP_V5.COM

V5.0-1

The following command is used as the example in SYS$MANAGER:SYSTARTUP_V5.COM for starting the Batch/Print Queue Manager:

```
$
$
```

The local buffer count value (specified with /BUFFER_COUNT=10), the initial allocation, and the subsequent file extension value (specified with /EXTEND_QUANTITY=25) are appropriate for a small system with one batch and one print queue. These values should be increased for standalone time-sharing and clustered machines that support many queues.

The default values for /BUFFER_COUNT and EXTEND_QUANTITY are 50 and 100, respectively. These values are generally adequate to support 5 to 20 queues where the total number of concurrent jobs is typically less than 50. To efficiently support more queues and jobs, Digital recommends specifying larger values for these qualifiers when starting the queue manager. Note that the value for the /EXTEND_QUANTITY qualifier should be the same for all nodes in a cluster.

Increasing the local buffer count decreases the number of direct I/O reads on the queue file required to perform Batch/Print operations at the expense of job controller working set size and locking activity. When memory is available, a large number of local buffers increases system performance. However, if a small amount of memory is available, using 100 or more local buffers can decrease performance by causing excessive page faulting of the job controller process.

The value for extend quantity should be at least 20 percent of the size of the queue file when the queue file is in a steady state. If the value for the extend quantity is too small, fragmentation of the queue file can occur as a result of the many file extend operations being performed on the disk.

3.43 STARTNET.COM — Problem Corrected

V5.1

VMS Versions 5.0, 5.0-1, and 5.0-2 did not assign logical names necessary for downline loading if the VAX-11 PSI product was installed.

This problem has been corrected in VMS Version 5.1. A new STARTNET.COM file is provided. If you have made site-specific changes to the STARTNET.COM file, you need to apply them to the new version. Your old STARTNET.COM file is copied to STARTNET.PRE_V51_COPY to avoid losing your existing edits. You should delete this file when you have edited the new STARTNET.COM file.

3.44 SYSGEN Parameters

The following sections describe new or changed SYSGEN parameters.

3.44.1 MULTIPROCESSING Default Value

V5.0

The default value of the MULTIPROCESSING system parameter in VMS Version 5.0 is 3. As a result, the default behavior of the VMS operating system is to load the streamlined system synchronization image and set the multiprocessing-enabled bit only if the hardware configuration is capable of multiprocessing and two or more processors are available. Otherwise, the operating system loads the uniprocessing synchronization image. (You can find additional discussion of this topic in the *VMS Device Support Manual*.)

Note that the *VMS System Generation Utility Manual* and *VMS Device Support Manual* erroneously report the default value as 1.

3.44.2 RECNXINTERVAL

V5.0

The SYSGEN parameter RECNXINTERVAL continues to specify the minimum amount of time that the connection manager will attempt to restore a failed connection to another node of a VAXcluster. However, a change was made so that the value specified is maximized against the time that it takes for a remote node to discover that the connection is broken.

The change means that the effective value used to timeout a failed connection is the greater of RECNXINTERVAL on the local node or a value supplied by the remote node that is dependent on the type of hardware port and the value of certain SYSGEN parameters. This change was made to ensure that a node will not be removed from a VAXcluster before it is able to discover that a communication problem exists.

It is possible to have a different value of effective RECNXINTERVAL for different connections. The Show Cluster Utility displays the effective value in the RECNXINTERVAL field.

The current values of RECNXINTERVAL are as follows:

RECNXINTERVAL	Minimum Value Based on Remote Port Type	Effective Value
20 seconds Ethernet port[1]	16 seconds	20 seconds
20 seconds CI port	30 seconds	30 seconds[2]

[1] Default value of the parameter.

[2] Computed from SYSGEN parameters as 3 * max(2 * PAPOLLINTERVAL, PASTIMOUT. This results in a value of 30 seconds using default parameter values.)

3.44.3 TAPE_ALLOCLS

V5.2

Beginning with VMS Version 5.2, SYSGEN contains the new parameter, TAPE_ALLOCLS.

TAPE_ALLOCLS determines the tape allocation class for the system. The tape allocation class is used to create a unique cluster wide device name for multiple access paths to the same tape.

The TAPE_ALLOCLS parameter can also be used to generate a unique cluster-wide name for tape devices with identical unit numbers.

3.45 SYSMAN Utility — Notes

The following notes pertain to the SYSMAN Utility.

3.45.1 PARAMETER SET/STARTUP Command Does Not Work

V5.2

The SYSMAN command PARAMETER SET/STARTUP does not work correctly. Use the System Generation Utility (SYSGEN) to set the name of your site-independent startup command procedure.

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

3.45.2 SET PROFILE Command — Problem

V5.0

The SYSMAN Utility SET PROFILE command allows you to set a default directory and privileges when executing SYSMAN commands on a remote system.

If a remote operation is aborted by occasionally pressing CTRL/C, the profile you set using the SET PROFILE command may be reset to the default specified in the user authorization file (UAF) for that remote node. After pressing CTRL/C, you should check your default directory and privileges and then (if necessary) reenter the SET PROFILE subcommand before you enter any additional SYSMAN commands.

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

3.46 System Dump Analyzer — Requirements for VIRTUALPAGECNT System Parameter

V5.0-2

The Version 5.0 *VMS System Dump Analyzer Utility Manual* recommends that you set the system parameter VIRTUALPAGECNT to the size of the system dump file plus 3000 in order to do the following:

- Maintain sufficient virtual address space for the System Dump Analyzer to map a dump
- Load any required symbol tables
- Store stack information

Digital now recommends that you set VIRTUALPAGECNT to at least the size of the system dump file plus 4700. If your SDA sessions require many symbols (and invoke the READ/EXECUTIVE command), you should set VIRTUALPAGECNT to the size of the dump file plus 5750.

3.47 TDRIVER.MAR — Corrections

V5.2

A new version of the file TDRIVER.MAR is supplied in SYS$EXAMPLES:. This version follows the recommended customer conventions of using underscores (_) in symbol names instead of dollar signs ($).

The unit initialization routine now returns a status in R0 that is required for BI device drivers.

The numeric constant offset for VEC$L_ISR was replaced with the correct symbolic offset. In addition, several comments have been corrected.

3.48 TU81–Plus Tape Drives — Data Loss

V5.0-1

Version 5.0 of the VMS operating system contained a problem with RMS that caused data to be lost without reporting an error message on some systems. The data loss occurred when an RMS $GET operation was issued to a TU81–Plus tape drive on a VAX 82xx or VAX 83xx computer. The problem did not affect the Backup Utility, but it did affect other utilities such as the COPY and CONVERT utilities.

Version 5.0-1 of the VMS operating system fixes this problem.

3.49 User Environment Test Package (UETP) Notes

The following notes pertain to the User Environment Test Package (UETP).

3.49.1 DECnet Phase Notes

The following changes have been made to the DECnet phase of the User Environment Test Package (UETP).

3.49.1.1 Change in Defaults for DECnet Installation

V5.2

To increase security, the default setup for a DECnet installation has changed. When NETCONFIG.COM is run and the defaults are selected, a default DECnet account is no longer created, and default access is no longer provided for the FAL network object and the NML network object. (See also Section 3.40.2.)

In VMS Version 5.2, the UETP DECnet phase has been changed so as not to depend on default access control for the NML object; however, UETP still assumes the existence of default access control for the FAL object (this assumption will be removed in a future release of VMS). When you install DECnet according to the defaults presented by the procedure NETCONFIG.COM, the UETP DECnet phase may produce error messages that were not seen in earlier versions.

If default FAL access is disabled at the remote node selected by UETP for DECnet testing (the adjacent node on each active circuit, or a node defined by the group logical name UETP$NODE_ADDRESS), messages similar to the following will appear:

```
%UETP-W-TEXT, The process -SVA019841_0001- returned a final status of:
%COPY-E-OPENOUT, error opening !AS as output
```

These messages are followed by:

```
%COPY-E-OPENOUT, error opening 9999"":::SVA019841.D1; as output
-RMS-E-CRE, ACP file create failed
-SYSTEM-F-INVLOGIN, login information invalid at remote node
%COPY-W-NOTCOPIED, SYS$COMMON:[SYSTEST]UETP.COM;2 not copied
%UETP-E-TEXT, Remote file test data error
```

3.49.1.2 UETP$NODE_ADDRESS Problem Corrected

V5.2

In releases of UETP since VMS Version 5.0, the user has been able optionally to define the group logical name UETP$NODE_ADDRESS, which specifies a remote node to be used in the DECnet phase of UETP testing. Prior to VMS Version 5.2, when UETP$NODE_ADDRESS was defined and the node it pointed to was unable to participate in DECnet testing, the DCL command procedure UETDNET00.COM sometimes produced DCL errors and failed to complete. This problem has been fixed in VMS Version 5.2.

3.49.1.3 Effect of Defining UETP$NODE_ADDRESS

V5.2

The UETP DECnet phase does three stages of testing:

1. The local node
2. Each adjacent node that it can reach
3. The state of each active, testable circuit

The last stage is performed by the image UETNETS00.EXE.

You can optionally define the group logical name UETP$NODE_ADDRESS, which specifies a remote node to be used in the test. A side effect of using UETP$NODE_ADDRESS is that only one circuit (the first ACTIVE circuit that NCP finds) is written into UETININET.DAT, a file that normally contains all active, testable circuits. Because UETNETS00.EXE uses UETININET.DAT, it tests only that circuit.

3.49.1.4 Support for RV60 Optical Disk Drive

V5.2

For VMS Version 5.2, the tape test portion of UETP has been enhanced to support the RV60 optical disk drives that come with the RV64 optical-disk storage system.

The RV64 is an optical-disk jukebox which stores up to 64 optical disks and uses a robot arm to load them into its optical disk drive, the RV60. There are up to four RV60 drives in the RV64. The Jukebox Control Software (JCS) is a layered product on the VMS operating system that comes with the RV64 and is responsible for controlling the robot arm. To run UETP on the RV64, use JCS to load an optical disk in each of the RV60 drives. Initialize these disks with the label **UETP**, but do not mount them. UETP will test all the RV60s present in the RV64 simultaneously. Unlike the other tape tests, UETP does not re-initialize the optical disks at the end of the test.

3.49.2 UETP Modifications

V5.1

The following UETP modifications were made for VMS Version 5.1.

- VMS Version 5.1 updates the UETSUPDEV.DAT file in UETP to support the DELQA Ethernet controller.
- VMS Version 5.1 updates the UETTAPE00 test in UETP to support the RV20 Optical Disc Drive.
- In versions prior to Version 5.1, UETP disabled, by default, the DECnet and the LAT software during its DEVICE phase in order to test communication devices. However, due to increasing VMS dependence on DECnet and the LAT, UETP no longer disables DECnet or the LAT.

  If DECnet or the LAT software is running during the DEVICE phase, the UETUNAS00 test displays the message:

```
-UETP-W-TEXT, Device is in use by DECnet or another application.
```

  Other UETP communication device tests display the message:

```
"-SYSTEM-W-DEVALLOC, device already allocated to another user"
```

  This message will be updated to match the message printed by the UETUNAS00 test in a future release of VMS.

- If you are running UETP from a DECterm window on a VAXstation that is running DECwindows, the UETTTYS00 test might give the following messages:

```
%UETP-W-TEXT, The process -UETTYS00_000n- returned a final status of:
%SYSTEM-F-BADPARAM, bad parameter value
```

  and

```
-UETP-E_DEUNUS, UETTTYS00 device TWAn: is unusable, error code=00018272
-RMS-E-DNR, device not ready, not mounted, or unavailable
```

  This is a known problem and will be fixed in a future release of VMS.

3.49.3 UETP Support Added For TA90 Tape Drives

V5.1-1

In VMS Version 5.1-1, UETP now supports the TA90 tape drive.

3.49.4 SYSUAF Quotas for SYSTEST and SYSTEST_CLIG Accounts

V5.0

For VMS Version 5.0, the following SYSUAF quotas must be used for the SYSTEST and SYSTEST_CLIG accounts:

/ASTLM=100
/BIOLM=18
/CPU=no limit
/DIOLM=55
/BYTLM=32768
/ENQLM=300
/FILLM=100
/PGFLQUOTA=20480
/PRCLM=8
/TQELM=20
/WSDEFAULT=256
/WSQUOTA=512
/WSEXTENT=2048

3.50 UNIBUS Devices — Notes

The following notes pertain to UNIBUS devices.

3.50.1 Floating Interrupt Vector Change

V5.0

In VMS Version 5.0, the algorithm that is used to allocate interrupt vectors for UNIBUS peripherals has changed.

Because of this change, when systems are autoconfigured during booting, it is possible that some systems may require UNIBUS peripherals to have their interrupt vectors modified by a Digital Field Service representative. Note that this change does not affect most VMS systems.

The kinds of systems affected include any system with two or more UDA, KDA, RQDX or BDA controllers on the same bus with any other device that has a hard-wired floating interrupt vector alignment of four, such as the following:

- RX211
- DR11W OR DRV11W
- DMZ32
- LNV21
- VS100
- VSV21
- IBQ01

New Behavior Versus Old Behavior

SYSGEN has been modified to support the old and new vector allocation algorithms. The new algorithm is used as the default behavior. The old algorithm is available for the next two releases of the VMS operating system. Selection of the autoconfiguration algorithm is controlled by the SYSGEN parameter VMS8. This parameter can be set to give either the old or new algorithm behavior according to the following chart.

VMS8 Value	Description
VMS8 = 0	Gives new behavior algorithm with error messages
VMS8 = 1	Gives old behavior algorithm with error messages
VMS8 = 2	Gives new behavior algorithm without error messages

In Case of Problems

While using the new behavior algorithm, if SYSGEN detects a difference between the old algorithm and the new one, it signals the following error message:

```
SYSGEN-W-MISCONFUNI, UNIBUS device DMF32 has been misconfigured,
interrupt vector should be 000324
```

This error is written to the terminal that is running SYSGEN, OPCOM, and the error logger. If this error message is received, you should call your Digital Field Service Representative or proceed as follows:

1. Reboot the system, using a conversational boot (R5 = 1).
2. Set the SYSGEN parameter VMS8 = 1 (see the preceding chart of VMS8 parameter values). This allows the system to boot successfully.
3. Continue to boot the system.
4. At this point, the following message is displayed:

   ```
   %SYSGEN-W-NONSTDUNI, UNIBUS in nonstandard configuration,
   device DMF32 vector is wrong
   ```

5. When the system has booted, execute the following command to determine the present interrupt vector settings:

   ```
   $
   ```

6. Next, use the CONFIGURE command to determine the correct vectors for all devices. Enter the following commands:

   ```
   $
   SYSGEN>
   DEVICE>
   DEVICE>
   ```

7. Compare the present (incorrect) interrupt vector settings with the just determined new (correct) interrupt vector settings.
8. The boards with the incorrect interrupt vectors must now be rejumpered with the new (correct) interrupt vector settings.

9 Once all of the incorrect vectors have been rejumpered, the system should be rebooted with VMS8 set to 2. Setting VMS8 = 2 disables error messages for bad vector problems and uses the new vector allocation algorithm.

CAUTION: Setting VMS8=2 without changing the proper interrupt vectors may result in system failures and crashes.

Note: As long as VMS8 is set to 1, and a device found on the UNIBUS is misconfigured, the following message is displayed:

```
%SYSGEN-W-NONSTDUNI, UNIBUS in nonstandard configuration,
device DMF32 vector is wrong
```

This message means that a device has been found on the UNIBUS whose interrupt vector is incorrect and should be changed. This message will continue to be displayed until the situation is corrected.

3.50.2 VAX 8800 Systems Running SMP — Known Problem

V5.0

There is a known problem with UNIBUS operations on VAX 8800 processors when running symmetrical multiprocessing (SMP). The VAX 8800 NBIA (memory interconnect to VAXBI adapter) and UBA (UNIBUS adapter) can deadlock while waiting for each other to complete certain operations. Because both adapters can process exactly one transaction at a time and because they can also request the assistance of the other in order to complete a transaction, the deadlock situation is quite probable.

In order to avoid this deadlock situation, the VMS operating system forces PRIMARY affinity for all UNIBUS controllers configured in a VAX 8800 system. The enforcement of PRIMARY affinity prevents the deadlock situation from occurring. The requirement to limit access for UNIBUS devices to the PRIMARY processor is a VAX 8800 restriction and does not apply to other SMP systems.

If you have written a UNIBUS device driver that has been converted to run on SMP configurations in VMS Version 5.0, you may want to allow that driver to execute on both CPUs in a VAX 8800 configuration. In order to allow this, the driver must first guarantee that it does not perform any READ-MODIFY-WRITE operations to I/O address space. For example, it cannot perform a BISW #x,(R2), where R2 is pointing to a UNIBUS Control and Status Register (CSR). If a device driver has been verified to behave correctly, then it can circumvent the restriction that forces all I/O operations to execute on the PRIMARY processor.

In order for a device driver to circumvent the PRIMARY affinity policy, it must set the UCB$L_AFFINITY field of the unit control block (UCB) to -1 in the device driver's UNIT or CONTROLLER INITIALIZATION routine.

3.51 VAXcluster Notes

The following sections contain release notes concerning various aspects of VAXcluster operation.

3.51.1 Cluster Disk Reappearance Verified After Major Node Reboot

V5.1

In some Local Area VAXcluster configurations, remote disks do not reappear after a major node reboot. Disks on satellite nodes that are being mount verified return an erroneous "offline" status in response to a "get unit status" command from the rebooting disk. As a result, the rebooting boot node does not create the proper data structure for the disk.

Version 5.1 of the VMS operating system corrects this problem.

3.51.2 Color VAXstations in Clustered Environments

V5.0

The hardware initialization of the VAXstation II/GPX hardware takes a significant period of time. Normally this is not an issue, since the initialization occurs very early in the boot process when the system is preparing the console window.

It is possible to operate the VAXstation II/GPX workstation with a separate console terminal attached to the console port. However, by doing this, the VAXstations II/GPX initialization occurs during startup processing, which is after the system joins a VAXcluster. This initialization may cause the connections to the cluster to be lost and result in a fatal CLUEXIT error shortly after joining the cluster.

To avoid this situation, the SYSGEN parameter RECNXINTERVAL should be increased to at least 40 seconds on *all* nodes in a cluster that contain VAXstation/GPX workstations with separate console terminals.

3.51.3 CPUs Supported in a VAXcluster

V5.2

With VMS Version 5.2, the number of CPUs supported in a VAXcluster is increased from 42 to 96. See Section 3.6.1 for information on properly configuring larger clusters.

3.51.4 Dynamic Selection of Resource Managers

V5.2

Beginning with VMS Version 5.2, the manager of a resource that is locked by processes residing on multiple nodes may be automatically changed to distribute load and improve performance in a VAXcluster system. In the past, the manager of a resource was only changed during a lock rebuild or when all locks were released on the resource.

The net effect of the adjustment procedure is to distribute management of resources among nodes having a non-zero value of the SYSGEN parameter LOCKDIRWT. This procedure tends to move management of resources away from nodes with a zero value of LOCKDIRWT.

Note: The distribution algorithm is likely to change in a future release of VMS.

The distribution procedure operates as follows:

- If multiple nodes with a non-zero value for LOCKDIRWT have locks on a resource, the directory node is favored as the resource manager, if it holds locks on the resource.
- If a node with a zero value for LOCKDIRWT and a node with a non-zero value have locks on the resource, then the non-zero node is favored as the resource manager.
- Resources locked by processes on only one node continue to be managed by that node.

In all other cases, no attempt is made to change the manager of the resource.

This function is disabled in a cluster containing nodes running earlier versions of VMS.

3.51.5 VAXcluster Ethernet Adapter Restriction

V5.0

A node within a Local Area or mixed-interconnect VAXcluster cannot have more that one Ethernet adapter physically present. If you have more than one Ethernet adapter on a processor, the cluster software may not select the adapter that you wish it to use. The resulting failure modes are configuration dependent and can include problems such as failure to join the cluster, or, if the extra Ethernet adapters are on a boot node, the satellites may be unable to boot.

This restriction will be removed in a future release.

3.51.6 Reduction in VAXcluster State Transition Time

V5.2

In VMS Version 5.0, the lock rebuild operation was changed to eliminate or greatly reduce the perceived effect of adding or removing nodes in mixed-interconnect and Local Area VAXcluster (LAVc) configurations. VMS Version 5.2 extends these changes to all VAXcluster configurations.

In general, the time required to rebuild the lock database depends on whether a node is added or removed from the cluster. The effect of adding a node is normally minimal. The effect of removing a node varies with the number of locks managed by the node, but in most cases is minimal.

The *VMS Version 5.0 Release Notes* included the following description:

> When a node on which the SYSGEN parameter LOCKDIRWT is set to zero joins a VAXcluster environment and there are at least two nodes already present with a non-zero value for LOCKDIRWT, no lock rebuild is performed. This is typically the case when a satellite boots into a Local Area or Mixed-Interconnect VAXcluster configuration that includes at least two boot servers or disk servers.
>
> When one or more nodes on which the SYSGEN parameter LOCKDIRWT is set to zero are removed from a VAXcluster configuration and at least two nodes remain with a non-zero value for LOCKDIRWT, a partial rebuild is performed. A partial rebuild

typically takes a small number of seconds to release locks held by the removed nodes and to ensure that a new resource manager is selected for resources that were managed by the removed nodes. This type of rebuild is typically performed in a Local Area or Mixed-Interconnect VAXcluster configuration having at least two boot nodes when a satellite node shuts down.

This modified lock rebuild is not used during a rolling upgrade.

That behavior is still used when a cluster contains some nodes running VMS Version 5.2 and other nodes running an earlier version of VMS. However, the entire lock database must be rebuilt when the first old-protocol node joins a cluster containing VMS Version 5.2 nodes.

3.51.7 Show Cluster Utility — Notes

The following sections describe problems and restrictions with the Show Cluster Utility.

3.51.7.1 HW_TYPE Field Changes

V5.0

Prior to VMS Version 5.0, the format of the HW_TYPE field in the SYSTEMS class was a 4-character string representing the hardware type of the remote system. In VMS Version 5.0, the format of the HW_TYPE field is now more descriptive. For example, the type "V780" is now displayed as "VAX–11/780".

The valid hardware types that can be specified when using the ADD or REMOVE /TYPE commands are defined as those hardware types that may appear in the HW_TYPE field. Because the format of this field has changed, the hardware types that can be specified in these commands have also changed. Note, however, that because a system running a version of the VMS operating system prior to Version 5.0 may coexist with systems running VMS Version 5.0 (though not necessarily be a cluster member), it is still possible for systems using the 4-character format to appear in the display. This is also true of HSC50 and HSC70 nodes that still use the 4-character method. For this reason, the 4-character hardware types are still valid.

This means that if you previously used the following command to remove VAX–11/780 systems from the display, you should understand that this command only removes systems for which a hardware type of V780 is displayed and not Version 5.0 systems that display the new VAX–11/780 hardware type.

```
Command >
```

Digital recommends that you use the following command to ensure that VAX–11/780 systems using both the new Version 5.0 format and the old format are removed from the display.

```
Command >
```

Specifying both formats should continue until versions of the VMS operating system prior to Version 5.0 are no longer in use, at which time "V780" may be dropped from the command.

Also, because of the special characters in the new format, you must enclose the string in quotation marks as in the previous example. Quotes are optional when using the 4-character format.

3.51.7.2 WRITE/ALL Method for Determining Page Size

V5.0

Prior to VMS Version 5.0, the output page size when using the WRITE/ALL command, is assumed to be 132 columns and 66 lines per page. All 66 lines are assumed to be usable lines. Because the output page may not always be 66 lines and since it is often desirable to allow for margins at the top and bottom of the page, the VMS Version 5.0 Show Cluster Utility no longer makes this assumption. Instead, SHOW CLUSTER uses the Run-Time Library routine LIB$LP_LINES and the logical name SYS$LP_LINES to determine the actual number of usable lines per page.

Note that this is regarded as a temporary solution. In a future version of the VMS operating system, Digital expects to modify SHOW CLUSTER to provide a means of specifying the actual number of columns and lines per output page.

For additional information regarding LIB$LP_LINES, see the *VMS RTL Library (LIB$) Manual*.

3.51.8 Shutdown Notification on Clusters

V5.0

Whenever you execute the orderly shutdown procedure (SYS$SYSTEM:SHUTDOWN.COM) on one VAXcluster member system, users on all member systems are notified. Clusterwide notification is required, because users logged in to any member system may be affected by the shutdown of another system in various ways:

- Users may have batch jobs running on other systems.
- If terminal servers are in operation, users may have alternate terminal sessions in progress (for example an editing session) on the system being shut down.

Because shutdown messages include the name of the member system being shut down, users need only check the messages carefully to avoid logging out of a system unnecessarily.

Note that, for those reasons, clusterwide notification is not affected by the shutdown procedure's REPLY /NODE= option. If, for some reason, you want to limit shutdown notification to specific member systems, define the logical name SHUTDOWN$INFORM_NODES before executing the shutdown procedure. For example:

```
$
$
```

In this example, only users on systems MOE and LARRY will be notified.

3.51.9 VAXBI 5 — Restriction

V5.0

The VAX 8820, 8830, and 8840 support up to six VAXBIs (VAXBI 0 through VAXBI 5). Never configure the last VAXBI, VAXBI 5, to have a node 15. The system cannot recognize devices connected to node 15 on VAXBI 5.

3.52 VMS Executive — Changes

The following sections describe changes in the VMS executive.

3.52.1 Changes to Process Paging File Assignment

V5.0

In Version 4.n, a process was assigned to a paging file at process creation time. The page file chosen was the one with the most free, or unallocated, blocks.

In Version 5.0, a process may use up to four page files simultaneously. The set of page files used changes dynamically as the process executes.

One consequence of this change is that paging file global sections are no longer forced to have their backing store in the primary pagefile, SYS$SPECIFIC:[SYSEXE]PAGEFILE.SYS. Frequent users of pagefile global sections are RMS global buffers.

3.52.2 Extended Working Set Sizes

V5.0

Prior to VMS Version 5.0, the maximum working set extent allowed was 64K pages. For Version 5.0, working set extent can be as large as 100K pages. However, working set quota is still limited to 64K pages.

3.52.3 Paging File Recommendation

V5.0

Digital recommends that the Version 4.n process of creating a minimal, primary paging file on the system disk and significantly larger paging files on alternate disks be reexamined. In particular, the best paging file load balancing tends to occur when all paging files are created approximately the same size.

3.52.4 PAGEFILE.SYS — Changes

V5.0

In Version 4.n, when the system was booted, SYSINIT looked for the primary page file, SYS$SYSROOT:[SYSEXE]PAGEFILE.SYS. If PAGEFILE.SYS was not found an error message was displayed and the boot operation continued even though the VMS operating system did not support a configuration without a primary paging file.

In Version 5.0, if PAGEFILE.SYS is not found, SYSINIT issues the following informational message:

```
"%SYSINIT-I- PAGEFILE.SYS not found - system initialization continuing..."
```

Then, in STARTUP.COM, before any of the system overhead processes are created, (for example, OPCOM, JOBCTL) the startup procedure searches for SYS$MANAGER:SYPAGSWPFILES.COM. If the site-specific file is found, it is invoked.

In addition, an abbreviated version of SYPAGSWPFILES.COM is included in Version 5.0.

You may include any necessary commands in SYPAGSWPFILES.COM to accomplish the installation of paging and swap files (for example, volume initialization, mounting disks, SYSGEN INSTALL commands). The CONFIGURE process is running at the time the site-specific COM file is invoked so that in the absence of any controller or device errors, HSC-based disks will eventually be recognized by the system.

When control returns to STARTUP, at least one paging file must be successfully installed. If one is not installed, STARTUP reports the following error:

```
%STARTUP-E-NOPAGFIL, no page files have been successfully installed."
```

In effect, these changes make the notion of a primary paging and swap file almost obsolete.

3.53 VMS Kits Provided on RX33 Diskettes

V5.2

If you receive your VMS kit on RX33 diskettes, the DECwindows kit is provided on a TK50. Please be aware that the upgrade procedure will delete any DECwindows files currently on your system during Phase 1. This is done prior to asking what VMS and DECwindows options you may wish to select. If you are not able to use the TK50 DECwindows kit during the upgrade (no drive available), please be sure *not* to select any of the DECwindows options. If you need to have DECwindows installed on the system, please install DECwindows using the same kit and method previously used.

3.54 VMSINSTAL — CHECK_VMS_VERSION Callback Enhancement

V5.0-1

The CHECK_VMS_VERSION callback in VMSINSTAL has been enhanced for Version 5.0-1 of the VMS operating system. This callback still functions as described in the *VMS Developer's Guide to VMSINSTAL*, but now allows you to specify a maintenance release for the minimum_version and maximum_version parameters.

Although the format for specifying versions to this callback has changed, products that use this callback are not affected because the old format is still supported. However, Digital recommends that you convert products that use the old format to the new format with the next release of the product.

Also note that, if you need to specify a particular maintenance release when passing the minimum and maximum versions for a product, you must use the new format. The new format for expressing VMS versions is as follows:

vv.u-mh

where:

- *vv* indicates a version
- *u* indicates an update
- *m* indicates a maintenance level
- *h* indicates a limited hardware release (LHR)

For example, 5.0-1 indicates that the version of the product is 5, the update is 0, and the maintenance level is 1.

The CHECK_VMS_VERSION callback has the following format:

```
CHECK_VMS_VERSION symbol minimum_version [option] [maximum_version]
```

The parameters on the command line indicate the following:

- *symbol* is the name of a global symbol that will be defined with a TRUE/FALSE Boolean value that indicates the results of the version check.
- *minimum_version* is the minimum VMS version required to install the product. This parameter is passed in the form "vv.u[-mh]" (for example 5.0 or 5.0-1). To pass a maintenance release, you must use this format.

  The format "vvu" is also supported (for example 050) and is the minimum value that you can provide for this parameter.
- *option* is used to limit the product installation to a field test version of the VMS operating system. If you specify F on the command line, the product is restricted to the specific field test version of the VMS operating system that is specified by the minimum_version parameter.
- *maximum_version* is the maximum VMS version required to install the product. Use this parameter if a product will not function above a certain version. This parameter uses the same format as the minimum_version parameter.

For example, to restrict a product installation to VMS Versions 4.6 to 5.0, you could use the following command line:

```
$
```

However, if you want to restrict the product installation to VMS Versions 4.6 to 5.0-1, you need to use the new format:

```
$
```

3.55 VWS Workstations — Setting of Multiprocessing SYSGEN Parameter

V5.0

If you have a workstation that is running VWS, do not set the SYSGEN parameter MULTIPROCESSING to the value 2. Multiprocessing is not supported for VAX workstations.

4 Programmer Release Notes

This chapter includes information that is of interest to both the application and system programmer.

4.1 Activating an Image With System Version Mismatch — Change

V5.2

Prior to VMS Version 5.2, running an image linked with a system symbol table (SYS$SYSTEM:SYS.STB) other than that of the running system resulted in a successful image activation with CMKRNL and CMEXEC privileges removed.

Starting with VMS Version 5.2, running such an image fails and displays the following error message:

```
SS$_SYSVERDIF, system version mismatch, please relink
```

You should inspect user programs that activate other images linked against the system symbol table (for example, programs that call LIB$FIND_IMAGE_SYMBOL) to determine whether they depend on the obsolete behavior of VMS versions prior to Version 5.2. If so, remove the dependency on the obsolete behavior from any such user program; then relink, under VMS Version 5.2, both the calling program and the image that failed to activate.

4.2 VAX Ada — Run-Time Library Notes

The following sections contain information about the VAX Ada Run-Time Library.

4.2.1 Restriction on END_OF_FILE Function

V5.0

The END_OF_FILE function of packages SEQUENTIAL_IO and SEQUENTIAL_MIXED_IO will raise USE_ERROR when called for a file that is opened on a remote DECnet node, due to an RMS restriction. Other packages are not affected. Until the restriction is removed, the error can be avoided by opening the file using a FORM string argument to the OPEN or CREATE procedures of the following:

```
"FILE;SEQUENTIAL_ONLY NO;"
```

Note: Disabling the "sequential only" mode incurs a performance penalty on all network file access.

4.2.2 Routines to Improve AST Handling and Time Slicing

V5.2

Two routines have been added to the Ada run-time library to improve AST handling and time slicing:

- EXPAND_AST_PACKET_POOL (Section 4.2.2.2)
- REQUEST_TIME_SLICE (Section 4.2.2.3)

A new Ada package that you can use to call these routines from your Ada program is described in Section 4.2.2.1.

4.2.2.1 Ada SYSTEM_RUNTIME_TUNING Package

V5.2

You can use the following Ada package to call the new routines described here from your Ada program. This package is supplied in the VAX Ada Version 2.0 predefined program library.

```
package SYSTEM_RUNTIME_TUNING is

  subtype AST_PACKET_REQUEST_TYPE is NATURAL range 0 .. 1_048_576;

  procedure EXPAND_AST_PACKET_POOL (
      REQUESTED_PACKETS       : in AST_PACKET_REQUEST_TYPE;
      ACTUAL_NUMBER           : out NATURAL;
      TOTAL_NUMBER            : out NATURAL);

  pragma INTERFACE(RTL, EXPAND_AST_PACKET_POOL);
  pragma IMPORT_PROCEDURE(EXPAND_AST_PACKET_POOL,
      "ADA$EXPAND_AST_PACKET_POOL");


  procedure REQUEST_TIME_SLICE (REQUESTED_VALUE :     DURATION);

  pragma INTERFACE(RTL, REQUEST_TIME_SLICE);
  pragma IMPORT_PROCEDURE(REQUEST_TIME_SLICE,
      "ADA$SET_TIME_SLICE");

end SYSTEM_RUNTIME_TUNING;
```

4.2.2.2 EXPAND_AST_PACKET_POOL Routine

V5.2

This routine adds more AST packets to the pool of packets used by the AST_ENTRY attribute. It supports the creation of up to 1048576 packets. The call succeeds only if there is enough virtual memory to satisfy the request. (A single AST packet currently consumes 32 bytes of dynamic memory.)

When you use the AST_ENTRY attribute to handle an AST, an AST packet is used by the Ada run-time library to hold the AST parameter. An AST packet is in use from the time the AST is delivered by VMS until the receiving task completes the accept statement receiving the AST parameter. If the peak number of ASTs delivered by VMS, but not yet accepted, exceeds the size of the AST packet pool, an unrecoverable error occurs. The error message states that the AST packet pool has been exhausted. The EXPAND_AST_PACKET_POOL routine can help eliminate that error by increasing the size of the AST packet pool.

Before you increase the AST packet pool, try to minimize the peak number of AST packets required by your program. To do this, if possible ensure that the accepting task has a very high priority, and is not delayed by an interaction with any other task before or during the accept statement

for the AST. Only after you have concluded that the AST arrival rate is so high that it momentarily exceeds your program's rate of servicing the ASTs, should you consider using this routine to increase the size of the pool.

Note: Using this routine will not help if your program's average AST arrival rate is greater than its average AST service rate, because eventually your program will still run out of AST packets. In this case, you need to revise your program to reduce the AST arrival rate. How you do that depends on your application.

Input Parameters

The REQUESTED_PACKETS parameter is the minimum number of additional packets desired. More may be allocated because of rounding to the next storage boundary. To determine the current size of the pool, you can specify 0 for the value of the REQUESTED_PACKETS parameter.

Output Parameters

The ACTUAL_NUMBER parameter indicates the number of packets that were added to the pool.

The TOTAL_NUMBER parameter indicates the total number of AST packets in the pool. (Note that this number includes AST packets currently in use for the delivery of an AST.)

Exceptions

The STORAGE_ERROR exception is raised if the request could not be satisfied because of insufficient memory. When the STORAGE_ERROR exception is raised, an attempt is made to release any AST packets allocated in partial fulfillment of the request.

The PROGRAM_ERROR exception may be raised for certain other errors. If the PROGRAM_ERROR exception is raised, a chained condition indicates a detailed reason for the failure.

Other exceptions may be raised as well.

4.2.2.3 REQUEST_TIME_SLICE Routine

V5.2

This routine conditionally modifies the time-slice setting of the program. This entry point can only make time slicing run faster than it is already running, or enable it, if it is not enabled. The request is always overridden by the value specified by a pragma TIME_SLICE in an Ada main program or by a debugger SET TASK /TIME_SLICE command.

This routine is primarily intended to be called from within an Ada shareable image or an object file exported by an ACS EXPORT command, where it cannot be decided in advance whether there will be an Ada main program. However, this routine can also be used to override an Ada main program that does not specify a pragma TIME_SLICE (and as often as desired).

This call has no effect if any of the following are true:

- The REQUESTED_VALUE argument is 0.0 or negative (time slicing cannot be disabled by this routine).

- The REQUESTED_VALUE argument is greater than a previously specified time-slice value that successfully set the time slice.
- Time slicing has either been activated or turned off by a pragma TIME_SLICE.
- A debugger SET TASK/TIME_SLICE=t command has been issued.

If none of the above conditions is true, then the REQUESTED_VALUE argument will set the time slice.

In the following cases, the time slice set by this call will be overridden:

- An image containing an Ada main program that has a pragma TIME_SLICE is activated.
- A debugger TASK/TIME_SLICE=t command is issued.
- The REQUEST_TIME_SLICE routine is called again with a REQUESTED_VALUE greater than zero but less than the value set by this call.

Input Parameters

The REQUESTED_VALUE parameter is the requested new time-slice value.

Exceptions

The PROGRAM_ERROR exception may be raised for certain errors. If the PROGRAM_ERROR exception is raised, a chained condition indicates a detailed reason for the failure.

Other exceptions may be raised as well.

4.3 BIIC — Examining Self-Test Status

V5.2

VMS Version 5.2 includes a new macro, BI_NODE_RESET, that must be used by all VAXBI device drivers that initiate a BIIC self-test.

The *VMS Device Support Manual* lists precautions a VAXBI device driver must take when initiating a BIIC self-test. These precautions have been revised as follows:

> Normally, only diagnostics initiate a self test by setting the SST bit in the BIIC. A VAXBI driver that sets this bit must take special precautions to avoid a machine check and to avoid undetected corruption of VAXBI memory. These precautions include the following steps:
>
> 1 Use the $PRTCTINI macro to begin a machine check protection block, supplying the location of the end of the block in the **label** argument and the mask value #<MCHK$M_NEXM!MCHK$M_LOG> in the **mask** argument. (Note that you must include an invocation of the $MCHKDEF macro in the driver to use these symbols.) Code within the block executes at IPL 31.

2 Invoke the BI_NODE_RESET macro as follows:

```
BI_NODE_RESET  CSR=R4
```

The BI_NODE_RESET macro uses the recommended instruction sequence to disable arbitration on the VAXBI node to be reset, and sets the node-reset and self-test status bits in BIIC$L_BICSR. The use of any instruction sequence, other than that defined by the BI_NODE_RESET macro, to perform these actions may cause an undefined condition on the VAXBI bus.

3 Use the $PRTCTEND macro to end the machine check protection block. You must specify in the **label** argument the same value that you specified in the **label** argument to the $PRTCTINI macro.

4 Do not access the BIIC registers for at least one millisecond. You may not even check the state of the STS bit during this interval.

5 Do not access any other address on the VAXBI node until the self test is completed.

A description of the BI_NODE_RESET macro follows.

BI_NODE_RESET

Initiates BIIC self-test on the specified VAXBI node.

FORMAT

BI_NODE_RESET *csr*

PARAMETERS

csr
General purpose register that contains the address of the VAXBI node's control and status register (CSR).

DESCRIPTION

The BI_NODE_RESET macro uses the recommended instruction sequence to disable arbitration on the specified VAXBI node, and sets the node reset and self-test status bits in the BIIC CSR. The use of any instruction sequence, other than that defined by the BI_NODE_RESET macro, to perform these actions may cause an undefined condition on the VAXBI bus.

4.4 VAX C Notes

The following sections contain information concerning VAX C.

4.4.1 Mixing D_FLOAT and G_FLOAT Modules

V5.2

If you have VAX C Version 3.0 or later and VMS Version 5.2, you can mix D_FLOAT and G_FLOAT modules within the same program. To do this, include the files STDIO.H, STDLIB.H, MATH.H, and UNIXLIB.H in all G_FLOAT modules of the program and compile those G_FLOAT modules with /DEFINE=("CC$mixed_float". Then link all modules against the files VAXCRTL.EXE or VAXCRTL.OLB.

Note: You must use the include files shipped with V3.0 of VAX C or later, or compiling with /DEFINE=CC$mixed_float will have no effect.

Modules that use only D_FLOAT variables do not have to contain the above include files. Similarly, they do not need to be compiled with the /DEFINE option.

If you are linking a program against the file VAXCRTL.OLB, including the definition files listed above in each module and compiling each module with the /DEFINE option will produce a minor gain in program efficiency and may significantly reduce the size of any executable produced.

Note: Digital strongly recommends linking against the file VAXCRTL.EXE instead of VAXCRTLG.EXE. Libraries linked against the file VAXCRTLG.EXE may not be useable by programs that use D_FLOAT variables and library routines.

4.4.2 VAX C Run-Time Library — Changes

V5.0

Beginning with VMS Version 5.0, the VAX C Run-Time Library (RTL) contains five new malloc routines (malloc_opt). The new malloc routines take advantage of the VMS RTL memory management routines LIB$GET_VM and LIB$FREE_VM. The performance and capabilities of these routines have been considerably improved. This includes using a zone algorithm that is first fit with no boundary tag. Subsequently, each allocation is zero-filled and aligned on an octaword boundary.

Previous versions of the malloc routines imitated the UNIX version of malloc for memory allocation or deallocation procedures. The new malloc routines do not imitate this behavior. This is exemplified when you try to sequence a freeing of dynamic memory and then try to access that memory.

Equivalent VMS routines have been created for each malloc routine. For example the VMS operating system routine corresponding to malloc is VAXC$MALLOC_OPT. To take advantage of this feature, you may find it helpful to include the following macro definitions at the beginning of your program.

```
#define malloc VAXC$MALLOC_OPT
#define calloc VAXC$CALLOC_OPT
#define free  VAXC$FREE_OPT
#define cfree VAXC$CFREE_OPT
#define realloc VAXC$REALLOC_OPT
```

Note that these routines are reentrant and may be used in asynchronous system trap (AST) routines.

4.4.3 VAX C Run-Time Library — Socket Routines

V5.2

The following socket routines are available in the VAX C Run-Time Library. Socket routines are used for interprocess communication. The routines listed here are a subset of the Berkeley 4.2Bsd Inter-Process Communication library. They provide facilities for creating and using AF_INET (Internet) sockets, and require that Version 1.2 of the VMS/Ultrix Connection product be installed in order to operate. See the VMS/Ultrix Connection Version 1.2 release notes for information on how to access these routines. See the *ULTRIX-32 Supplementary Documents, Volume III* for an overview of their use.

Version 3.0 of the VAX C product contains the include files in.h, inet.h, netdb.h, and socket.h which support the use of these routines. The routine names listed here are not currently defined as symbols in the VAX C Run-Time Library, but may be defined in some future release of VMS. When they becomed defined, they may cause multiple definition warnings in programs linking against the VAX C Run-Time Library which also define these symbols:

- accept
- bind
- connect
- gethostbyaddr
- gethostbyname
- gethostent
- gethostname
- getnetbyaddr
- getnetbyname
- getnetent
- getpeername
- getservbyname
- getsockname
- getsockopt
- htonl
- htons
- inet_addr
- inet_lnaof
- inet_makeaddr
- inet_netof
- inet_network
- inet_ntoa
- listen
- ntohl
- ntohs
- recv
- recvfrom
- recvmsg
- select
- send
- sendmsg
- sendto
- setsockopt
- shutdown
- socket
- socket_pair
- vaxc$get_sdc

4.5 $CANCEL is an Asynchronous Operation

V5.2

The $CANCEL system service performs an asynchronous cancel operation. This means that the application *must* wait for each I/O operation issued to the driver to complete prior to checking the status for that operation.

For a more detailed discussion see Section 3.41

4.6 Debugger Notes

The following sections contain information about the Debugger.

In VMS Version 5.2, the debugger is available through two user interfaces:

- Command, as in previous VMS versions
- DECwindows, which is new with VMS Version 5.2

Unless specified otherwise, the release notes apply to both interfaces of the debugger.

4.6.1 Incompatibilities with Previous Versions of the Debugger

V5.2

The following subjects are covered in this section:

- Changes from single-process to two-process (default) and multiprocess debugging configurations
- Use of the single-process debugging configuration (used in previous VMS versions) in certain cases
- Changes in the use of CTRL/Y and CTRL/C
- Changes to the debugger keypad key definitions

See also Section 4.6.2.1 and Section 4.6.2.2.

4.6.1.1 Changes from Single-Process to Two-Process or Multiprocess Debugging Configurations

V5.2

Before VMS Version 5.2, you could use the debugger only with one-process programs. The debugger and the program being debugged ran in the same process. This debugging configuration is referred to as the **single-process configuration** in these release notes.

Starting with VMS Version 5.2, you can use the debugger with multiprocess programs (programs that run in more than one process). To accommodate this and other new capabilities, such as the DECwindows interface, the debugger now consists of two parts that run in separate processes: a relatively small kernel debugger image (DEBUG.EXE), and a larger main debugger image (DEBUGSHR.EXE), which contains most of the debugger code.

This separation reduces potential interference between the debugger and the program being debugged. The separation also makes it possible to have two debugging configurations—default and multiprocess:

- When you use the debugger with a one-process program, the kernel debugger runs in one process along with the program and the main debugger runs in a subprocess. This two-process debugging configuration is the **default configuration**. It results when the logical name DBG$PROCESS is either undefined or has the value DEFAULT.
- When you use the debugger with a multiprocess program, a kernel debugger runs in each process of the program, and the main debugger runs in a subprocess. The **multiprocess configuration**, which results when the logical name DBG$PROCESS has the value MULTIPROCESS, enables one main debugger to communicate with several kernel debuggers in the same VMS job tree.

4.6.1.2 Use of Single-Process (Pre-Version 5.2) Debugging Configuration

V5.2

When you invoke the VMS Version 5.2 debugger, if a separate process cannot be created to run the main debugger image, the debugger issues one or more messages and automatically uses the single-process configuration (where the debugger shares a single process with the user program). For example, if quotas are not sufficient to create a subprocess for the main debugger, the messages are as follows:

```
%DEBUG-E-CANTCREATEMAIN, could not create the VAX DEBUG subprocess
-SYSTEM-F-EXQUOTA, exceeded quota
%DEBUG-I-SHRPRC, VAX DEBUG will share user process
```

In the single-process configuration, you can use the debugger to debug a program that normally runs in only one process. However, you cannot use the additional debugger features that are available with VMS Version 5.2:

- You cannot use the multiprocess debugging configuration.
- You cannot use the debugger's DECwindows interface.
- You do not have the benefit of reduced interference between the debugger and the program being debugged.

In the single-process configuration, use the sequence CTRL/Y and then the DCL command DEBUG to abort a debugger command or program execution from within a debugging session. Using CTRL/Y returns you to the DCL level. The DCL command DEBUG invokes the debugger, which then displays its prompt. You cannot use the SET ABORT_KEY command to reassign the abort function to another control-key sequence.

The single-process configuration avoids the restrictions on VMS Version 5.2 that are described in Section 4.6.2.1 and Section 4.6.2.2. If you wish to use the single-process debugging configuration because it avoids the restrictions described in Section 4.6.2.1 and Section 4.6.2.2, you can do so by making the following logical name assignment before invoking the debugger:

```
$
```

NOTE: Use the single-process configuration (established when the definition of DBG$PROCESS is NONE) only when necessary to avoid the restrictions of the default configuration (see Section 4.6.2.1 and Section 4.6.2.2). The single-process

configuration is unsupported and may not be available in future releases of the debugger. Please submit a Software Problem Report (SPR) if you encounter any problems using the default or multiprocess configurations (other than those mentioned in these release notes).

4.6.1.3 Changes in Uses of CTRL/Y and CTRL/C

V5.2

In previous versions of the debugger, you could use CTRL/Y as follows:

- From within a debugging session to interrupt program execution or to abort a debugger command
- From the DCL level to interrupt a program that was executing freely (You could then invoke the debugger by entering the DCL command DEBUG.)

You can now use CTRL/Y only from the DCL level.

The following sections explain how you can now interrupt program execution or abort a debugger command from within a debugging session.

4.6.1.4 Use of CTRL/C in the Command Interface

V5.2

When using the debugger's command interface, you should now enter CTRL/C rather than CTRL/Y to abort a debugger command or interrupt program execution from within a debugging session.

CTRL/C aborts these operations without exiting the debugging session (the debugger prompt is displayed after you enter CTRL/C).

If your program already has a CTRL/C AST routine enabled, use the new SET ABORT_KEY command to reassign the abort function to another control-key sequence. The SHOW ABORT_KEY command identifies the control-key sequence that is currently assigned the abort function.

If you enter CTRL/Y from within a debugging session, the effect (as in previous versions of the debugger) is to interrupt the debugging session and return control to the DCL level. The effect is the same as using CTRL/Y with any program or utility running on VMS.

4.6.1.5 Use of the Stop Button in the DECwindows Interface

V5.2

To abort a debugger command or interrupt program execution from within a DECwindows debugging session, click on the Stop button in the main window. The Stop button aborts these operations without exiting the debugging session.

When a debugger window has the input focus, pressing CTRL/Y or CTRL/W has no effect.

When the DECterm (terminal emulator) window from which you invoked the debugger has the input focus, pressing CTRL/Y interrupts the debugging session and returns control to the DCL level (as with previous versions of the debugger). The effect is the same as using CTRL/Y with any program or utility running on VMS.

4.6.1.6 Changes to Debugger Keypad Key Definitions

V5.2

The following previously unused keypad key combinations now enable you to display process-specific source and instruction displays: GOLD–KP9, BLUE–KP9, BLUE–KP7, BLUE–KP3, BLUE–KP1.

For symmetry, the command string that was previously assigned to the sequence BLUE-KP3 (SELECT/SOURCE %NEXT_SOURCE) has been moved to the previously unused sequence GOLD-COMMA.

Type HELP KEYPAD (at the debugger prompt) for summary information arranged in keypad layout. To obtain complete information about the command bindings, use the SHOW KEY command.

Use the key definitions that manipulate process-specific displays only with multiprocess programs.

4.6.2 Debugger Problems or Restrictions

V5.2

This section describes the following known problems or restrictions with the debugger in VMS Version 5.2:

- Use of the abort key or Stop button after a SPAWN command
- Use of debugger commands in DCL command procedures
- Debugger commands that are disabled in the DECwindows interface
- Use of the DECwindows interface

4.6.2.1 Use of the Abort Key or Stop Button after a SPAWN Command

V5.2

If you use the SPAWN command either from the DCL level or from within a debugging session, the debugger abort key or Stop button is disabled after you log out or return from the spawned subprocess.

In the debugger command interface, the abort key is CTRL/C by default. In the DECwindows interface, the abort button is the Stop button in the main window. See Section 4.6.1.3 for more information on the abort function.

The only way to re-enable the abort key or Stop button is to log out and log back in.

4.6.2.2 Use of Debugger Commands in DCL Command Procedures

V5.2

With previous versions of the debugger, you could use a DCL command procedure to invoke the debugger and issue debugger commands contained in that command procedure. For example:

```
$ ! Procedure to run PROG2 under debugger
$ ! control and issue debugger commands
$ !
$ RUN PROG2
SET BREAK %LINE 17
GO
EXIT
$ SHOW SYSTEM
$ LOGOUT
```

Starting with this version of the debugger, you can no longer put debugger commands directly into a command procedure. Instead, you must create a temporary file containing the debugger commands and assign the logical name DBG$INPUT to point to that file. For example:

```
$
$
$
$
$
$
```

Another workaround is to establish a single-process debugging configuration, as described in Section 4.6.1.

4.6.2.3 Debugger Commands Disabled in DECwindows Interface

V5.2

When using the DECwindows interface, you can enter debugger commands in the COMMAND box. Note, however, that the following commands are disabled in this mode of operation (the debugger issues a message when you try to enter a disabled command):

- CANCEL WINDOW
- EXPAND
- MOVE (the diagnostic message mentions EXPAND, not MOVE)
- SELECT/PROGRAM
- SET MARGINS
- SET MODE NOSCREEN
- SET OUTPUT [NO]SCREEN_LOG
- SET TERMINAL
- SET WINDOW
- SHOW MARGINS
- SHOW TERMINAL
- SHOW WINDOW

4.6.2.4 Use of the DECwindows Interface

V5.2

The following known problems or restrictions are specific to the DECwindows interface.

NOTE: The startup time for the DECwindows debugger is about 1 minute, depending on network and system load. Do not attempt to manipulate the user interface until the following three debugger windows are completely initialized: the main window, the source window, and the output window. Any attempt to manipulate the debugger interface before these windows are initialized may freeze your workstation.

In addition, when you first invoke the debugger's online help system, it may take up to a minute to display the first help topic. Subsequent help topics are displayed within a few seconds after you invoke them.

- The Modules dialog box does not correctly cancel all modules when used with an Ada program. (To display the Modules dialog box, choose Modules . . . from the Data menu).
- The SCROLL/BOTTOM and SCROLL/TOP commands do not work correctly.
- The SCROLL/LEFT and SCROLL/RIGHT commands do not work.
- If the Scope field of the Show Variable dialog box becomes greyed, it can never become ungreyed. (To display the Show Variable dialog box, choose Variables from the Data menu, then choose Show Variable . . . from the Variables submenu.)
- The SHOW DISPLAY command does not correctly show the values of the window parameters (height, width, x, y) in pixels.
- The EXTRACT/SCREEN_LAYOUT command can be used to save the current settings of the debug windows (by creating a debugger command procedure with the necessary information), but with the following restrictions:
  - The command does not correctly show the values of the window parameters (height, width, x, y) in pixels.
  - The command issues a SET TERMINAL command, which is disabled in the DECwindows debugger.
  - The command issues a CANCEL DISPLAY/ALL command, which causes the debugger to produce an internal error if not deleted from the command procedure.
  - The command does not issue the window information for the COMMAND box or the main window.
- The Full button in the Examine Code dialog box may be ungreyed incorrectly. (To display the Examine Code dialog box, choose Code from the Data menu, then choose Examine Code . . . from the Code submenu.)
- The state of the With Operands and Full buttons in the Examine Code dialog box does not reflect a previous SET MODE OPERANDS command.
- You cannot make a debugger window a NODYNAMIC window by means of a DISPLAY/NODYNAMIC command.
- The command SELECT/OUTPUT PROMPT does not cause debugger output to be sent to the PROMPT window.
- The Deposit Code . . . dialog box generates a syntax error when you deposit instructions while running an Ada program. (To display the Deposit Code . . . dialog box, choose Code from the Data menu, then choose Deposit Code . . . from the Code submenu.)

- The SET OUTPUT NOTERMINAL command should be disabled, but is not.
- The default window size of the predefined register window, REG, is not large enough to display three columns of register information.
- If any of the text shown in the main window is sufficiently large to run off the right edge of the window, the main window fails to expand to show all the information. The STOP button also disappears in that case. To work around this problem, expand the main window such that it is wide enough to show the STOP button.
- When the debugger issues a debugger command as you manipulate the DECwindows interface with the mouse, the command is echoed in the COMMAND box in all uppercase characters, even if the language is case sensitive.
- When using the Windows dialog box to modify a debugger window, be sure to select the name of the window from the list box of the dialog box. When displayed, the Windows dialog box might have a window name preselected in the Window field, but relying on this preselection can produce internal errors. (To display the Windows dialog box, choose Windows . . . from the Customize menu.)
- Do not give the PROMPT display the SCROLL attribute; this causes an ACCVIO error. Also, if the PROMPT display is given the OUTPUT attribute, debugger output is lost.
- You cannot select more than one entry from the list box of a dialog box.
- If you edit a command recalled in the COMMAND box, press CTRL/E (move to end of line) before pressing the RETURN key. This prevents the COMMAND box from breaking the command line into two parts, causing syntax errors.
- If you are reading mail in DECWindows Mail, displaying a debugger dialog box may fill in the text of the last read mail message into the dialog box. To work around this problem, select an object in a debug window, then cancel the selection by clicking on it again.
- If you try to use the Exit option in the Processes dialog box and the Process field is empty, the debugger signals an ACCVIO. (To display the Processes dialog box, choose Processes . . . from the Data menu.)
- You can enter only integer data in the Height, Width, X and Y text fields of the Windows dialog box. Do not use expressions (for example, %PAGE). (To display the Windows dialog box, choose Windows . . . from the Customize menu).

4.6.3 Obsolete Commands

V5.2

The following debugger commands and command qualifiers are obsolete in VMS Version 5.2, and are no longer documented:

Obsolete Command or Qualifier	Reason
ALLOCATE	The debugger now allocates and deallocates memory automatically. This command now has no effect.
CANCEL EXCEPTION BREAK	This command duplicates the effect of the newer command CANCEL BREAK/EXCEPTION, which better conforms to the general command format for canceling breakpoints.
SET DISPLAY	The DISPLAY command now enables you to create a new display, in addition to modifying an existing display.
SET EXCEPTION BREAK	This command duplicates the effect of the newer command SET BREAK/EXCEPTION, which better conforms to the general command format for setting breakpoints.
SET MODULE/ALLOCATE	The debugger now allocates and deallocates memory automatically. This qualifier now has no effect.
UNDEFINE	This command duplicates the effect of the newer command DELETE, which conforms to the analogous DCL command DELETE.
UNDEFINE/KEY	This command duplicates the effect of the newer command DELETE/KEY, which conforms to the analogous DCL command DELETE/KEY.

4.6.4 Debugging SMG Programs — Restriction

V5.0

The debugger uses the VMS Screen Management Facility (SMG) to implement screen mode. If your program also calls SMG routines and you debug it with the debugger running on the same terminal, there is likely to be interference between your program and the debugger.

To avoid this problem, debug the program using two terminals or a VAXstation. This technique is described in the *VMS Debugger Manual*.

4.6.5 Using the Debugger on a VAXstation — Problem

V5.0

There is a problem with the handling of CTRL/Y when the debugger is running in its own window—that is, if you have entered the command SET MODE SEPARATE. CTRL/Y is ignored when the keyboard is attached to the debugger window. To make CTRL/Y take effect, attach the keyboard to the window from which you invoked the debugger (by pointing at that window with the mouse); then, press CTRL/Y.

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

4.7 DECnet — File Access Protocol Extensions

V5.1

The DECnet file access protocol (DAP) support in VMS has been enhanced to properly handle compound document (for example, DDIF) files. This support is fully upward compatible with earlier versions of the DAP protocol. One enhancement has extended the actual length of the SYSCAP field in the CONFIG message. The field is still below the defined maximum field length as defined in earlier versions of the DAP protocol; therefore, the change is compatible with implementations conforming to DAP Versions 5.6 and later.

4.8 DECwindows Notes for Programmers

The following sections contain information concerning DECwindows.

4.8.1 VAX C Definition File Requirements

V5.1

During the VAX C installation procedure, you have the option to extract the VAX C definition files (.h files), or leave the .h files in the text library. If you extract the definition files, you are able to use #include control lines of the following form:

```
#include <filename.H>
```

All of the DECwindows sample C programs assume that the .h files were extracted; the samples contain #include <module_name.h> notation for the included files. The DECwindows programming documentation also makes this assumption.

VAX C should be installed using the option to extract the library modules.

If you have already installed VAX C and you did not extract the .h files, the DECwindows sample C programs do not work. To correct this problem, re-install VAX C and extract the .h files.

4.8.2 DECterm Port Routine

V5.1-1

A DECTERM_PORT routine exists to create a DECterm window. This routine is found in the SYS$SHARE:DECW$TERMINALSHR image. An example of linking with this routine follows:

```
$

$
```

The DECTERM_PORT portion of this routine follows:

```
DECTERM_PORT
decterm_port_sec
```

Use this routine to create a DECterm window that emulates a VT320 terminal.

- VAX Format

  VAX FORMAT VMS Status Code = DECW$TERM_PORT (display,setup_file,customization,result_dev,result_len)

The VAX format arguments are described in Table 4–1.

Table 4–1 VAX Format Argument Information

Argument	Usage	Data Type	Access	Mechanism
display	char_string	char string	read	By descriptor
setup_file	char_string	char string	read	By descriptor
customization	char_string	char string	read	By descriptor
result_dev	char_string	char string	write	By descriptor
result_len	word	word	write	Reference

- MIT C Format

  MIT C FORMAT VMS Status Code = DECwTermPort (display,setup_file,customization,result_dev,result_len)

```
DECwTermPort (display,setup_file,customization,result_dev,
              result_len)
      char *display;
      char *setup_file;
      char *customization;
      char *result_dev;
      short *result_len;
```

Arguments

- *display*

  This character string identifies the server and screen on which the created DECterm appears. If the string address is 0, the default display is used. This argument is specified as the address of a descriptor for the VAX binding and the address of a null-terminated string for the C binding.

- *setup_file*

  This character string specifies the name of the setup file. The setup file changes DECterm's initial settings. If the string address is 0, the default setup file, DECW$USER_DEFAULTS:DECW$TERMINAL_DEFAULT.DAT, is used. This argument is specified as the address of a descriptor for the VAX binding and the address of a null-terminated string for the C binding. The maximum length of the string is 200 characters.

- *customization*

  This character string specifies setup options. These strings override the default values established in resource files and the setup file. If the string address is 0, default values will not be overridden. The syntax is the same as the syntax for resource and setup files:

```
"param:  value \n param:  value \n param:  value...."
```

  (In languages other than C, replace "\n" with a line feed character, ASCII code 10.)

  This argument is passed by a descriptor or null-terminated string. The maximum-allowed length is 512 characters.

Customization parameters are not currently documented in the VMS implementation of DECwindows. In general, the recommended way to create a DECterm with customized parameters is to create a DECterm through the Session Manager, change settings through Customize, save those settings with a non-default file name, and then pass that file name as the setup parameter to the DECTERM_PORT routine.

- *result_dev*

  This character string specifies the virtual terminal device name for the created DECterm. This string is used in applications that assign the created DECterm or pass the name to a new process. This argument is specified as a descriptor for the VAX binding and as the address of a character array for the C binding. The area pointed to by the result_dev descriptor must be exactly 50 characters long; if it is shorter than that, other variables may be over-written.

- *result_len*

  This is the address of a 16-bit word. The actual length of the returned device name is written into this address. When using the VAX calling format, you can point this argument directly at the result_dev descriptor to trim the descriptor for subsequent use.

Description

DECTERM_PORT sends a message to a mailbox created by a DECterm controller, telling the controller to create a DECterm, display it on the screen, and return the associated virtual terminal device name for the caller to use in other system calls, such as $ASSIGN, $QIO, $CREPRC, or LIB$SPAWN. If you use LIB$SPAWN, you must first assign the DECterm, since LIB$SPAWN assigns and then deassigns the device. The virtual terminal will be destroyed if the last channel to it is deassigned.

The logical name pointing to the controller's mailbox has the following format:

```
DECW$DECTERM_MAILBOX_node::screen
```

where "node" is the name of the remote node and "screen" is the DECwindows screen number, which usually is 0. The value of "node" for the local node is the local node name if DECnet is running, or 0 if DECnet is disabled.

The image file for the DECterm controller is SYS$SYSTEM:DECW$TERMINAL.EXE, and it must be run with PHY_IO, SHARE, and SYSNAM privileges. Normally it is installed with the necessary privileges.

To create a DECterm on a remote system, check to see if a DECterm controller exists for that system. You can do this by finding the mailbox's logical name, and then making sure the associated mailbox actually exists (in case a controller was terminated and the logical name wasn't deleted). The controller determines which DEcwindows screen to write to by using the last display name set by the SET DISPLAY command.

The following command procedure checks for the existence of the mailbox and creates a DECterm controller if necessary (using the C program RUN_CONTROLLER given below). It then runs the program CREATE_DECTERM, which sends a message on the mailbox using DECwTermPort and then creates a process on the virtual terminal device. The C source code for CREATE_DECTERM is given after the command procedure, followed by the DCL command to compile and link CREATE_DECTERM and run the command procedure REMOTE_DECTERM.COM.

File REMOTE_DECTERM.COM:

```
$!
$! P1 = node name
$!
$ node = P1
$ if node .eqs. "" then inquire node "Node name"
$ mailbox = f$trnlnm("decw$decterm_mailbox_" + node + "::0.0")
$ if mailbox .nes. ""
$    then
$    if f$getdvi(mailbox, "exists") then goto skip_run_controller
$ endif
$ term_file = "sys$scratch:term_" + f$getjpi("","pid") + ".tmp"
$ term_file = f$parse( term_file,,,,"no_conceal" )
$ open/write term 'term_file'
$ write term "$ set display/create/node=" + node + "::0.0"
$ write term "$ run sys$system:decw$terminal"
$ close term
$ run/detach/input='term_file'-
     /output=t.tmp-
     /error=nl:-
     /process="DECterm_''node'"-
     sys$system:loginout.exe
$skip_run_controller:
$ mcr sys$login:create_decterm 'node'::0.0
$ exit
```

File CREATE_DECTERM.C:

```
/*
 * Usage is: mcr sys$login:create_decterm node::screen
 */

#include descrip     /* descriptor definitions */
#include ssdef       /* system status codes */
#include prcdef      /* stsflg bits for creating process */

main( argc, argv )
    int argc;
    char **argv;
{
    char *display;
    int status, stsflg;
    short device_length;

    /* this must be exactly 50 characters */
    char device_name[50];

    $DESCRIPTOR( command, "SYS$SYSTEM:LOGINOUT.EXE" );
    $DESCRIPTOR( input_file, "" );
    $DESCRIPTOR( output_file, "" );

    /* first parameter is the display name */
    display = argv[1];
```

```
    /* send the message to the controller */
    status = DECwTermPort( display, 0, 0, device_name,
                           &device_length );
    if ( status != SS$_NORMAL )
        printf( "DECterm creation failed, status is %x\n",
                status );
    else
        {  /* create a process that is already logged in */
        /* input from TWn: */
        input_file.dsc$w_length = device_length;
        input_file.dsc$a_pointer = device_name;

        /* output to TWn: */
        output_file.dsc$w_length = device_length;
        output_file.dsc$a_pointer = device_name;

        /* make it detached, interactive, logged in */
        stsflg = PRC$M_DETACH | PRC$M_INTER | PRC$M_NOPASSWORD;

        /* create the process */
        status = sys$creprc( 0, &command, &input_file,
                     &output_file, 0, 0, 0, 0, 0, 0, 0, stsflg );
        if ( status != SS$_NORMAL )
            printf( "Could not run LOGINOUT.EXE, status is %x\n",
                    status );
        }
}
```

To compile, link and run CREATE_DECTERM (in SYS$LOGIN:)

```
$ cc create_decterm
$ link create_decterm, sys$input/opt
sys$share:decw$xlibshr/share
sys$library:decw$dwtlibshr/share
sys$share:vaxcrtl/share
sys$share:decw$terminalshr/share
$ @remote_decterm mynode
```

4.8.3 DECterm Fonts

V5.1

All DECwindows terminal fonts are for private use by DECterm, and should not be used by other applications. There are several problems with the terminal emulator fonts:

- The terminal fonts supplied with DECwindows currently use a version of the DEC Multinational Character Set even though the font names say they use the ISOLatin1 character set. This discrepancy will be resolved in a future release of VMS by changing the encoding of these fonts to be true ISOLatin1.
- There are character set encoding problems with DEC-DECtech fonts. The character set is a 7-bit DECTechnical set that is private to Digital.
- There are missing or incorrect characters in many fonts.
- There are various spacing problems in many fonts.
- The 28-point fonts are not actually two times the height of the 14-point font (DECterm requirement).
- Line-drawing characters in many fonts (particularly Double_Wide) do not join properly.

- The Narrow, Wide, and DoubleWide 14-point fonts at 75 dots per inch (dpi) do not have the same cell size as the Normal and Bold 14-point fonts.

4.8.4 DECterm — Text Mode Locator Support

V5.1-1 The following sections describe information specific to DECterm text mode locator support.

DECterm extends the locator input model of the VT330/VT340 by supporting locator input in text mode as well as in ReGIS. This extended model is also supported by the VWS terminal emulator. The ReGIS locator is described in the *DECterm Graphics Programming Manual.*

4.8.4.1 Overview

V5.1-1 The mouse or locator input cursor is always visible on the workstation screen. When locator reporting is enabled in the terminal emulator and the locator input cursor is within a terminal window, individual locator events, such as locator button transitions or movement, may be programmed to send locator reports to the host (user application).

Each locator report includes the specific event that initiated the report, the current state of the locator keys, and the cursor's input coordinates at the time of the event.

4.8.4.2 Locator Input Model

V5.1-1 The following sections describe information specific to the DECterm locator input model.

Enabling Locator Reporting

Locator reporting can be selectively enabled from the host using a DEC private control sequence. When enabled (the power-up default is disabled), individual locator events such as locator button transitions or movement may be programmed to send locator reports to the host.

```
DECELR  -  DEC Enable Locator Reports

CSI  Ps ; Pu  '    z
              2/7  7/10

Ps may assume the following values
  0  locator disabled (default)
  1  locator reports enabled
  2  one shot (allow one report, then disable)

Pu specifies the coordinate units for locator reports
  0  (or omitted) default to character cells
  1  device physical pixels
  2  character cells
```

One-shot mode is provided for applications that desire simple graphics input similar to Tektronix GIN mode (no unsolicited reports). If parameter value 2 is selected, the next trigger event that occurs will generate a single locator report. No further locator reports will occur (the locator will be disabled), until another DECELR sequence is received.

The coordinate units for locator position reports may be selected to two of the two coordinate systems used by terminal software at the lowest level. Physical pixels is the "least common denominator", and is useful for computing sixel positions.

Interaction Of Terminal Emulator And Workstation Locator Handling

When locator reporting is enabled in the terminal emulator, button presses and mouse movement activate text locator functions, and are not used for local emulator functions such as select/stuff and quickcopy.

Reporting Locator Position

When a selected trigger event occurs such as a button press or release, the terminal transmits a locator position report as follows.

```
DECLRP  -  DEC Locator Report

CSI  Pe ; Pb ; Pr ; Pc ; Pp  &    w
                             2/6  7/7

  Pe is the event code
  Pb is the button code
  Pr is the row coordinate
  Pc is the column coordinate
  Pp is the third coordinate (page number)

Pe, the event code indicates what event caused this report
to be generated.  The following event codes are defined:

  0  -  request, the terminal received an explicit request
        for a locator report, but the locator is unavailable
  1  -  request, the terminal received an explicit
        request for a locator report
  2  -  left button down
  3  -  left button up
  4  -  middle button down
  5  -  middle button up
  6  -  right button down
  7  -  right button up
  8  -  fourth button down
  9  -  fourth button up
 10  -  locator outside filter rectangle
```

Note: The fourth button is described for completeness, and is used in graphical tablets. DECterm V1.0 supports only three mouse buttons.

Pb is the button code, ASCII decimal 0-15 indicating which buttons are depressed. The state of the four buttons on the locator correspond to the low four bits of the decimal value, "1" means button depressed

```
  0  -  no buttons down
  1  -  right
  2  -  middle
  4  -  left
  8  -  fourth

Pr is the row coordinate of the locator position in the
page, encoded as an ASCII decimal value.

Pc is the column coordinate of the locator position in the
page, encoded as an ASCII decimal value.
```

```
Pp is the page coordinate of the locator position encoded
as an ASCII decimal value.  The page coordinate may be
omitted if the locator is on page one (the default).
```

Note: Since DECterm does not support the VT330/VT340 page functions, this field will be omitted in future versions of DECterm. In DECterm V1.0 the field is always set to 1.

Each locator report includes both the specific transition that caused this event, and the current button state. This allows software to determine what event just occurred and which buttons are depressed without keeping track of previous events or button state. In a multiprocess shared locator environment, an application may not know the previous button state. This dual reporting also allows applications to recover from lost locator reports.

DECterm uses the DECwindows mouse as the locator for all windows; the locator position is never occluded by another window. Pressing a button on the locator will give the window containing the input cursor the input focus (possibly bringing that window to the top). If locator reporting is enabled in the new window, a locator report will be transmitted. This report is sent to prevent loss of information. Applications can be designed to ignore such reports if desired.

If the input cursor is inside a window, but outside the range of defined coordinates for that window, pressing a button on the locator will not generate a report. For example, when the input cursor is outside the active scrolling region, and the origin mode has been set to relative a report would not be generated. To use the locator to adjust scroll margins, the origin mode must be absolute.

If the input cursor is not contained in any window, pressing a button on the locator will have no effect on the emulator.

Filter Rectangles

Filter rectangles add filtered movement events to the list of locator transitions that can generate reports.

```
DECEFR  -  DEC Enable Filter Rectangle

     CSI  Pt ; Pl ; Pb ; Pr  '    w
                                2/7  7/7

      Pt  -  Top    boundary of filter rectangle
      Pl  -  Left   boundary of filter rectangle
      Pb  -  Bottom boundary of filter rectangle
      Pr  -  Right  boundary of filter rectangle
```

The DECEFR control sequence defines the coordinates of a filter rectangle, and activates it. Anytime DECterm detects that the locator is outside a filter rectangle, DECterm generates an outside rectangle event and the rectangle is disabled. Filter rectangles are always treated as "one-shot" events. Defining a new rectangle re-activates it.

Applications can re-define the rectangle at any time, even if it is already active. If a rectangle that does not contain the locator is specified, the terminal will generate an outside rectangle report immediately and deactivate it.

Note: Because of a problem in DECterm V1.0, the reported position in this situation may not be correct. The reported position contains the last locator position returned by a button press or motion event, rather than the current position

Pt, Pl, Pb, and Pr are in coordinates units specified by the last DECELR sequence. The filter rectangle includes the boundaries (similar to other rectangular area operations). The origin is coordinate pair 1:1 in the upper left corner. If any parameters are omitted, they are defaulted to the current locator position. Sending DECEFR with no parameters will cause the application to be notified for any locator movement ("unfiltered movement event").

DECELR always cancels any previous filter rectangle definition. This guarantees there will never be an outstanding filter rectangle when locator reports are enabled by an application.

Selecting Locator Events

The locator events that are allowed to generate unsolicited reports may be individually selected using the Select Locator Events control. The locator can report both up and down transitions when the exact sequence of button activiations is significant. This control allows application software to select the events it wants reported.

```
DECSLE  -  Select Locator Events

CSI  P...P  '    {
            2/7  7/11

P...P is one or more selective parameters which may assume
the following values:

  0  respond only to explicit host requests
     (default, also cancels any pending filter rectangle)
  1  report button down transitions
  2  do not report button down transitions
  3  report button up transitions
  4  do not report button up transitions
```

Requesting A Locator Position Report

The host may explicitly request a locator position report any time locator reporting is enabled (DECELR). Upon receiving such a request, the terminal will immediately send a single locator report (DECLRP) with event code 1 indicating the last locator position. If the session receiving the request is not currently active (the locator is being used in another session), the last known locator position and state for this session will be used. If the locator is disabled or unavailable, the report will specify event code 0.

Note: Because of a problem in DECterm V1.0, DECterm will not issue a locator report unless locator reporting is enabled.

```
DECRQLP  -  DEC Request Locator Position

CSI Ps  '    |
        2/7  7/12
  Ps
```

```
0  (or omitted) default to 1
1  transmit a single DECLRP locator report
all others ignored
```

4.8.4.3 Locator Device Support

V5.1-1

Locator support is currently planned as an extension to the level 3 character cell architecture. The primary device attributes response will report the text locator extension as parameter value 29.

Host software may request a Device Status Report (DSR) to determine whether a locator is available. Upon receiving the appropriate DSR request, DECterm will respond that the locator is ready.

Host software may also request the locator type or identification. DECterm will always respond "Locator type cannot be determined".

```
Locator Device Status Report (DSR)

Host request locator device status        CSI ? 55 n
Locator ready                             CSI ? 50 n
Locator busy                              CSI ? 58 n
```

Note: DECterm V1.0 incorrectly replies with CSI ? 59 n.

```
Host requests identification of locator   CSI ? 56 n
Locator type cannot be determined         CSI ? 57 ; 0 n
```

4.8.5 DECwindows Problems Corrected

V5.2

The following DECwindows problems have been corrected in VMS Version 5.2:

- If a client used the CopyGC request, the DECwindows server could continue to grow until the virtual page count quota VIRTUALPAGCNT was exhaused.
- If a client used bad coordinates when specifying a clip region in a graphics context, the DECwindows server could continue to grow until the quota VIRTUALPAGCNT was exhausted.
- If a client drew to a pixmap using a clip region in the graphics context, the DECwindows server could continue to grow until the quota VIRTUALPAGCNT was exhausted.
- If a client did not clear the undefined bits in the flag field of the XColor structure during a StoreColors request, the DECwindows server could go into an infinite loop.
- The DECwindows server in some cases failed to paint window backgrounds correctly on VAXstation 3520 and 3540 systems.
- Certain drawing operations were not immediately started by the driver for GPX color graphics systems. The new image GADRIVER.EXE improves the performance of the XGetImage and XPutImage operations.
- Changing the dash pattern in a graphics context by using XSetDashes was ignored on GPX systems.

- Polylines whose end points were greater than 4095 pixels apart were not drawn correctly by the GPX DECwindows server.
- XDrawText requests to a partially occluded window failed to draw text with certain strings and fonts that had a very large maximum character width.
- The DECwindows server could corrupt window displays when using the writing function GX_AND_INVERTED to a 1-bit deep pixmap.
- Applications calling the DECW$TERM_PORT routine inadvertently had all timer requests cancelled.
- A string in the routine DECW$TERM_PORT was improperly terminated, requiring the use of the routine FIX_DECWTERMPORT_BUG.
- The DECwindows server sometimes disconnected clients prematurely when, after a quick retry, the server assumed that the client was not functioning properly. This problem has been fixed by increasing the time interval between retries.

4.8.6 DECwindows Server and Driver Notes

V5.1

The DECwindows server is Digital's implementation of the X Window System's server. The server is the component of the architecture that allows application interfaces to look the same on all supported systems.

Device drivers process requests from the server to the display and from the input devices to the server.

VAXstation configurations are subject to the following notes:

- On a VAXstation 2000, the keyboard and mouse serial lines are TTA0 and TTA1, respectively. Terminal operations such as SET/SHOW TERM, or SET HOST DTE do not work for these devices. The terminal lines are TTA2 and TTA3.
- The server looks for a number at the end of its process name. If it finds a number, it considers that number to be its server number and listens for connections on that number rather than on 0. The default value is 0. This is normally resolved by the DECwindows startup command files.
- You cannot depend on the values of BlackPixel and WhitePixel being 0 and 1. Their values will differ depending on the hardware.
- Put Image is restricted to a maximum width of 1024 pixels for GPX servers.
- The X11 protocol allows the server to "arbitrarily transform" the components of a cursor in order to meet the requirements of the display. Since neither the VAXstation 2000 nor the VAXstation II monochrome workstations support recoloring cursors, you should not expect the colors you specify for the cursor to actually be reproduced on the hardware.

- The DECwindows server contains a facility called a **condition handler** that detects problems that might otherwise cause the server to stop and tries to let the server continue. When the condition handler intercepts a problem like this, it either sends an "Implementation" error to the client, disconnects the client, or both.

  When the condition handler recovers from an error, the server might lose resources such as memory. Therefore, after a number of these interceptions, the condition handler broadcasts a warning message to all users on the workstation indicating that the server may be running in a degraded mode and suggesting that it be restarted. If you get messages like this, you should restart the server at the next convenient opportunity. Enter the following command from a privileged account (it may be one logged into a DECwindows terminal emulator window) to restart the server:

  ```
  $
  ```

- There are a number of X11 protocol requests and corresponding XLIB requests that take an unsigned word ("short" in C) as a width or height argument. A common application mistake is to calculate a width or height incorrectly and end up with a small negative number. The protocol interprets this as a large unsigned number. The DECwindows server does not deal with large widths and heights correctly in many cases. You may get an "Implementation" error returned by the condition handler, or by the server directly detecting that the number is too large.

  Note that the numbers that the server has trouble with are generally greater than 32767, or combinations of coordinates and width/height that add up to greater than 32767. Coordinates in the range of any supported display devices are much smaller than this number.

  The server acts in an unfriendly way when a client does not read its events often enough. After the server runs out of buffer space when trying to write events, errors, or replies to a client, it hangs in HIB state and retries the write to the client periodically. The server does not service other clients while it is in this state.

  After a timeout period, the server disconnects the offending client and services the remaining clients. This problem can happen when a client is working slowly and generating a lot of requests. However, the most common occurrence is when the client is being debugged. Future releases of the server will deal more gracefully with this problem. In the meantime, there is little possibility of a workaround, except to process events as often as possible.

- Under some circumstances the XCopyGC routine causes the server's memory use to grow slightly. If you do a large number of XCopyGC requests, the server gets larger and slower until it starts returning "Implementation" errors to the client. There is no known workaround other than to restart the server.

4.8.7 DECwindows Server — Increasing the Limit on the Number of Clients

V5.2

The DECwindows Server is currently limited internally to a maximum of 32 clients. (Note that all DECterm windows started by the session manager count as a total of only one client). However, there are other limitations that may further restrict the number of clients that can connect to a server.

One such additional limitation is the Enqueue Quota with which the server is running. Any client that connects using the local transport mechanism will use up two lock-queue entries. Thus, if the server has an Enqueue Quota of 30, a maximum of 15 local clients will be allowed.

If you need a larger Enqueue Quota on the server, you can increase the value of the SYSGEN parameter PQL_DENQLM (the default enqueue limit for any process that does not specify the enqueue limit when it is created).

4.8.8 Xlib Programming Notes

V5.1

The following restrictions apply to the Xlib programming library routines.

- The PUT IMAGE routine does not implement image format conversions for Z pixmap images with a depth greater than one if the depth does not match the depth of the server.
- If you call the CIRCULATE SUBWINDOWS and CIRCULATE SUBWINDOWS DOWN routines with a **direction** argument of Lower Highest, the routines do not circulate the subwindows if the highest mapped child that occludes other children is not completely visible.

  One instance of this case is when the highest mapped child that occludes other windows is clipped to the parent window. Although the child is not occluded, it is not totally visible.

  Additionally, if a totally visible window is found, lowering it to the bottom of the window stack can result in screen corruption.

  A workaround for this problem is to use the CONFIGURE WINDOW routine to alter the stacking order of the child windows. In addition to providing an alternative method for changing windows stacking order, CONFIGURE WINDOW does not expend time finding the highest occluding visible window.

4.8.9 XUI Toolkit Notes

The following information describes specific features of the XUI Toolkit:

VMS Version 5.1 Notes

V5.1

The following notes pertain to VMS Version 5.1.

- **Help widget**

  The Help widget does not support XtSetValues of many text resources. This will be corrected in a future release.

- **Unseen LeaveWindow events**

  There is a problem with widgets that pop up spring loaded widgets directly over themselves. The first widget does not see the LeaveWindow event that is produced as the popped up widget is mapped into the pointer location. This is due to a problem in the MIT R3 Intrinsics event dispatching mechanism.

  For example, a widget specifies the following translation syntax:

  ```
  <EnterWindow>: highlight( )
  <LeaveWindow>: un_highlight( )
  <Btn2Down>: popup_menu( )
  ```

  When the pointer enters the widget's window, the widget highlights. When MB2 is pressed, the popup menu is displayed. A LeaveWindow event should be dispatched to de-highlight the widget as the pointer is moved into the popup menu. This LeaveWindow event is not delivered and the widget is left in the highlighted state when the menu pops down.

  This will be fixed in a future release.

- **Dialog Box Race Condition**

  XUI Toolkit dialog boxes perform an XGrabKey on the TAB key so that they can "synchronously" transfer focus to the next child within the Dialog Box. If a Dialog Box receives a TAB key while the Toolkit is "filtering" events (for example, while another modal dialog box is up), the original Dialog Box does not see the TAB event and never calls XAllowEvents to unfreeze the keyboard. You must quit the application and restart it to unfreeze the keyboard.

  This will be fixed in a future release.

- **Attached Dialog Box**

  A call to XtSetValues on the attachments of a child of an attached dialog box can result in an infinite loop. Attachments should be set up when the child is created. In most cases this problem can also be avoided by performing the XtSetValues while the widget is unmanaged.

- **Listbox widget**

  The Shift/MB1 double-click for the extend-confirm callback is not supported. This will be fixed in a future release.

- **Cut and Paste**

  The cut and paste routines do not work for transferring data between applications running on host machines having different byte orders. This will be fixed in a future release.

VMS Version 5.2 Notes

V5.2

The following notes pertain to VMS Version 5.2.

- **Changes to the XUI Toolkit**

  In VMS Version 5.2 of the XUI toolkit, you can initialize both the Intrinsics and DRM as many times as the application requires. This is an extension of the MIT R3 Intrinsics and should not be used by applications that want to remain R3 compatible. This change has been proposed to the X Consortium for inclusion in R4.

- **Discrepancies between the XUI Toolkit and the MIT R3 Intrinsics**

  In VMS Version 5.2, the version number of the Intrinsics supplied as part of the DECwindows kit does not match that of the MIT R3 Intrinsics. The MIT R3 Intrinsics version number is 11003, while the XUI Intrinsics are 7001. Applications or widgets that switch between the two Intrinsics should be aware of this.

  The XUI routine XtNameToWidget does not conform to the MIT R3 Intrinsics. The MIT R3 specification states that the first component of the names parameter is matched against the children of the passed reference widget; the XUI implementation matches the first component of the names parameter against the reference widget, not the children. To use the XUI version, add the name of the reference widget to the beginning of the name list.

- **Restriction on Defining Accelerators for Gadgets**

  When you use accelerators on push-button and toggle-button gadgets, only the first gadget child of a widget parent may have a # operator, such as #override, in its button accelerator specification. All gadget button accelerators of a widget parent have the same # operator as the first gadget child.

  See the *VMS DECwindows Guide to Application Programming* for information about specifying an accelerator for a push-button or toggle-button gadget and for information on the syntax of creating an accelerator specification.

4.9 DCL Subroutine Entry Point Scoping Modifications

V5.1

To make the scoping of subroutine entry points more intuitive, the following restrictions will be added to a future release of VMS:

- Subroutine entry points that are defined within another subroutine are local to that subroutine. You cannot call a subroutine if the subroutine entry point is within a separate subroutine block. For example, the following call will **not** work in a future release of VMS:

```
$ CALL BAR
$
$ MAIN: SUBROUTINE
$
$     BAR: SUBROUTINE
$     ENDSUBROUTINE
$
$ ENDSUBROUTINE
```

The following call **will** work because BAR is defined within MAIN:

```
$ MAIN: SUBROUTINE
$
$     BAR: SUBROUTINE
$     ENDSUBROUTINE
$
$ CALL BAR
$ ENDSUBROUTINE
```

- If a subroutine entry point is located within an IF-THEN-ELSE block, you cannot call this subroutine from outside the IF-THEN-ELSE block. For example, the following call will not be allowed in a future release of VMS:

```
$ IF 1
$ THEN
$    FOO:SUBROUTINE
$    ENDSUBROUTINE
$ ENDIF
$ CALL FOO
```

- Every SUBROUTINE command must have a matching ENDSUBROUTINE command to delimit the subroutine. This is not a new restriction, but it will be more strictly enforced.

  In the following example, the entry point for subroutine B is defined within subroutine A because there is not an ENDSUBROUTINE to delimit A (i.e., the EXIT is not a delimiter of A). Therefore, subroutine B is inaccessible from outside subroutine A.

```
$ A: SUBROUTINE
$ EXIT
$
$ B: SUBROUTINE
$ ENDSUBROUTINE
```

4.10 Directory Size Limitation Removed — Effect on RSX-11 Compatibility Mode

V5.2

VMS Version 5.2 contains file-system enhancements that substantially improve the performance of large directories. In addition, the former restriction of directory size to 1024 blocks has been removed. Directory size is now limited only by the availability of contiguous disk space.

Lifting this size restriction may have an impact on programs that execute in RSX-11 compatibility mode and use wildcards in file operations. Because the RSX-11 wildcard context cannot address directories larger than 1024 blocks, such programs cannot process files in a directory that exceeds 1024 blocks in size. This is a permanent restriction of the RSX-11

environment, and users must avoid using directories exceeding 1024 blocks in this environment.

4.11 Failure of Customer-Written Programs Using $CMKRNL

V5.2

The VAX Procedure Calling Standard prohibits the exchange of information between calling and called procedures by means of any general purpose register with the exception of registers R0 and R1, which are reserved for the return of status from the called procedure to the calling procedure. As described in the *Introduction to VMS System Routines*, registers R2 through R11 must be preserved across a procedure call. If a called procedure uses these registers, it must save and restore them using the procedure entry mask mechanism.

Certain customer-written programs that violate these requirements of the VAX Procedure Calling Standard will fail under VMS Version 5.0, even though they may have run successfully under previous versions of VMS. Specifically, these programs contain code that invokes a procedure by means of the $CMKRNL system service, and attempt to pass context to that procedure in R2.

A change in the system service dispatcher in VMS Version 5.0 exposes this coding infraction. Prior to VMS Version 5.0, the dispatcher modified only R4; as of VMS Version 5.0, it also uses R2. In future releases of VMS, it may use other registers.

These customer-written programs (and other programs using prohibited registers) should be fixed to comply with the VAX Procedure Calling Standard and pass context to the procedure called through $CMKRNL in an appropriate way, such as within an argument list. See the *Introduction to VMS System Routines* for additional information.

4.12 File Definition Language (FDL)

The following sections pertain to the File Definition Language (FDL).

4.12.1 Restriction on Comments Containing Semicolons

V5.0

The File Definition Language (FDL) no longer processes files with comment lines containing semicolons. However, you can use a semicolon on a comment line if the line is enclosed within quotation marks. For example:

```
!"This line is okay; there are quotes setting off the comment"
```

4.12.2 Support for Printing with Vertical Format Units (VFU)

V5.2

The VMS File Definition Language (FDL) Facility now supports printers that have Vertical Format Units (VFU). You implement this by using the following bit configuration for the control bytes in your RMS records and

by specifying the PRINT keyword for the CARRIAGE_CONTROL attribute of the RECORD Section in FDL:

```
Bit 7  Bit 6  Bit 5  Bit 4  Bit 3  Bit 2  Bit 1  Bit 0
  1      1      0      0      x      x      x      x
```

This encoding directs the device to skip to the VFU channel (1 to 16) specified by bits 3 to 0. Devices that do not have hardware VFUs translate each of these codes as a 1-line advance.

4.13 GBBDriver — New Output Driver

V5.2

A new output driver (GBBDriver) has been developed for the 3520 and 3540 VAXstations. The existing output driver GCBDriver supports output for the new VAXstation 3100. For the server programmer, a new function modifier (GB$K_LEGGS_WAIT_FOR_PKT) in the output-buffered and output-direct FDT $QIO call has been added in the output driver. This expands the existing packet wait $QIO function in output drivers to support the wait function for Low End Graphics Subsystem (LEGSS) packets.

4.14 IF-THEN-ELSE Construct and $STATUS Symbol

V5.2

Most DCL commands generate status values when they complete. However, there are several commands that do not change the values of $STATUS and $SEVERITY, for example, IF, GOTO, CONTINUE, and STOP. A list of these commands can be found in the *Guide to Using VMS Command Procedures*.

The IF-THEN-ELSE-ENDIF construct was incorrectly setting $STATUS, which masked the resulting status condition from commands executed within the block. In VMS Version 5.2, this has been fixed to maintain the last value of $STATUS set inside an IF-THEN-ELSE-ENDIF block. A command procedure can then test the value of $STATUS following the ENDIF command.

4.15 Local Area Terminal (LAT) — LTDRIVER Changes

This section describes changes made to the LTDRIVER.

4.15.1 Problem Corrected

V5.0

The LTDRIVER now supports the use of the TT$M_BREAK feature provided by the terminal driver. The DECserver 100 does not support the use of this feature, however.

4.15.2 New QIO Support

V5.0

There are new port driver QIOs supported by LTDRIVER that allow an applications port to be remapped and allow a new static rating value to be applied to an existing service name.

4.16 LIBRARIAN Routines — Caution when Using Locate Mode

V5.0

When you use the Librarian Utility (LIBRARIAN) in locate mode, the contents of a descriptor may not point to an internal LBR buffer for subsequent LBR routine calls.

4.17 Linking Workstation Applications on Nonworkstation Systems

V5.0

The UISSHR.EXE executable image allows you to link workstation applications on nonworkstation systems. VMS Version 5.0 updates the UISSHR.EXE image to include all UIS calls supported in Version 4.0 of the VMS Workstation software.

4.18 Locating Nonpaged System Patch Space

V5.1

When using the XDELTA debugging tool to debug a device driver not supplied by Digital, a system programmer might need to construct a complex breakpoint or store an XDELTA command string in nonpaged system patch space. The *VMS Device Support Manual* discusses these needs, but contains outdated information about how the programmer can locate this space.

VMS Version 5.0 introduced a change in the composition of the VMS executive image. No longer does the VMS executive comprise a single, large, static executive image, but rather a set of vectors and a set of independently loadable images. The system map file (SYS$SYSTEM:SYS.MAP) does not contain information useful in determining the location of local symbols within these loadable images.

Each of the loadable images of the executive contains an area reserved as nonpaged system patch space. In each loadable image, the symbol PAT$A_NONPAGED contains a descriptor that identifies the location and size of the unused nonpaged system patch space in that image. This descriptor has the following form:

```
PAT$A_NONPAGED::
     .LONG     size-in-bytes
     .LONG     offset to patch-space-start-address
```

A process having suitable privileges can access unused system patch space in any of the loadable images of the VMS executive. For instance, a system programmer debugging a device driver can deposit an XDELTA command string in the nonpaged system patch space of any of the loadable images.

To determine the size of patch space and its starting address in any given loadable image of the VMS executive, perform the following steps:

1 Enter the following commands to display a list of all images of the VMS executive that have been loaded into memory:

```
$
SDA>
SDA>
```

The XDELTA command ;L also displays a list of the loaded images.

2 Note the base address of the image whose patch space you want to use.

For example, you might have selected PROCESS_MANAGEMENT and determined its base address to be $80127C00_{16}$.

3 Determine the image value of the nonpaged system patch space descriptor (PAT$A_NONPAGED) in the selected image.

For example, to determine the image value of PAT$A_NONPAGED in PROCESS_MANAGEMENT, enter the following commands from the DCL prompt:

```
$
$
```

For example, you might have determined the image value of PAT$A_NONPAGED in SYS$LOADABLE_IMAGES:PROCESS_MANAGEMENT.EXE to be 6544_{16}.

4 Using the Patch Utility, locate PAT$A_NONPAGED in the image and examine its contents.

For example, the following session locates and examines PAT$A_NONPAGED in SYS$LOADABLE_IMAGES:PROCESS_MANAGEMENT.EXE:

```
$
PATCH>
00006544:  00000077
PATCH>
00006548:  00000F85
PATCH>
```

In this example, the Patch Utility output shows that there are 77_{16} bytes remaining in the nonpaged system patch space of PROCESS_MANAGEMENT and that the available patch space starts at offset $F85_{16}$ into the image.

5 Calculate the starting address of nonpaged system patch space in the selected loadable executive image by adding the offset from the descriptor to the base address of the image you determined in Step 2.

For instance, the base address of nonpaged system patch space in SYS$LOADABLE_IMAGES:PROCESS_MANAGEMENT.EXE is $80127C00_{16}$ +$F85_{16}$, or $80128B85_{16}$.

4.19 Poor Man's Lockdown

V5.1

Certain privileged code, written prior to VMS Version 5.0, utilizes a technique, commonly known as "poor man's lockdown," whereby one or two pages of code are locked into a process or system working set as a side-effect of elevating IPL. Such code has one of the following forms:

```
ASSUME  10$-. LE 511        ; Check for contiguity of pages
SETIPL   10$
   .
   .
   .
;Code to be locked into memory
   .
   .
   .
RSB
10$:    .LONG  IPL$xxxx
```

The effect of this coding technique is that, because the system must determine the value of the argument to the SETIPL macro from location 10$, it must fault into memory the page in which 10$ resides. As a result, before the code actually elevates IPL, the pages in which the SETIPL macro and 10$ reside will become memory-resident. In this way, the code can avoid a page fault while executing the code between the SETIPL and 10$ at elevated IPL. The ASSUME macro guarantees that the pages to be faulted are contiguous.

This technique has several limitations:

- It cannot lock more than two virtually contiguous pages.
- Beginning with VMS Version 5.0, it is only useful in locking per-process pages, not system pages. In a VMS multiprocessing system, a page in the system working set could be faulted in by one processor, only to be removed from the system working set by another processor.

  To lock system pages, you must use the LOCK_SYSTEM_PAGES and UNLOCK_SYSTEM_PAGES macros as described in the next section. (Note that you cannot use these macros to lock per-process pages in memory.)
- Prior to VMS Version 5.0, IPLs were the means by which system tasks were prioritized and access to system data was synchronized. Code executing at an elevated IPL would effectively block other code in the system from executing at or below that IPL. In VMS Version 5.0, which introduced symmetric multiprocessing, and later versions of the operating system, merely raising IPL does not synchronize systemwide activity or enforce orderly access to data.

  Sometimes it might be necessary only to block tasks or synchronize activity on the local processor. In these instances, raising IPL provides sufficient synchronization and "poor man's lockdown" behaves as it did before VMS Version 5.0. For instance, use of "poor man's lockdown" to lock a code segment executing at IPL$_RESCHED effectively prevents process deletion and rescheduling while the code executes at nonpageable IPL.

  However, if a locked code segment must access system data structures at an elevated IPL—for instance, at IPL$_SYNCH—it must obtain the spin lock associated with the database by using one of the spin lock synchronization macros (LOCK, FORKLOCK, or DEVICELOCK). After accessing the data, it must release the acquired spinlock by invoking UNLOCK, FORKUNLOCK, or DEVICEUNLOCK. Appendix B of the *VMS Device Support Manual* discusses the spin lock synchronization

macros. A description of LOCK_SYSTEM_PAGES and UNLOCK_SYSTEM_PAGES follows.

LOCK_SYSTEM_PAGES

Locks a paged code segment in system memory.

FORMAT

LOCK_SYSTEM_PAGES *[startva],endva[,ipl]*

PARAMETERS

[startva]

System virtual address in the first page to be locked. If the **startva** argument is omitted, the starting virtual address defaults to the current PC.

endva

System virtual address in the last page to be locked.

[ipl]

IPL at which the locked code segment is to execute. If the **ipl** argument is omitted, the locked code segment executes at the current IPL.

DESCRIPTION

The LOCK_SYSTEM_PAGES macro calls a memory management routine to lock as many pages as necessary into the system working set. The macro accepts a virtual address that indicates the first page to be locked and a virtual address that indicates the last page to be locked. You can also supply the IPL at which the code in the locked pages is to execute.

The LOCK_SYSTEM_PAGES macro executes under the following conditions:

- The LOCK_SYSTEM_PAGES macro should be used only on system virtual addresses.
- All pages requested in a single LOCK_SYSTEM_PAGES macro call must be virtually contiguous. If you must lock discontiguous memory, you must invoke the LOCK_SYSTEM_PAGES macro once for each page or set of contiguous pages.
- You must invoke LOCK_SYSTEM_PAGES at IPL 2 or lower to allow pagefaulting to occur.
- When the locked code segment is finished, it must invoke the UNLOCK_SYSTEM_PAGES macro to release all previously locked pages. In other words, there must be exactly one UNLOCK_SYSTEM_PAGES macro call per LOCK_SYSTEM_PAGES macro call.
- When it invokes the UNLOCK_SYSTEM_PAGES macro, the code must ensure that the stack is exactly as it was when the LOCK_SYSTEM_PAGES macro was invoked. That is, if the code has pushed anything

on the stack, it must remove it before invoking UNLOCK_SYSTEM_PAGES.

- If the **ipl** argument is supplied to the LOCK_SYSTEM_PAGES macro, the locked code segment must invoke the appropriate system synchronization macros (LOCK, FORKLOCK, or DEVICELOCK and UNLOCK, FORKUNLOCK or DEVICEUNLOCK) to obtain and release any spin locks required to protect the resources accessed at the elevated IPL.
- If it specified the **ipl** argument to the LOCK_SYSTEM_PAGES macro, the code segment must restore the previous IPL, either explicitly, through the use of the **ipl** argument to the UNLOCK_SYSTEM_PAGES macro, or through the use of one of the system synchronization macros.

EXAMPLE

```
                TSTB    (R0)                    ; Fault in page
30$:            LOCK_SYSTEM_PAGES,-
                        END=100$                ; Lock down pages
                LOCK    LOCKNAME=MMG,-          ; Synch with MMG
                        SAVIPL=-(SP)            ; Save current IPL
                MOVL    W^MMG$GL_SYSPHD,R3      ; Get system PHD

                        .
                        .
                        .


                UNLOCK  LOCKNAME=MMG,-          ; Unlock MMG
                        NEWIPL=(SP)+            ; Restore IPL
                UNLOCK_SYSTEM_PAGES             ; Unlock pages
100$:
```

In this example, the LOCK_SYSTEM_PAGES macro locks all pages between labels 30$ and 100$ into the system working set. The UNLOCK_SYSTEM_PAGES macro does the co-routine return to unlock those pages locked by the LOCK_SYSTEM_PAGES macro call.

UNLOCK_SYSTEM_PAGES

Terminates a request to lock down a series of system pages.

FORMAT UNLOCK_SYSTEM_PAGES *[ipl]*

PARAMETERS

[ipl]
IPL at which to continue execution.

DESCRIPTION

The UNLOCK_SYSTEM_PAGES macro terminates a request to lock down a series of contiguous system pages. In a code segment that uses this locking technique, there must be exactly one UNLOCK_SYSTEM_PAGES macro call per LOCK_SYSTEM_PAGES macro call. When the locked code segment completes, it must invoke the UNLOCK_SYSTEM_PAGES macro to release all previously locked pages.

The UNLOCK_SYSTEM_PAGES macro executes under the following conditions:

- When it invokes the UNLOCK_SYSTEM_PAGES macro, the code must ensure that the stack is exactly as it was when the LOCK_SYSTEM_ PAGES macro was invoked. That is, if the code has pushed anything on the stack, it must remove it before invoking UNLOCK_SYSTEM_ PAGES.
- If it specified the **ipl** argument to the LOCK_SYSTEM_PAGES macro, the code segment must restore the previous IPL, either explicitly, through the use of the **ipl** argument to the UNLOCK_SYSTEM_ PAGES macro, or through the use of one of the system synchronization macros (UNLOCK, FORKUNLOCK or DEVICEUNLOCK). If it lowers IPL, the locked code segment must invoke the appropriate system synchronization macro to release any spin locks that were required to protect the resources accessed at the elevated IPL.

4.20 LPA11–K Driver (LADRIVER) — Changing Timeouts Allowed

V5.0

The driver for the LPA11–K (LADRIVER) times out all $QIOs after two seconds if they have not completed. The driver does not provide any parameters that allow the user to change the length of the timeout.

In VMS Version 4.4 and subsequent versions, the timeout period that is applied to all $QIOs can be changed with the following patch commands executed from a suitably privileged account:

```
$
PATCH>
PATCH>
OLD>
OLD>
NEW>
NEW>
PATCH>
PATCH>
```

Substitute the desired timeout value for the "0000003C" in the example above. When you reboot, the system loads the new copy of the driver containing the new timeout value.

4.21 VAX MACRO Notes

The following sections describe the restrictions, fixed problems, and known problems for VAX MACRO.

4.21.1 VAX MACRO Product Requirements

V5.0

VAX MACRO requires that your system run the minimum versions of the following software products:

- VMS Version 4.4 or greater
- VAX LSE Version 2.2 or greater
- SCA Version 1.1 or greater
- VAX DEBUG Version 5.0 or greater (for enhanced VAX DEBUG support only)

4.21.2 Caution on Using NOP Instruction as a Delay Mechanism

V5.0

Digital recommends that you do not use the VAX MACRO instruction NOP (No Operation) as a means of delaying program execution.

The delay time caused by the NOP instruction is dependent on processor type. For instance, the VAX 8600, VAX 8650, VAX 8800, VAX 8700, VAX 8550, or VAX 8530 processors execute the NOP instruction more quickly than other VAX processors.

Whenever you must have a program wait for a specified time period, you should use a macro or code sequence that is not dependent on the processor's internal speed. For example, you can use the TIMEDWAIT macro, which is documented in the *VMS Device Support Manual*. You

can also use the Set Timer ($SETIMR) and Wait for Single Event Flag ($WAITFR) system services, as described in the *VMS System Services Reference Manual*, to force such delays.

4.21.3 VAX MACRO Problems

V5.0

The following is a list of all known problems in VAX MACRO Version 5.0:

- Use of the /DIAGNOSTICS command qualifier together with the /ANALYSIS_DATA command qualifier (that is, in the same command) causes the assembler to incur an access violation. To avoid this problem, use these two qualifiers in separate commands.
- Source line correlation DST (Debug Symbol Table) records are generated incorrectly for repeat loops (.REPEAT, .IRP, and .IRPC constructs). This problem causes VAX DEBUG to display incorrect source records for all but the first iteration of a loop.
- If there are two or more errors in a VAX MACRO source record, the generated diagnostic message does not point correctly (using the "!" character) at other than the first error.
- When concatenated source files are assembled, the records from each source file are numbered sequentially starting from 1 in the assembly listing. Thus, the same line number can be assigned to multiple source records within a single module. This problem may also occur in the generated DST, causing VAX DEBUG to display incorrect source records. To avoid this problem for debugging purposes, manually concatenate the source files (forming a single input file) before assembly.
- Assembly errors may occur if the string ".REPT" is found in comments included within a .REPT loop.
- The %EXTRACT macro string operator is parsed by the assembler even if it is found in an unsatisfied conditional block or in a comment.
- If the total length of an actual argument in a macro call is larger than that which the assembler can handle (about one block of data), the "%MACRO-E-LINTOOLONG, Line too long" error message is normally issued. For some lengths, however, only messages indicating some kind of incorrect syntax are issued. For still other lengths, the assembler incurs an access violation.
- No diagnostic message is given if a keyword actual argument is associated with a created local label. To avoid this problem, do not specify a keyword actual argument for a formal argument that is a created local label. (Refer to the "Created Local Labels" section of the "Macro Arguments and String Operators" chapter in the *VAX MACRO and Instruction Set Reference Manual*.
- For branch instructions located in the body of a macro, the assembler does not always generate a diagnostic for a branch out of range.
- The assembler may abort with "%SYSTEM-F-RADRMOD, reserved addressing fault . . . " if the register mask operator used with the .ENTRY directive is mistyped as "M^" or "M" instead of "^M".

- When evaluating a large expression, the assembler may incorrectly generate a "store word" instruction instead of the correct "store longword" instruction—causing the linker to generate a truncation error.
- The assembler does not correctly compute the required size of an absolute offset unless it is defined in the same PSECT as the code being generated.
- The %LENGTH operator does not work within .REPEAT blocks.
- If the .IIF directive is not contained within one line of source code, then the continuation lines are assembled even if the condition is not satisfied. To avoid this problem, express the condition to be tested and the conditional assembly block completely within the line containing the .IIF directive. If the use of a continuation line is necessary, use the .IF directive instead of .IIF.
- When .DISABLE is specified with multiple items, some of the items may be ignored. To avoid this problem, use individual .DISABLE directives for all the items.
- The assembler does not correctly evaluate expressions containing the arithmetic shift operator. For example, the expression <1@30+1@31-2> is evaluated as <<<1@<30+1>>@31>-2>.

  Because there is no operator precedence in VAX MACRO, the first arithmetic shift operation should occur before the add operation. To avoid this problem, you can force the order of evaluation to be correct by placing angle brackets around subexpressions containing the arithmetic shift operator.
- A "branch to subroutine" instruction immediately followed by a directive which computes an expression containing the ASCII operator (^A) may yield a "MACRO-W-DATATRUNC, Data truncation error" diagnostic. To avoid this problem, place a NOP (or any other) instruction immediately after the "branch to subroutine" instruction.
- The assembler may not handle quadword literals (or any literal larger than 32 bits) correctly. For example, the instruction MOVQ #<5@30>,R0 does not carry the high bit of the number 5 into R1; the bit is lost with no diagnostic message.
- If the last character in an .IRP argument list is a comma, the assembler expands the body of the .IRP loop n-1 times, where *n* is the number of elements (including null elements) in the list. To avoid this problem, always terminate the .IRP argument list with an element that is not null (that is, do not terminate the list with a comma).
- Attempting to assemble a VAX MACRO program using a command line in excess of 512 characters results in a corrupt object file.
- The assembler cannot display listing line numbers greater than 64K. To avoid this problem, do not assemble modules containing more than 64K source records. (Note the total size of a module when concatenating source files.)

4.22 Message Router Version 3.0 Installation

V5.0

The following list contains problems that occur when you use Message Router Version 3.0 with VMS Version 5.0:

- The Message Router VMS Gateway (MRGATE) requires the MAIL image to run with SYSPRV. Unlike VMS Version 4.n, VMS Version 5.0 does not install the MAIL image with SYSPRV. Therefore, if you are running MRGATE on VMS Version 5.0, log in to the SYSTEM account and use the following command to assign SYSPRV to the MAIL image:

  ```
  $
  ```

- If you are running the Directory Service part of Message Router Version 3.0 on VMS Version 5.0, you may see the following error messages:

  ```
  DDS-E-OPSYS, Operating system interface error
  LIB-E-BADBLOADR, bad block address
  ```

  These messages indicate that part of the virtual memory is not being released. You will see these error messages when new nodes join the Directory Service network and when the Directory Service servers are running (for example, when you use the MBMAN SUSPEND command). These are erroneous messages; the Directory Service continues working.

- The SUBMIT command works differently in VMS Version 5.0 from how it works in VMS Version 4.n. In Version 4.n, any logical names specified in the /LOG qualifier are translated at submission time. In Version 5.0, the logical names are translated when the job starts, at which point the logical names may not have been defined.

  If you are using exception reporting in Message Router Version 3.0, the change to the SUBMIT command in VMS Version 5.0 can cause the exception reporting batch submission to fail. The batch jobs are entered on the batch queue, but the jobs fail and do not leave a log file indicating the reason for failure, because no logical name is defined for the log file.

  To avoid this problem, edit the command procedure that starts up the exception reporting batch jobs (SYS$COMMON:[SYSMGR]MB$$ER_START.COM) as follows:

  1. Change the /LOG qualifier of the SUBMIT command from /LOG=MB$SCRATCH:MB$'component'_'node'.LOG to /LOG=MB$ROOT:[MB$SCRATCH]MB$'component'_'node'.LOG
  2. Change the /LOG qualifier of the SUBMIT command from /LOG=MB$SCRATCH:MB$NET_"mb$$mgmnt_node'.LOG to /LOG=MB$ROOT:[MB$SCRATCH]MB$NET_'mb$$mgmnt_node'.LOG

- The *Message Router Version 3.0 Release Notes* state that the verification procedures require DECnet and the Queue Manager be running. However, the verification procedures also require that at least one queue be defined in the system startup command procedure.

To do this, add the following command line to your system startup procedure:

```
$ INITIALIZE/QUEUE/BATCH queue name)
```

Refer to the *VMS DCL Dictionary* for more information about initializing batch queues.

4.23 Modular Executive — Notes

The following sections contain information on the Modular Executive.

4.23.1 New Description for $MTACCESS

V5.0

The $MTACCESS service allows sites to provide their own routine to interpret an output accessibility field in VOL1 and HDR1 labels of ANSI-labeled magnetic tapes. The site can override the default routine by providing an MTACCESS.EXE executive shareable image.

4.23.2 Instructions for Loading a Site-Specific Executive Loaded Image

V5.0

This section contains step-by-step instructions for preparing a site-specific executive loaded image, for loading this image into the operating system, and for removing the image. The example creates an MTACCESS.EXE executive loaded image. A similar example can be found in SYS$EXAMPLES:DOD_ERAPAT.MAR on the VMS operating system.

Preparing and Loading the Executive Loaded Image

1 Create the source module MTACCESS.MAR.

a. Include the following macro to define system service vector offsets:

```
$SYSVECTORDEF   ; Define system service vector offsets
```

b. Use the following macros to define the system service entry point:

```
SYSTEM_SERVICE MTACCESS, -      ; Entry point name
               <R2,R4>, -       ; Registers to save
               MODE=KERNEL,-    ; Mode of system service
               NARG=6           ; Number of arguments
```

The instruction following the preceding macros is the first instruction of the $MTACCESS system service.

c. Use the following macros to declare the desired program sections (PSECT):

```
DECLARE_PSECT   EXEC$PAGED_CODE ; Pageable code PSCET
DECLARE_PSECT   EXEC$PAGED_DATA ; Pageable data PSECT
DECLARE_PSECT   EXEC$NONPAGED_DATA   ; Nonpageable data PSEC
DECLARE_PSECT   EXEC$NONPAGED_CODE   ; Nonpageable code PSCE
```

2 Assemble the source module by using the following command:

```
$
```

3 Link the module to create an MTACCESS.EXE executive loaded image. You can link the module by using a command procedure as follows:

```
$ LINK /NOSYSSHR/NOTRACEBACK -
        /SHARE=MTACCESS -
        /MAP=MTACCESS /FULL /CROSS -
        /SYMBOL=MTACCESS -
        SYS$INPUT/OPTION
MTACCESS, -
SYS$LIBRARY:STARLET/INCLUDE:(SYS$DOINIT),-
SYS$SYSTEM:SYS.STB/SELECTIVE
VECTOR_TABLE=SYS$SYSTEM:SYS.STB
COLLECT=NONPAGED_READONLY_PSECTS/ATTRIBUTES=RESIDENT,-
        EXEC$NONPAGED_CODE
COLLECT=NONPAGED_READWRITE_PSECTS/ATTRIBUTES=RESIDENT,-
        EXEC$NONPAGED_DATA
COLLECT=PAGED_READONLY_PSECTS,-
        EXEC$PAGED_CODE
COLLECT=PAGED_READWRITE_PSECTS,-
        EXEC$PAGED_DATA
COLLECT=INITIALIZATION_PSECTS/ATTRIBUTES=INITIALIZATION_CODE,-
        EXEC$INIT_CODE,-
        EXEC$INIT_000,-
        EXEC$INIT_001,-
        EXEC$INIT_002,-
        EXEC$INIT_PFNTBL_000,-
        EXEC$INIT_PFNTBL_001,-
        EXEC$INIT_PFNTBL_002,-
        EXEC$INIT_SSTBL_000,-
        EXEC$INIT_SSTBL_001,-
        EXEC$INIT_SSTBL_002
```

4 Prepare the executive loaded image to be loaded.

a. Copy MTACCESS.EXE images produced by the preceding link command into the SYS$LOADABLE_IMAGES directory. Note that privilege is required to put files into this directory.

b. Add an entry for the MTACCESS.EXE image in the SYS$UPDATE:VMS$SYSTEM_IMAGES.IDX data file.

You add an entry by using the SYSMAN Utility. The SYSMAN command is as follows:

```
SYSMAN SYS_LOADABLE ADD _LOCAL_ image_name
    /LOAD_STEP = {INIT | SYSINIT} -
    /SEVERITY  = {WARNING | SUCCESS | FATAL | INFORMATION} -
    /MESSAGE   = "error message text"
```

The image name defines the file specification of the image to be loaded. The default directory is <SYS$LDR> and the default file type is EXE.

The /LOAD_STEP qualifier has the following two images:

INIT	Image to be loaded by the system initialization code.
SYSINIT	Image to be loaded by the SYSINIT process.

The /SEVERITY qualifier has the following parameters:

WARNING	If error loading the image, output the error message and continue processing.
SUCCESS	Continue even if there is an error loading the image. No message is issued.
FATAL	If error loading the image, output the error message and BUGCHECK.
INFORMATION	Always output the message and continue.

The /MESSAGE qualifier is a supplied error message text to be issued under the appropriate condition.

For example, you can add the following entry to VMS$SYSTEM_IMAGES.IDX for MTACCESS.EXE:

```
$ SYSMAN SYS_LOADABLE ADD _LOCAL_ MTACCESS -
_$ /LOAD_STOP = SYSINIT -
_$ /SEVERITY = WARNING -
_$ /MESSAGE = "failure to load installation-specific - $MTACCESS service"
```

This entry specifies that the MTACCESS.EXE image is to be loaded by the SYSINIT process during the bootstrap. If there is an error loading the image, the following messages are printed on the console terminal:

```
%SYSINIT-E-failure to load installation-specific $MTACCESS service
-SYSINIT-E-error loading <SYS$LDR>MTACCESS.EXE, status = "status"
```

c. Invoke the SYS$UPDATE:VMS$SYSTEM_IMAGES.COM command procedure to generate a new system image data file. The system bootstrap uses this image data file to load the appropriate images into the system.

d. Shut down and reboot the system, which loads the site-specific MTACCESS.EXE executive loaded image into the system. Subsequent calls to the $MTACCESS system service use the site-specific routine.

As the default, the system bootstrap loads all images described in the system image data file (VMS$SYSTEM_IMAGES.DATA). You can disable this function by setting the special SYSGEN parameter LOAD_SYS_IMAGES to 0.

Removing the Executive Loaded Image

You can remove an executive loaded image by using the following procedure:

1 Use the following SYSMAN command (based on the specific example in the preceding instructions).

```
$
```

2 Repeat steps c and d from instruction 4.

4.24 National Character Set (NCS) Notes

V5.2

The following are changes and enhancements to the National Character Set (NCS):

- An optional fifth parameter, **not-cvt**, was added to the NCS$CONVERT routine. That argument returns the number of characters in the input string that were not fully converted.
- NCS now fully supports all string descriptors for languages.

4.25 VAX Pascal Run-Time Library — Enhancements

V5.2

The VAX Pascal Run-Time Library has been enhanced in VMS V5.2. These enhancements will become active when used in conjuction with a future release of VAX Pascal, and will be described in detail in connection with that release.

Running a VAX Pascal application linked on VMS Version V5.2 on a system running a previous version of VMS is not supported.

4.26 VAX PL/I Run-Time Library Notes

V5.2

The following corrections and enhancements have been made to the VAX PL/I Run-Time Library (RTL):

- The COL and DCOL key types are now supported as datatypes for keys in indexed files. This allows PL/I programs to read indexed files containing keys with user-defined collating sequences. You can use the National Character Set utility to create these collating sequences.
- When converting from PICTURE to a FIXED or FLOAT datatype, the pictured value is first converted to an intermediate FIXED DECIMAL value with the same precision and scale as the PICTURE variable. This intermediate value is then converted to the desired FIXED or FLOAT datatype. Previously the picture value was converted to a FIXED DECIMAL (31,0) intermediate value, which resulted in the loss of all fractional digits.
- It is now possible to specify a boolean value with the record-locking options of the READ statement. The format for the options is NOLOCK(boolean), LOCK_ON_READ(boolean), LOCK_ON_WRITE(boolean), MANUAL_UNLOCKING(boolean), NONEXISTENT_RECORD(boolean), READ_REGARDLESS(boolean), and WAIT_FOR_RECORD(boolean), where boolean is a BIT(1) expression. Either a constant or a run-time variable expression is acceptable. The record-locking option is either enabled or disabled for the duration of the READ statement. For example:

```
DECLARE BOOL BIT(1) ALIGNED;
READ FILE(F) INTO(BUFF)
      OPTIONS(NOLOCK(BOOL), LOCK_ON_READ(^BOOL));
```

- The correct status RMS$_BUSY is now returned when you attempt to start a new I/O operation while another one is underway. Previously, the RMS status from an earlier I/O operation was returned rather than RMS$_BUSY.
- When a file is implicitly opened in response to an I/O request, the RTL now uses the declared file attributes rather than the default file attributes. This helps avoid conflicts between declared and default attributes. For example, the following READ statement now implicitly opens the file REL1 with the UPDATE attribute rather than the default attribute INPUT.

```
DCL REL1 FILE KEYED DIRECT UPDATE;
READ FILE(REL1) INTO(INREC) KEY(PART0);
```

- The ENCODE built-in function has been fixed to return the string "0" when the first parameter to ENCODE is 0. Previously the null string was returned.

4.27 Processor Register Definition Symbols

V5.0

The following internal processor registers (IPRs) are no longer common to all VAX processors. Their definitions have been removed from $PRDEF.

- NICR–Interval Clock Next Interval Register
- ICR–Interval Clock Interval Count Register
- TODR–Time of Day Register
- ACCS–Accelerator Control Status Register
- ACCR–Accelerator Reserved
- PME–Performance Monitor Enable

New CPU-specific processor register definition macros have been added to STARLET.MLB to define the CPU-specific IPRs. The macro names have the format $PRxxxDEF, where xxx is the number associated with the processor (for example, $PR780DEF will define PR780$_ ACCS).

The only legitimate references to these registers are in CPU-dependent code. These references must use the new CPU-dependent IPR definitions.

Note, however, that time-wait loops must never directly refer to the clocks. They must use a time-wait macro that is independent of the CPU. A new, CPU-independent, time-wait macro called TIMEDWAIT has been added to LIB.MLB. This should eliminate any need for hand-coded time-wait loops.

There should no longer be any references to PR$_ICR or PR$_TODR to do time-wait loops. TIMEDWAIT allows for up to six special-purpose instructions to be placed in its timing loop. However, the loop timing is based on having one BITx and one conditional branch instruction embedded within the loop. Therefore, if you have a loop with no embedded instructions, you may want to adjust the TIME argument accordingly. A good rule of thumb is to add 25 percent to the time argument if the loop has no embedded instructions.

To refer to PR$_TODR for logging purposes, use EXE$READ_TODR and EXE$WRITE_TODR. These two, new, loadable, CPU-dependent routines have been added for code that must reference this type of value.

4.28 Record Management Services (RMS) — Notes

The following sections pertain to the VMS Record Management Services (RMS).

4.28.1 Future Access Mode Changes to RMS

V5.2

In the next major release of VMS after Version 5.2, RMS will provide access mode protection. (See the *Introduction to VMS System Services* for a discussion of Access Modes). This protection will be enforced accross all relevant RMS System Services, and all memory owned by RMS will have its protection changed from the current setting of UREW (USER read; EXEC write), to EW (EXEC read/write).

4.28.2 RMS Statistics Restrictions

V5.1

The following restrictions apply to the use of RMS statistics:

- RMS statistics cannot be gathered on files residing on ODS-1 (On Disk Structure Level 1) disks.
- RMS statistics are not maintained for process-permanent file accesses. Process-permanent file accesses are those that are not released on image rundown. These are typically accesses resulting from the DCL OPEN command. If a file is accessed both as a process-permanent file and by a user image, then only operations done by the user image are counted in the RMS statistics. Enable or Disable the gathering of RMS statistics with the SET FILE/[NO]STATISTICS command.

4.28.3 RMS Support for Printing with Vertical Format Units (VFU)

V5.2

VMS RMS now supports printing devices that have Vertical Format Units (VFU). You implement VMS RMS support by selecting the FAB$V_PRN option in the FAB$B_RAT record attributes field and using appropriate bit configurations in the 2-byte fixed-length control fields that precede variable-length records. The bit configurations that encode VFU functions are:

```
Bit 7  Bit 6  Bit 5  Bit 4  Bit 3  Bit 2  Bit 1  Bit 0
  1      1      0      0      x      x      x      x
```

Bits 3 to 0 direct the device to skip to the appropriate channel (1 to 16) for positioning information. Devices that do not have hardware VFUs translate each of these codes as a one-line advance.

4.28.4 $TRUNCATE Service

V5.1

The RMS $TRUNCATE service is now sensitive to record-access mode.

In sequential record-access mode, you can use this service only immediately after setting the context of the current record by successfully executing a Get or a Find service.

In random-access-by-key mode, VMS RMS establishes the current record position as defined by the key of reference or by the relative record number, as applicable.

In random-access-by-RFA (Record File Address) mode, VMS RMS establishes the current record position as defined by the RFA.

4.28.5 XAB$V_NUL Option — Clarification

V5.0

The *VMS Record Management Services Manual* states that you can only use the XAB$V_NUL option with string-type keys. Actually, you can use this option with all key types. Note however, that RMS sets the null value to 0 for keys other than string-type keys.

4.29 Recovery Unit Journaling

The following sections contain information on recovery unit journaling.

4.29.1 Appending to Write-Shared Sequential Files

V5.2

If records are appended to a write-shared sequential file containing fixed-length records using recovery unit journaling, and the recovery unit is not committed (either SYS$ABORT_RU is called, or a system failure occurs), recovery will overwrite each appended record in the recovery unit with zeros. Subsequent readers of the file will read these zeroed records. This behavior is necessary, because other shared accessors may also have appended records to the file following the zeroed records, and those other record numbers cannot be changed. There is no support for deleted records in sequential files.

4.29.2 Key and Index Compression — Problems Fixed

V5.2

VMS Version 5.2 corrects problems that could occur in previous versions of VMS if recovery unit journaling was being used on RMS indexed files that had key compression and/or data record compression enabled. Note that key and data record compression are enabled by default for prolog 3 indexed files.

If you are using RMS Journaling on indexed files with key and/or data record compression enabled, you are encouraged to upgrade to VMS Version 5.2.

4.29.3 Moving Recovery-Unit Journaled RMS Indexed Files to Systems Running VMS Version 4.7 and Earlier

V5.2

There is a restriction on moving indexed files that have been marked for recovery unit journaling, modified within a recovery unit, and then unmarked for recovery unit journaling, to systems running VMS Version 4.7 and earlier where RMS Journaling Version 1.0 is not installed.

You must first make a new copy of the file using the Convert Utility. You can then transfer the converted copy of the file to the system running VMS Version 4.7 or earlier.

4.30 Reinstalling Languages — Requirement

V5.2

To most easily use a VMS system routine from a given high level language, language-specific definitions need to exist for the following:

- The routine
- The routine's error messages
- Routine-specific constants

For most languages, these definitions are built when the language is installed from files that exist in the VMS system. If the language is installed before a system routine exists, the language-specific definitions for that system routine will not be present.

The routines PROCESS_SCAN, DEVICE_SCAN, and the Clusterwide Process Services (CWPS) extensions did not exist prior to VMS Version 5.2. The libraries for a language that was installed prior to VMS Version 5.2 do not contain the definitions that should be used when using the new system routines from that language. A programmer attempting to use these features as documented in *VMS Version 5.2 New Features Manual* will be unsuccessful.

To build the language-specific definitions required for using PROCESS_SCAN, DEVICE_SCAN, and the CWPS extensions, reinstall each language for which you wish to use these routines.

4.31 Run-Time Library (RTL) — Notes

The following sections contain information concerning the Run-Time Library.

4.31.1 RTL Language Support Enhancements

V5.0

The following enhancements have been made to the Run-Time Library for language support:

- COBOL Support for ANSI 85 — File Status variable returns ANSI 85 values and new error messages to go with ANSI 85 file status codes.
- FORTRAN descending key support — Key direction specified in open statement.

4.31.2 RTL Library (LIB$)

The following sections pertain to the RTL Library.

4.31.2.1 LIB$GET_VM Routine Performance Degradation

V5.0-1

In Version 5.0 of the VMS operating system, performance degradation in the LIB$GET_VM routine occurred under the following conditions:

- The program created a zone, defaulting all the parameters.
- The program allocated many small pieces of memory that totaled a large portion of memory.
- The program made few calls to LIB$FREE_VM.

Version 5.0-1 of the VMS operating system fixes this performance degradation problem.

4.31.2.2 LIB$VERIFY_VM_ZONE and LIB$SHOW_VM_ZONE Zone Analysis Problem

V5.1

The routines LIB$VERIFY_VM_ZONE and LIB$SHOW_VM_ZONE can, under specific conditions, incorrectly determine that a virtual memory zone is corrupted.

If a program causes a zone to have one or more 8-byte blocks of free memory, the routine LIB$VERIFY_VM_ZONE incorrectly returns the status LIB$_BADZONE. In the same situation, LIB$SHOW_VM_ZONE reports that the area free list is corrupted with an invalid block size.

A sample of the incorrect output from LIB$SHOW_VM_ZONE is shown in Example 4–1. Note that it is the zone analysis that is incorrect; the memory zone itself is not corrupted.

Example 4–1 Sample Output of Routine LIB$SHOW_VM_ZONE

```
Zone ID = 00073600,  Zone name = ""

Algorithm = LIB$K_VM_FIRST_FIT

Flags = 00000000

Initial size =    16 pages     Current size = 16 pages in 1 area
Extend size  =    16 pages     Page limit   = None

Requests are rounded up to a multiple of 8 bytes,
naturally aligned on 8 byte boundaries

8 bytes have been freed and not yet reallocated

144 bytes are used for zone and area control blocks, or 1.7% overhead

Area Summary:

      First     Last       Pages    Bytes not yet
     address   address    assigned    allocated
    --------  --------  ----------  -------------
    00073E00  00075DFF          16           8184

Scanning Free List for Area at 00073E00
```

(continued on next page)

Example 4–1 (Cont.) Sample Output of Routine LIB$SHOW_VM_ZONE

```
**** ERROR -- invalid block size ****

Link Analysis for Current Block:

                          Previous   Current      Next
                          --------  --------  --------
         Block adr      :  00062EF0  00073E00  00062EF0
         Forw link (abs):  00073E00  00062EF0  00073E00

Block size =     8192

Block contents:

       (1) 00000000 00000000 00062EF0 00000008 ....?........... 00000 00073E00
           00000000 00000000 00000000 00000000 ................ 00010 00073E10
             (510 matching lines skipped)
```

(1) The key to recognizing that this is the known problem is the value 00000008 in the first longword of the block dump.

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

4.31.3 MTH$DCOSD, MTH$DSIND, and MTH$DTAND Functions

V5.0-1

The VMS Version 5.0 RTL math routines MTH$DCOSD, MTH$DSIND, and MTH$DTAND erroneously returned 0 when the return value was close to 0 but could still be represented as a D_float number.

Version 5.0-1 of the VMS operating system fixes this problem by returning the actual D_floating value when these routines are called.

4.31.4 RTL Parallel Processing Facility (PPL$)

The following sections describe changes and enhancements to the RTL Parallel Processing (PPL$) Facility:

4.31.4.1 PPL$AWAIT_EVENT — New Argument

V5.2

A new optional second argument named **output** is added to the routine PPL$AWAIT_EVENT. The **output** argument receives the **event-param** argument from the PPL$TRIGGER_EVENT routine. The value of the **event-param** argument is copied to the **output** argument when an event is triggered.

4.31.4.2 PPL$ENABLE_EVENT_SIGNAL — Restriction

V5.0

PPL$ENABLE_EVENT_SIGNAL provides for cross-process asynchronous signaling. This is a powerful mechanism, and it must be used only in carefully controlled environments.

Asynchronous exceptions are those which are not a direct result of the execution of the code, but rather are caused by some concurrent and not directly related event. For example, an AST interrupts a MOVC instruction and the AST routine attempts to reference an invalid address,

resulting in an access violation. The signaled exception is an ACCVIO, and it is not related to the interrupted MOVC instruction. Occurrences of asynchronous exceptions have previously been quite uncommon, and the majority of existing code expects to terminate upon receipt of such an exception. The PPL$ENABLE_EVENT_SIGNAL service introduces the means for use of asynchronous signals as a communications mechanism.

Delivery of an asynchronous signal to an arbitrary layered environment can result in unwinding code which is totally unprepared for it, resulting in corrupted data. For example, any RTL routine or the code of a layered product might be interrupted by such an exception. Code that executes in multiple threads under one process context is particularly vulnerable — for example, Ada tasking. Delivery of an asynchronous exception will interrupt the task that happens to be executing at the time, and will result in task termination. Do not use this routine in environments that support multi-tasking within a process.

To avoid the potential program data corruptions and unintended alterations of control flow implied by unexpected unwinding of an unprepared code section, use this asynchronous signaling capability only when the code that can be interrupted is your own. Also note that you can accomplish the same tasks in a less dangerous fashion — using the standard AST facilities — by use of the PPL$ENABLE_EVENT_AST routine.

4.31.4.3 PPL$FLUSH_SHARED_MEMORY — Flushes All Modifications

V5.2

The routine PPL$FLUSH_SHARED_MEMORY now flushes all modifications to shared memory. Previously, if more than one process modified shared memory, only the modifications of the process that called PPL$FLUSH_SHARED_MEMORY were flushed.

4.31.4.4 PPL$INITIALIZE — New Condition Value PPL$_ALTSIZE

V5.2

A new condition value named PPL$_ALTSIZE has been added to the routine PPL$INITIALIZE. When PPL$_ALTSIZE is returned, it signifies that the size of the PPL$ global section has been established and cannot be changed.

4.31.4.5 PPL$SPAWN — Creating Multiple Copies of a Program

V5.0

If you specify a value greater than 1 for the **copies** argument to PPL$SPAWN, each copy created will have the same subprocess information (for example, standard input and output files). If you want to specify different information for each subprocess, call PPL$SPAWN once for each subprocess.

4.31.5 RTL Screen Management (SMG$) Routines

The following notes cover corrections made in VMS Version 5.2 for problems in SMG$ routines.

4.31.5.1 Terminal Page Size

V5.2 Previous versions of SMG$ forced the terminal page size to 24 lines if you set it to 0 lines. This has now been fixed.

4.31.5.2 SMG$ Performance

V5.2 The performance of SMG$ has been improved by replacing calls to the $FAO system service with faster inline code.

4.31.5.3 SMG$ Routines — Image Exit

V5.2 In previous versions of the Run-Time Library (RTL) Screen Management (SMG$) facility, an access violation sometimes occurred at image exit. The SMG$ exit handlers did not correctly mark the pasteboard as deleted, allowing user exit handlers to access unallocated virtual memory by calling SMG$ routines.

This problem has now been corrected. If you call SMG$ routines from your exit handler, you may find that these calls now fail with the error SMG$_INVPAS_ID. To correct this, call SMG$CREATE_PASTEBOARD and/or SMG$CREATE_VIRTUAL_KEYBOARD before declaring your exit handler.

4.31.5.4 SMG$ADD_KEY_DEF

V5.2 Previous versions of the routines SMG$ADD_KEY_DEF and SMG$READ_COMPOSED_LINE only allowed you to define function, keypad, editing keypad, and control keys. In VMS Version 5.2 you can define all keyboard keys.

4.31.5.5 SMG$CHANGE_RENDITION

V5.2 The routine SMG$CHANGE_RENDITION did not always set the rendition correctly when user-defined attributes were specified for the **rendition-set** argument. This has now been fixed.

4.31.5.6 SMG$CREATE_MENU

V5.2 Previous versions of the routine SMG$CREATE_MENU only supported the fixed size string arrays generated by VAX FORTRAN for the CHOICES parameter. In VMS Version 5.2, this routine now supports string arrays generated by all VAX language compilers except for VAX BASIC dynamic string arrays. When calling this routine from VAX BASIC, you must continue to use a MAP statement to declare the string array passed as the CHOICES parameter.

4.31.5.7 SMG$CREATE_PASTEBOARD and SMG$CREATE_VIRTUAL_KEYBOARD — Restriction

V5.0 If a program calls both SMG$CREATE_PASTEBOARD and SMG$CREATE_VIRTUAL_KEYBOARD, make sure that SMG$CREATE_PASTEBOARD is called first. Otherwise, the program will not function correctly.

4.31.5.8 SMG$DELETE_PASTEBOARD

V5.2 The routine SMG$DELETE_PASTEBOARD no longer generates an access violation when called from an exit handler.

4.31.5.9 SMG$M_REVERSE

V5.2 In VMS versions previous to 5.0, specifying the routine SMG$M_REVERSE in a RENDITION_SET parameter to any of the SMG$ routines when the background color was set to SMG$C_COLOR_WHITE reversed the rendition. In VMS Versions 5.0 and 5.1, this had no effect. In VMS Version 5.2 this problem has been fixed.

4.31.5.10 SMG$READ_xxxx

V5.2 All SMG$READ_xxxx routines now protect themselves against multiple simultaneous use on the same virtual keyboard. The routines now return the status SMG$_KBDIN_USE instead of generating an access violation.

4.31.5.11 SMG$READ_VERIFY

V5.2 The routine SMG$READ_VERIFY now correctly updates the virtual cursor location in the virtual display. Previous versions always forced the virtual cursor to the next line in the display.

4.31.5.12 SMG$SNAPSHOT

V5.2 Previous versions of the routine SMG$SNAPSHOT did not trim blank characters from records output when the pasteboard was an RMS file. This has now been fixed.

4.32 Self-Modifying Item Lists with $GETxxx Services

V5.2 A problem can occur if you use self-modifying item lists with the following services:

- $GETDVI
- $GETDVIW
- $GETJPI
- $GETJPIW
- $GETLKI
- $GETLKIW
- $GETMSG
- $GETQUI
- $GETQUIW
- $GETSYI
- $GETSYIW
- $GETTIM
- $GETUAI

When any one of these services collects data, it makes multiple passes through the item list. The number of passes needed depends both on which item codes are referenced and the state of the target process. If the item list is self modifying—that is, if the addresses for the output buffers in the item list point back to the item list—the service replaces the item-list information with the collected data. Therefore, incorrect data may be returned or unexpected errors may occur when the service reads the item list again.

A program using self-modifying item lists that appears to work normally can fail when a system has processes that are swapped out of memory, or when a process is on a remote node. System load or the order of the item list entries can also cause such a program to fail.

To prevent confusing errors, Digital recommends that you not use self-modifying item lists.

4.33 SET HOST/DTE/DIAL Command — Problem and Solution

V5.0

The SET HOST/DTE/DIAL command does not work with the DMF-32 controller because the modem sends a response character to the host when it detects a carrier signal. The DMF-32 controller drops any input until it sees the carrier signal.

One solution is to modify the example autodialer provided in SYS$EXAMPLES:DT_DF03.MAR to perform an IO$_SENSEMODE!IO$M_RD_MODEM $QIO to check for a carrier signal. If set, the autodialer should assume success and continue.

5 Documentation Release Notes

This chapter contains additions and corrections to the VMS documentation set.

5.1 General Notes

The following sections contain documentation information that does not pertain to any specific book.

5.1.1 PostScript Previewer

V5.1

The VMS Version 5.1 documentation includes references to a DECwindows desktop application called the PostScript Previewer. Due to last-minute technical difficulties, Digital has not included this application in the VMS Version 5.1 release. Please disregard any documentation references to the PostScript Previewer.

5.1.2 Undocumented New VAX Computers

V5.0

The VMS Version 5.0 documentation does not reflect several changes and additions to the VAX computer nomenclature that were made after the documentation entered production. Among the new computers supported by VMS Version 5.0 are the VAX 8810 and VAX 8820-N. In the Version 5.0 documentation, references to the VAX 8700 can apply equally to the VAX 8810; references to the VAX 8800 can apply to the VAX 8820-N. (The VAX 8670 system, mentioned in the *VMS Device Support Manual*, does not exist.)

A VAX 8810 consists of a single processor. The VAX 8820-N is a tightly coupled multiprocessing system comprising two CPUs. Installation media for the VAX 8810 and VAX 8820-N systems include the RX50 floppy disk (for Standalone BACKUP) and magnetic tape. Accordingly, managers of these systems should refer to the manual *VMS Installation and Operations: VAX 8530, 8550, 8700, 8800* for instructions on installing the VMS operating system and information on processor operations.

5.2 *VMS Compound Document Architecture Manual*

V5.1

Chapter 5 of the *VMS Compound Document Architecture Manual* describes the guidelines for creating user-written front and back ends that work with the Compound Document Architecture (CDA) Converter to translate various file formats to and from a CDA in-memory representation. Both the DDIF$READ_format routine (for a front end) and the DDIF$WRITE_format routine (for a back end) support an argument called **standard-item-list**. This argument identifies the document source and can also contain processing options.

The documentation does not mention that Digital recommends that front and back ends, when they encounter unrecognizable items in the **standard-item-list** parameter, should not return an error. Instead, the unrecognized item should be ignored and processing should continue.

The documentation of the PostScript back end in Chapter 2 of the *VMS Compound Document Architecture Manual* does not mention that, in a PostScript processing option file, the processing option keyword can be preceded by a slash (/). For example:

```
PS/PAPER_ORIENTATION LANDSCAPE
```

5.3 *VMS Convert and Convert/Reclaim Utility Manual*

V5.0

The *VMS Convert and Convert/Reclaim Utility Manual* displays the following erroneous format for the Convert Utility routine CONV$PASS_FILES:

```
CONV$PASS_FILES input-filespec,output-filespec[,fdl-filespec]
                [,exception-filespec][,flags]
```

The correct format is as follows:

```
CONV$PASS_FILES input-filespec,output-filespec,[fdl-filespec]
                ,[exception-filespec],[flags]
```

Note that in the corrected version the comma delimiters for the optional arguments are shown external to the brackets. Note also that comma delimiters are to be eliminated for trailing optional arguments.

5.4 *VMS DCL Dictionary*

The following sections describe changes to the *VMS DCL Dictionary*.

5.4.1 CALL Command

V5.2

On page DCL-51, the description of the CALL command states:

> Local symbols defined in an outer subroutine level are available to any subroutine levels at an inner nesting level.

To prevent any confusion about accessing symbols within subroutines, the description should read as follows:

> Local symbols defined in an outer subroutine level can be read at any inner subroutine level, but they cannot be written to. If you assign a value to a symbol that is local to an outer subroutine level, a new symbol is created at the current subroutine level. However, the symbol in the outer procedure level is not modified.

5.4.2 CONNECT and DISCONNECT Commands

V5.2

On page DCL-58, the description of the CONNECT command states:

> When the connection between the physical terminal and the virtual terminal is broken, the process remains connected to the virtual terminal. If the process is executing an image, it continues until it needs terminal input or attempts to write to the terminal. At that point, it waits.

In fact, when the connection between the physical terminal and the virtual terminal is broken, you are logged out of your current process (and any images that the process is executing stop running) unless you have specifed the /NOLOGOUT qualifier.

If you have specifed the /NOLOGOUT qualifier, the process remains connected to the virtual terminal. If the process is executing an image, it continues until the process needs terminal input or attempts to write to the terminal. At that point, the process waits until the physical terminal is reconnected to the virtual terminal.

On page DCL-145, the description of the /CONTINUE qualifier of the DISCONNECT command states:

> Controls whether the CONTINUE command is executed in the current process just before connecting to another process. This permits an interrupted image to continue processing after the disconnect takes place.

In fact, the DISCONNECT/CONTINUE command permits an interrupted image to continue processing after the disconnect *only* until the process needs terminal input or attempts to write to the terminal. At that point, the process *waits* until the physical terminal is reconnected to the virtual terminal.

5.4.3 DIFFERENCES Command

V5.2

On page DCL-129, add the following to the description of the /CHANGE_BAR qualifier:

> If you use an exclamation point as the specified character, you must enclose it in quotation marks, for example, /CHANGE_BAR=("!",NUMBER).

On page DCL-130, "/COMMAND_DELIMITER" should read "/COMMENT_DELIMITER".

5.4.4 DIRECTORY Command

V5.2

On page DCL-139, add the following to the list of information that the /FULL qualifier displays for each file:

> Value of the stored semantics tag (where applicable)

5.4.5 F$ELEMENT Lexical Function

V5.2

On page DCL-238, in Example 1 replace the label "ERROR" with "END".

On page DCL-239, in the last line of Example 2 replace "CHAPTERS" with "NUM". Add the following to the explanation of Example 2:

> NEXT is initialized to zero. The procedure enters the *loop*. In the first iteration, NEXT is incremented to 1 and the result of the F$ELEMENT call is the string "1". The procedure runs the index for Chapter 1. In the second iteration, NEXT is incremented to 2 and the result of the F$ELEMENT call is the string "2". The procedure runs the index for Chapter 2. Processing continues until the result of the F$ELEMENT call is the delimiter specified in the call.

5.4.6 F$ENVIRONMENT Lexical Function

V5.2

On page DCL-241, the table states the following about the SYMBOL_ SCOPE item:

> The string is in a form that can be used with the SET PROTECTION/DEFAULT command to form a valid DCL command line.

In fact, this statement applies to the PROTECTION item and should be added to the information returned for that item.

5.4.7 F$GETJPI Lexical Function

V5.2

On page DCL-263, make the following changes to Table DCL-9:

- The item AUTHPR should be AUTHPRI.
- Add the following item:

```
CREPRC_FLAGS    Integer  Flags specified by the stsflg argument
                         in the $CREPRC call that created the process
```

- Add the following sentence to the description of the FREPOVA item:

  > Irrelevant if no image is running.

On page DCL-264, add the following items to Table DCL-9:

```
PRI             Integer   Process's current priority
UAF_FLAGS       Integer   User Authorization File (UAF) flags from the
                          UAF record of the user who owns the process
```

5.4.8 F$GETQUI Lexical Function

V5.2

On page DCL-266, add the following to the description of the return value:

> If the $GETQUI system service returns an error code, F$GETQUI returns a null string (" ").

On page DCL-268, add the following to the description of the keyword FREEZE_CONTEXT:

> If you do not specify this flag, the context is advanced to the next object.

On page DCL-269, add the following after the first sentence in the description:

> For example, in nested wildcard operations, $GETQUI returns information about objects defined within another object. Specifically, this mode allows you to query jobs contained in a selected queue or files contained in a selected job in a sequence of calls. After each call, the system saves the internal GETQUI context block (GQC) so that the GQC can provide the queue or job context necessary for subsequent calls. See the description of the $GETQUI system service in the *VMS System Services Reference Manual* for more information.

On page DCL-278, Example 2 reads:

```
$IF F$GETQUI("DISPLAY_QUEUE","QUEUE_STOPPED","VAX1_BATCH")
      THEN GOTO 500
```

The example should read:

```
$IF F$GETQUI("DISPLAY_QUEUE","QUEUE_STOPPED","VAX1_BATCH")
      .EQS. "TRUE" THEN GOTO 500
```

Add the following to the explanation of Example 2:

> If VAX1_BATCH is not in the system, F$GETQUI returns a null string (" ").

On pages DCL-278 and DCL-279, the following example replaces the last example found in the F$GETQUI lexical function section:

```
$  TEMP = F$GETQUI("CANCEL_OPERATION")
$ LOOP1:
$  QNAME = F$GETQUI("DISPLAY_QUEUE","QUEUE_NAME","*","BATCH")
$  IF QNAME .EQS. "" THEN EXIT
$  WRITE SYS$OUTPUT "Jobs in batch queue ", QNAME, " are:"
$ LOOP2:
$  JNAME = F$GETQUI("DISPLAY_JOB","JOB_NAME",,"ALL_JOBS")
$  IF JNAME .EQS. "" THEN GOTO LOOP1
$  WRITE SYS$OUTPUT "    ", JNAME
$ GOTO LOOP2
```

This sample command procedure searches through batch queues and displays all jobs currently residing in each queue. Because a wildcard queue name is specified ("*"), wildcard queue context is maintained across calls to F$GETQUI. This context is dissolved when the list of matching queues is exhausted. Furthermore, F$GETQUI returns a null string ("") to denote that no more objects match the specified search criteria. Finally, an initial cancel operation is performed to dissolve any wildcard context for the process that may still exist from a previously aborted search sequence (for example, abort of a SHOW QUEUE command or the running of this command procedure).

5.4.9 F$GETSYI Lexical Function

V5.2

On page DCL-282, Table DCL-12 lists the following items codes as valid for the local nodes or for other nodes in the VAXcluster:

- ACTIVECPU_CNT
- AVAILCPU_CNT

In fact, these item codes are valid only for the local node, and should be added to Table DCL-11.

5.4.10 LINK Command

V5.2

On page DCL-318, the description of the /DEBUG qualifier states: "By default, VAX Symbolic Debugger is linked with the image." This implies that the default for this qualifier is /DEBUG. In fact, the default is /NODEBUG.

To prevent any confusion about the default, the description of the /DEBUG qualifier should read as follows: "If a debugger is linked (that is, if you specify the /DEBUG qualifier), by default the VAX Symbolic Debugger is linked with the image."

5.4.11 PRINT Command

V5.2

On page DCL-358, the description of the /USER qualifier states that this qualifier requires CMKRNL privilege and R (read) access to the user authorization file (UAF). In fact, the /USER qualifier requires OPER privilege, E (execute) access to the queue, or W (write) access to the queue.

5.4.12 REPLY Command

V5.2

On page DCL-376, insert the following after the second sentence:

> However, if the file system requests a new volume, the operator can reuse a scratch volume by mounting it and entering the REPLY/INITIALIZE_TAPE command. The operator also can mount a blank volume and enter the REPLY/BLANK_TAPE command. In any case, the operator can add the message "label" to the REPLY command to specify the volume's label. The double quotation marks are required syntax.

On page DCL-379, add the following paragraph to the description of the /INITIALIZE_TAPE qualifier:

> If the tape drive cannot read the volume, the mount fails and an error message is returned. Use the /BLANK_TAPE qualifier to override the checking of information on a volume label.

5.4.13 SEARCH Command

V5.2

On page DCL-418, add the following description of the /HIGHLIGHT qualifier:

Controls whether the actual strings that are matched are emphasized in the output. The emphasis, or highlighting, can be one of several options:

BLINK	The matched strings are highlighted using the ANSI blink character attribute (advanced video only).
BOLD	The matched strings are highlighted using the ANSI bold character attribute (advanced video only). If /HIGHLIGHT is used without an option, BOLD is assumed.
REVERSE	The matched strings are highlighted with the ANSI underline video attribute (possible without advanced video).
UNDERLINE	The matched strings are highlighted with the ANSI underline video attribute (possible without advanced video). Without the advanced video option, either REVERSE or UNDERLINE will appear depending on whether the cursor is selected as block or underline. The two options REVERSE and UNDERLINE have the same effect.
HARDCOPY(=option)	This specifies that the strings should be highlighted in a manner suitable for most hardcopy printers. Hardcopy highlighting has two options: OVERSTRIKE and UNDERLINE. With overstrike highlighting, matched strings are double-printed, so that they appear darker. The matched strings are underlined with the underscore character. Hardcopy printing is accomplished by adding a carriage return and spacing back over the line to overprint the string or underlines. Note that this can as much as double the length of the line, and perhaps lead to truncation if the device buffer size is too small. Digital recommends that you use /HIGHLIGHT=UNDERLINE with the Digital LN01 printer instead of /HIGHLIGHT=HARDCOPY=UNDERLINE. The LN01 ignores OVERSTRIKE highlighting. Digital recommends that you use either /HIGHLIGHT=BOLD or /HIGHLIGHT=UNDERLINE with the Digital LN03 printer instead of /HIGHLIGHT=HARDCOPY=UNDERLINE. The LN03 ignores OVERSTRIKE highlighting.

Note that /HIGHLIGHT=BOLD is the default on ANSI video terminals with advanced video; /HIGHLIGHT=REVERSE is the default on ANSI video terminals without advanced video; and /NOHIGHLIGHT is the default for all other output.

5.4.14 SET CONTROL Command

V5.2

On page DCL-444, the description of the SET CONTROL command reads:

> When SET NOCONTROL=Y is in effect, the INTERRUPT message is displayed, but no interruption takes place.

The description should read as follows:

> When you press CTRL/Y and SET NOCONTROL=Y is in effect, the INTERRUPT message is displayed, but no interruption takes place. Note that DCL maintains an outstanding CTRL/Y AST to the terminal driver. This affects captive command procedures when using the SET HOST command. For more information, see the description of the SET HOST command in the *VMS DCL Dictionary*.

5.4.15 SET ENTRY Command

V5.2

On page DCL-456, the description of this command states that SET ENTRY accepts a single entry number. In fact, the command accepts a list of entry numbers.

5.4.16 SET HOST/DTE Command

V5.2

On page DCL-472, the description of the SET HOST/DTE command states:

> When connecting directly to another system, the out-going port should be set to NOTYPEAHEAD. This avoids the possibility of an endless loop, wherein noise on the line causes each of the ports to attempt to initiate a login. (Note that the terminal to which you have connected by way of SET HOST/DTE must be set to TYPEAHEAD to allow the login.)

To prevent any confusion, the description should read as follows:

> When two systems are connected permanently, before you start up the system, set the out-going port to NOTYPE_AHEAD. Setting the out-going port to NOTYPE_AHEAD avoids the possibility of an endless loop, wherein noise on the line causes each port to attempt to initiate a login. If you use a line set up in this way, when you enter the ALLOCATE command, set the terminal line to which you will connect by way of SET HOST/DTE to TYPE_AHEAD to allow the login.

5.4.17 SET RESTART_VALUE Command

V5.2

On page DCL-512, it is not obvious from the descriptions of the symbols $RESTART and BATCH$RESTART that these symbols are maintained quite differently. BATCH$RESTART is a normal global symbol, while $RESTART is a special type of symbol that cannot be deleted.

5.4.18 SET SYMBOL Command

V5.2

On page DCL-520, the description of the SET SYMBOL command states:

> For example, if SET SYMBOL/SCOPE=NOLOCAL was specified at procedure levels 2 and 4, procedure level 2 can access only level 2 local symbols. Level 3 can access levels 2 and 3 local symbols and level 4 can access only level 4 local symbols.

To prevent any confusion about accessing symbols within subroutines, the description should read as follows:

> For example, if SET SYMBOL/SCOPE=NOLOCAL was specified at procedure levels 2 and 4, procedure level 2 can read and write to only level 2 local symbols. Level 3 can read (but not write to) level 2 local symbols and can read and write to level 3 local symbols. Level 4 can read and write to only level 4 local symbols.
>
> Local symbols defined in an outer subroutine level can be read at any inner subroutine level, but they cannot be written to. If you assign a value to a symbol that is local to an outer subroutine level, a new symbol is created at the current subroutine level. However, the symbol in the outer procedure level is not modified.

5.4.19 SET TERMINAL Command

V5.2

On page DCL-525, the description of the /ALTYPEAHD qualifier states:

> Sets the size of the type-ahead buffer when used with the /PERMANENT qualifier.

This statement is inaccurate. The description of the /ALTYPEAHD qualifier should read as follows:

> Causes the terminal driver to create a permanent, alternate type-ahead buffer. The SYSGEN parameter TTY_ALTYPAHD determines the size of the type-ahead buffer. This specification is effective at your next login and stays in effect until you reboot your VAX computer.

5.4.20 SHOW CPU Command

V5.2

On page DCL-554, add the following to the description of the /FULL qualifier:

> The SHOW CPU/FULL command lists the current process on each configured processor without stopping other activity on the system. The current process may change while the data are displayed. As a result, there might be apparent inconsistencies in the display. For example, a process might be listed as the current process on more than one CPU.

5.4.21 SHOW PROCESS Command

V5.2

On page DCL-597, the description of Example 1 states that the SHOW PROCESS command display includes the default device and directory. In fact, the SHOW PROCESS command displays the default device only for processes on the same node and the default directory only for the current process.

5.4.22 SHOW SYSTEM Command

V5.2

On page DCL-614, add the following to the description:

> The SHOW SYSTEM command examines the processes on the system without stopping activity on the system. This means that process information can change during the time that SHOW SYSTEM collects the data to be displayed, resulting in minor inconsistencies in the SHOW SYSTEM display. For example, SHOW SYSTEM might display two processes scheduled in the state CUR on the same CPU.

5.4.23 SUBMIT command

V5.2

On page DCL-657, the description of the SUBMIT command reads:

> Once a batch job has been queued, the version of the file submitted is processed, even if a newer version of the file is created before the batch job runs.

Add the following paragraph to the description:

> In addition, even if you substitute another file with the same name and version number as the file queued, the original version of the file submitted is processed.

On page DCL-658, the description of the /AFTER qualifier for this command refers to the command SET TIME/CLUSTER. The SYSMAN command CONFIGURATION SET TIME has superseded the command SET TIME/CLUSTER.

On page DCL-663, the description of the /USER qualifier states that this qualifier requires CMKRNL privilege and R (read) access to the user authorization file (UAF). In fact, the /USER qualifier requires OPER privilege, E (execute) access to the queue, or W (write) access to the queue.

5.5 *Guide to DECnet-VAX Networking*

V5.2

On page 2-26 near the end of Example 2-1 of the *Guide to DECnet-VAX Networking*, the field **ast_ret_status** replaces the field **return_status** the ast_routine, on the lines beginning with switch (ast_param.type) and ending with exit (return_status). The replacement code is as follows:

```
{
    switch ( ast_param.type ) {
        case NET_RD  : ast_ret_status = insque_buffer ( LIVE_QUE,
                                        lct[ast_param.ndx].cur_buff );   break;
        case NET_WRT : ast_ret_status = insque_buffer ( LIVE_QUE,
                                        lct[ast_param.ndx].cur_buff );   break;
        case NET_CMD : ast_ret_status = que_and_reissue ();              break;
        default      : ast_ret_status = SS$_BADPARAM;                    break;
    }

    if (ast_ret_status & STS$M_SUCCESS)  {
        ast_ret_status = sys$wake ( 0, 0 );
        if ( ast_ret_status != SS$_NORMAL )
            exit ( ast_ret_status );
    }
    else
        exit ( ast_ret_status );
}
```

5.6 *VMS DECnet Test Sender/DECnet Test Receiver Utility Manual*

V5.0

On page DTS-8, the following qualifiers to the DATA command are listed. These qualifiers are no longer supported:

- FLOW=flow-control
- NOFLOW
- RQUEUE=number
- SQUEUE=number
- NAK=number
- NONAK
- BACK=number
- NOBACK

On page DTS-9, the example contains the unsupported qualifier /FLOW=MESSAGE. The correct command line in the example should be as follows:

```
_TEST:   DATA/PRINT/TYPE=SEQ/SIZE=128/SECONDS=10
```

On page DTS-13, the following qualifiers to the INTERRUPT command are listed. These qualifiers are no longer supported:

- RQUEUE=number
- SQUEUE=number

5.7 *VMS DECwindows Device Driver Manual*

V5.2

On page 1-4 of the *VMS DECwindows Device Driver Manual*, add the following to Table 1-1:

- YEDRIVER — under "System Type", add:

```
VAXstation 3100, 3520, and 3540.
```

- GABDRIVER — under "Workstation Type", add:

```
VAXstation 3100/GPX.
```

- GCBDRIVER — under "System Type", add:

```
VAXstation 3100 (monochrome).
```

- In the Output Driver section, add the new line:

```
GBBDRIVER     M-bus CPU and LEGSS video controller    GBA0    VAXstation 3520,3540.
```

On page 1-4, add the following paragraph:

> GBBDriver is an output driver for the VAXstation 3520 and 3540 Low End Subsystem (LEGSS) color monitors. The existing output drivers GABDriver and GCBDriver also support the VAXstation 3100.

On page 6-9, the $QIO GPB Wait function is now renamed "Packet Wait", which now supports two function modifier options.

In Table 6-4, expand the **p1** "Required Data" description to read as follows:

> IO$K_DECW_GPBWAIT function modifier for GPX packet buffers, or GB$K_LEGSS_WAIT_FOR_PKT for LEGSS packets.

5.8 *VMS DECwindows Toolkit Routines Reference Manual*

The following sections pertain to the *VMS DECwindows Toolkit Routines Reference Manual*.

5.8.1 Additional Attribute resizable

V5.1

The low-level routines ATTACHED DIALOG BOX CREATE and ATTACHED DIALOG BOX POPUP CREATE have an additional constraint attribute, **resizable**. In the VAX binding the attribute name is DWT$C_NRESIZABLE; in the C binding the attribute name is DwtNresizable. This is a Boolean attribute that, if true, allows the attached dialog box to change the size of its child widgets. The default is true.

5.8.2 Toolkit Routine Corrections

V5.2

On page 2-45 of the *VMS DECwindows Toolkit Routines Reference Manual*, the **display** argument in the intrinsic routine APPLICATION CREATE SHELL should be documented as a pointer in the VAX binding and a **display in the C binding.

On page 2-93, the return for the VAX binding for the intrinsic routine DISPLAY INITIALIZE is incorrectly listed as *result*. The return is void for the VAX binding, just as for the C binding. For the VAX binding the **display_name** argument should list usage and data type as unsigned longword and mechanism as reference.

On page 2-93, the VAX format for the intrinsic routine DISPLAY INITIALIZE (XT$Display_Initialize) incorrectly lists the argument **display_name** as a character string passed by descriptor. The **display_name** argument is actually a display structure pointer passed by value.

On page 4-25, the VAX format for the DRM routine REGISTER CLASS (DWT$Register_Class) lists the argument **class code** as passed by reference. The routine currently requires the argument **class code** to be passed by value.

On pages 5-3, 5-4, and 5-14, in the compound string routines ADD FONT LIST (page 5-3), CREATE FONT LIST (page 5-4), and GET NEXT SEGMENT (page 5-14), the following text should be added to the description of the argument **charset**:

> Values for this argument can be found in the required file CDA$DEF with a file type of the appropriate programming language.

On page 5-16, the **context** argument for the compound string routine INIT GET SEGMENT returns a quadword, not an unsigned longword as documented for the VAX binding.

On page 7-16, the **map_callback** argument for the high-level routine DIALOG BOX is supported only if the **style** argument is modal or modeless. The **map_callback** argument is ignored if the **style** argument is work area.

On page 7-38, the high-level routine LIST BOX ITEM EXISTS incorrectly lists the return as a Boolean. The correct return is **position** described as follows: If the specified item is found, the routine returns an integer that gives the position of the item in the list box. If the item is not found, the routine returns a zero.

On page 7-48, the **map_callback** argument in the high-level routine MENU is supported only if the **format** argument is pop-up or pull-down. The **map_callback** argumentis ignored if the **format** argument is work area.

On page 7-57, the high-level routine OPTION MENU is missing the **sub_menu_id** argument. This argument follows the **label** argument and precedes the **entry_callback** argument. The description of the argument is identical to that in the low-level routine OPTION MENU CREATE.

On page 8-7, the following text should be added to the description of the **destroy_callback** attribute:

> Unlike all other toolkit callbacks, **destroy_callback** returns only two valid arguments: widget_id and tag. The callback_data argument is null. Therefore, applications should avoid setting **destroy_callback** to call general callback routines (for example, routines to handle activate, arm, disarm, and similar actions), because these routines depend on the callback_data argument. For Ada developers this is particularly important, since Ada requires a meaningful value for the callback_data argument.

On page 8-92, the **map_callback** and **unmap_callback** attributes should be removed from the low-level routine MENU CREATE, as they are not supported for work area menus. Consequently, these attributes are not inherited by the subclasses of menu documented in the low-level routines MENU BAR CREATE (page 8-89), OPTION MENU CREATE (page 8-110), and RADIO BOX CREATE (page 8-123).

On pages 8-102 and 8-104, the attributes **map_callback** and **unmap_callback** should be added to the low-level routines MENU POPUP CREATE (page 8-102) and MENU PULLDOWN CREATE (page 8-104).

5.9 *VMS DECwindows User Interface Language Reference Manual*

V5.1

This section describes corrections to the *VMS DECwindows User Interface Language Reference Manual.*

UIL incorrectly allows the specification of three character sets that are not supported in XUI. The unsupported character sets are as follows:

DEC_MCS
DEC_HEBREW
DEC_HEBREW_LR

Future versions of the UIL compiler will issue an error message when these character sets are encountered.

Support for international applications is provided through use of other supported character sets. In particular, support for Hebrew applications is provide by the ISO_HEBREW character set. Support for applications using the DEC_MCS character set is provided by the ISO_LATIN1 (the default character set in UIL).

The following sections of the *VMS DECwindows User Interface Language Reference Manual* are affected:

- Table 2–5, Examples of String Literal Syntax. Substitute ISO_HEBREW for DEC_HEBREW.
- First paragraph in Section 2.4.1.1, Compound String Literals. Substitute ISO_HEBREW for DEC_HEBREW.
- Section 2.4.1.1, code example. Substitute ISO_HEBREW for DEC_HEBREW.

```
#DEC_HEBREW"txet werbeh"&#ISO_LATIN8"latin text"
```

- Table 2–7, UIL-Supported Character Sets. Eliminate DEC_MCS, DEC_HEBREW and DEC_HEBREW_LR.
- Section 2.4.1.3, code example following Table 2–7. Substitute ISO_HEBREW for DEC_HEBREW.

```
#DEC_HEBREW"tfel ot thgir morf og sretcarahc"
```

- Table 2–8, Parsing Rules for Character Sets. Eliminate DEC_MCS, DEC_HEBREW and DEC_HEBREW_LR.

- Section 2.4.1.3, fourth paragraph following Table 2–8. Substitute ISO_ HEBREW for DEC_HEBREW.

5.10 *VMS DECwindows User's Guide*

V5.2

In the section "Composing Special Characters" in Chapter 2, the procedure for composing and canceling special characters should read as follows:

1. Find the character you want to create in column 1.
2. To compose a 3-stroke sequence, press and hold the Compose key while you press the space bar, and then type the two characters in column 2.

   To compose a 2-stroke sequence, type the two characters in column 3. The desired character is displayed.

To cancel a compose sequence, press and hold the Compose key while you press the space bar, or press the <⌫ key, Tab key, Return key, or Enter key.

In the section "Displaying Remote Applications on Your Workstation Monitor" in Chapter 8, the seventh paragraph should read as follows:

> You can then run your application on ZEPHYR for display on HUBBUB if you are authorized to do so. See the section Running Applications from a DCL Command Line for more information.

In the section "Displaying Remote Applications on Your Workstation Monitor" in Chapter 8, the last paragraph is inaccurate and should be ignored.

5.11 *VMS DECwindows Xlib Programming Volume*

The following sections contain corrections to the *VMS DECwindows Xlib Programming Volume.*

5.11.1 Creating Cursors

V5.1-1

An error exists in the *VMS DECwindows Xlib Programming Volume*, Section 6.8.1, Creating Cursors. The section describes how to create a predefined DECwindows cursor. The error in both bindings occurs in the code example in which the third argument of the SET FONT routine should be a font id not a font name.

- VAX Binding

  On page 6-34 of the *VMS DECwindows Guide to Xlib Programming: VAX Binding*, the code example appears as follows:

```
INTEGER*4 CURSOR_FONT
INTEGER*4 GLYPHCURSOR

RECORD/ X$COLOR/ FORE_COLOR, BACK_COLOR
   .
   .
   .
CURSOR_FONT = X$LOAD_FONT(DPY, 'DECW$CURSOR')
CALL X$SET_FONT(DPY, GC, 'DECW$CURSOR')
GLYPHCURSOR = X$CREATE_GLYPH_CURSOR(DPY, CURSOR_FONT,
1       CURSOR_FONT, DECW$C_SELECT_CURSOR,
1       DECW$C_SELECT_CURSOR + 1, FORE_COLOR, BACK_COLOR)
CALL X$DEFINE_CURSOR(DPY, WIN, GLYPHCURSOR)
```

The fifth line in this example should be the following:

```
CALL X$SET_FONT(DPY, GC, CURSOR_FONT)
```

- MIT C Binding

  On page 6-36 of the *VMS DECwindows Guide to Xlib Programming: MIT C Binding*, the code example appears as follows:

```
Font cursorfont
Cursor glyphcursor;
XColor forecolor, backcolor;
   .
   .
   .
cursorfont = XLoadFont(dpy, "decw$cursor");
XSetFont(dpy, gc, "decw$cursor");

glyphcursor = XCreateGlyphCursor(dpy, cursorfont, cursorfont,
     decw$c_select_cursor, decw$c_select_cursor + 1,
     &forecolor, &backcolor);
XDefineCursor(dpy, win, glyphcursor);
```

The fifth line in this example should be the following:

```
XSetFont(dpy, gc, cursorfont);
```

Refer to the *VMS DECwindows Xlib Routines Reference Manual* for more information about the SET FONT routine.

5.11.2 Storing Color Values

V5.1-1

An error exists in the *VMS DECwindows Xlib Programming Volume*, Section 5.4.3, Storing Color Values. The third argument of the STORE COLOR and STORE COLORS routine is incorrectly documented as *screen_def_return* and *screen_defs_return*, respectively. It should be as follows:

- VAX Binding

  On page 5-21 of the *VMS DECwindows Guide to Xlib Programming: VAX Binding*, the STORE COLOR routine has the following format:

```
X$STORE_COLOR(display, colormap_id, color_def)
```

The STORE COLORS routine has the following format:

```
X$STORE_COLORS(display, colormap_id, color_defs,
      num_colors)
```

- MIT C Binding

  On page 5-19 of the *VMS DECwindows Guide to Xlib Programming: MIT C Binding*, the STORE COLOR routine has the following format:

```
XStoreColor(display, colormap_id, color_def)
```

  The STORE COLORS routine has the following format:

```
XStoreColors(display, colormap_id, color_defs,
      num_colors)
```

  Refer to the *VMS DECwindows Xlib Routines Reference Manual* for more information about these routines.

5.12 *VMS DECwindows Xlib Routines Reference Manual*

V5.2

On page 6-27 of the *VMS DECwindows Xlib Routines Reference Manual*, replace the first two paragraphs in the description section of CREATE IMAGE with the following text:

> CREATE IMAGE allocates memory for the image data structure. It initializes the image data structure with the values you specify in the arguments. The image data structure returned by CREATE IMAGE is initialized according to attributes of the server and not the client.
>
> While CREATE IMAGE allocates the memory needed for an image structure for the specified display, it does not allocate space for the image itself. Rather, it initializes the structure byte-order, bit-order, and bitmap-unit values from the display and returns a pointer to the image data structure. Use this pointer in subsequent routines to refer to the image data structure.

Xlib Routine Corrections

V5.1

This section describes corrections to the *VMS DECwindows Xlib Routines Reference Manual*. The corrections are as follows:

- The **data** argument in X$CREATE_IMAGE should be passed by reference, not by descriptor.
- The **buff_return** argument in X$LOOKUP_STRING should be passed by descriptor, not by reference.
- The **name_len_return** argument in X$GET_ATOM_NAME uses a longword instead of of a word.
- The **len_return** argument in X$LIS_FONTS uses a longword instead of a word.
- The documentation for X$DELETE_MODIFIERMAP_ENTRY is incorrect. X$DELETE_MODIFIERMAP_ENTRY returns a status that indicates whether the routine completed successfully.

The corrected version of the X$DELETE_MODIFIERMAP_ENTRY routine is as follows:

DELETE MODIFIERMAP ENTRY

Deletes an entry from a modifier key map structure.

VAX FORMAT

status_return =
X$DELETE_MODIFIERMAP_ENTRY
(modifier_keys, keycode_entry, modifier, modifier_keys_return)

argument information

Argument	Usage	Data Type	Access	Mechanism
status_return	cond_value	uns longword	write	value
modifier_keys_ return	record	x$modifier_ keymap	write	reference
modifier_keys	record	x$modifier_ keymap	read	reference
keycode_entry	identifier	uns longword	read	reference
modifier	longword	uns longword	read	reference
modifier_keys_ return	record	x$modifier_ keymap	write	reference

MIT C FORMAT

modifier_keys_return = **XDeleteModifiermapEntry**
(modifier_keys, keycode_entry, modifier)

argument information

```
XModifierKeymap XDeleteModifiermapEntry(modifier_keys,
                                        keycode_entry,
                                        modifier)
  XModifierKeymap *modifier_keys;
  KeyCode  keycode_entry;
  int modifier;
```

RETURNS

status_return (VAX only)

Whether the routine completed successfully.

modifier_keys_return (MIT C only)

A pointer to a modifier keys structure. DELETE MODIFIER MAP ENTRY returns the revised modifier key map structure to this client-supplied structure.

ARGUMENTS

modifier_keys
A pointer to the modifier key map structure from which you want to delete an entry.

keycode_entry
The key code that is to be deleted.

modifier
The modifier for which you want to delete a key symbol. There are eight modifiers in the order (starting from zero) shift, lock, control, mod1, mod2, mod3, mod4, and mod5. You can pass the integer value or one of the following constants:

VAX	MIT C
X$C_SHIFT_MAP_INDEX	Shift
X$C_LOCK_MAP_INDEX	Lock
X$C_CONTROL_MAP_INDEX	Control
X$C_MOD1_MAP_INDEX	Mod1
X$C_MOD2_MAP_INDEX	Mod2
X$C_MOD3_MAP_INDEX	Mod3
X$C_MOD4_MAP_INDEX	Mod4
X$C_MOD5_MAP_INDEX	Mod5

modifier_keys_return (VAX only)
A pointer to a modifier keys structure. DELETE MODIFIER MAP ENTRY returns the revised modifier key map structure to this client-supplied structure.

DESCRIPTION

DELETE MODIFIERMAP ENTRY deletes the specified key code from the set that controls the specified modifier. DELETE MODIFIERMAP ENTRY returns the resulting modifier key map structure.

The modifier map is not shrunk if all of the rows in a column are zero and the number of keys per modifier is 1. See the INSERT MODIFIERMAP ENTRY routine for more information.

5.13 *VMS Device Support Manual*

The following sections contain information concerning the *VMS Device Support Manual*.

5.13.1 Corrections to Routine Descriptions

V5.0

The following corrections should be made to the *VMS Device Support Manual*.

- The description of the operating system routine EXE$ALOPHYCNTG contains incorrect information regarding its synchronization method.

  EXE$ALOPHYCNTG returns control to its caller at IPL$_SYNCH, *not* at the caller's IPL as stated in the manual.

- The description of the operating system routines EXE$DEBIT_BYTCNT_ALO and EXE$DEBIT_BYTCNT_BYTLM_ALO neglected to mention that these routines can return an additional status value in R0.

  Because these routines call EXE$ALLOCBUF to allocate memory, they can pass the return status SS$_INSFMEM to their callers if sufficient memory is not available to satisfy the request.

- The *VMS Device Support Manual* erroneously lists R1 as being destroyed by the actions of the executive routine COM$POST. In fact, COM$POST only destroys R0.

This information will be added to a future revision of the *VMS Device Support Manual*.

5.13.2 Synchronization Corrections

V5.0-1

The description of the operating system routines EXE$DEBIT_BYTCNT_ALO and EXE$DEBIT_BYCNT_BYTLM_ALO in the *VMS Device Support Manual* contain incorrect information in the description of their synchronization method.

Each of these routines returns control to its caller at IPL$_ASTDEL and *not* at the caller's IPL as stated in the manual. (Note that the similar routines EXE$DEBIT_BYTCNT(_NW) and EXE$DEBIT_BYTCNT_BYTLM(_NW) *do* return control to the caller at the caller's IPL as documented.)

A future revision of the *VMS Device Support Manual* will contain this correction.

5.14 *VMS File Definition Language Facility Manual*

V5.2

On page FDL-51 of the *VMS File Definition Language Facility Manual*, the following description of the /GRANULARITY qualifier for the DCL command EDIT/FDL supersedes the printed description.

/GRANULARITY Qualifier

The /GRANULARITY qualifier specifies the number of key-associated areas in an indexed file. A file may contain from 1 to 255 key-associated areas and each area may contain one or more index levels from one or more keys.

Each key definition may specify one or more of the following area designations, DATA_AREA, LEVEL1_INDEX_AREA and INDEX_AREA. During input processing, optimization and redesign functions assign two areas per key: one for data and one for both indexes. During output processing, the area designators are adjusted according to the granularity specified. Checks are made to exclude areas that have no key indexes and to create new key-indexed areas where none previously existed.

To assign more than two areas per key (DOUBLE) or nonstandard key/area associations, you must use an interactive session. Start the interactive session by specifying the qualifier /GRANULARITY=DOUBLE, and then create the new areas. Then set the corresponding area designators to reference the newly created areas on a per key basis.

Format

/GRANULARITY=\n

Qualifier Value

n

The following table shows the relationship between granularity, key and area for various qualifier values. The acceptable values are the numerics 1, 2, 3, 4 or the literal values ONE, TWO, THREE, FOUR or DOUBLE. The default is a granularity of three (3).

Granularity	Key and Area Relationships
1	All indexes for all keys are assigned to AREA 0
2	Primary KEY data to AREA 0. All other indexes for all other keys to AREA 1.
3	Primary KEY data to AREA 0; Primary KEY indexes to AREA 1. All other indexes for all other keys to AREA 2.
4	Primary KEY data to AREA 0; Primary KEY indexes to AREA 1. All other key data to AREA 2; all other key indexes to AREA 3.
DOUBLE	Primary KEY data to AREA 0; Primary KEY indexes to AREA 1. All other key data to AREA (key_number*2); all other key indexes to AREA ((key_number*2)+1).

Example

```
$
```

This command begins an interactive session in which the output granularity will be 2. TEMP_DATA.FDL is the name of the FDL file being processed.

5.15 *VMS I/O User's Reference Manual: Part II*

V5.0

The last paragraph of Section 3.3.3 in the *VMS I/O User's Reference Manual: Part II* should be corrected as follows:

> Table 3-3 lists the device characteristics for the set mode and set characteristics function. The device class value **must be** DC$_REALTIME. The device type value **must be** DT$_DR11W or DT$_XA_DRV11WA. These values are defined by the $DCDEF macro.

5.16 *VMS Linker Utility Manual*

V5.0

The *VMS Linker Utility Manual* noted that a linker produces a debugger symbol table (DST) only if the /DEBUG qualifier is specified at link time. In fact, the linker produces a DST when either the /DEBUG or /TRACEBACK qualifier is specified at link time.

5.17 *VMS Mail Utility Manual*

V5.0

MAIL now displays or selects messages from the current folder. If there is no currently selected folder, MAIL displays or selects messages from the NEWMAIL folder. If there is no new mail, MAIL displays the MAIL folder.

5.18 *VMS Monitor Utility Manual*

V5.2

On page MON–42 of the *VMS Monitor Utility Manual*, the Network Control Program (NCP) example is no longer valid. Refer to the following example instead:

```
$
$
NCP>




NCP>




NCP>
```

V5.0

On page MON-97 in the *VMS Monitor Utility Manual*, the contents of a file named SUBMON.COM are listed. This file is shown as part of an example of a MONITOR data collection procedure and is included in the SYS$EXAMPLES directory. The following lines of that file establish working set values of 100:

```
/WORKING_SET=100 -
/MAXIMUM_WORKING_SET=100 -
```

These values are too low. They should be changed to a higher value such as 512, in order to avoid excessive paging by the detached MONITOR process. You may also want to consider raising the working set extent from 512 to 1024 or higher.

5.19 *VMS Networking Manual*

V5.2

On page 3-58 of the *VMS Networking Manual*, the following description of correcting asynchronous buffer problems is added.

An insufficient number of receive buffers on asynchronous DDCMP lines can cause such network problems as timeouts and loss of packets. If these problems occur, you can enter the Network Control Program (NCP) command SHOW CIRCUIT to confirm whether an insufficient number of receive buffers is the cause:

```
$
NCP>
```

Check the Remote Buffer Errors listed for the circuit. If the counters show any Remote Buffer Errors that include the words "buffer unavailable," you should increase the number of receive buffers for the line. First, use the NCP command SHOW LINE *line-id* CHARACTERISTICS to find out the current number of receive buffers for the line, as in the following command:

```
NCP>
```

The resulting display lists the characteristics for the line, including the number of receive buffers. For example:

```
Line Volatile Characteristics as of 5-APR-1989 9:32:50

   Line = TT-0-0

   Receive Buffers          = 4
   Controller               = normal
   Duplex                   = full
   Protocol                 = DDCMP point
   Retransmit timer         = 3000
   Line speed               = 9600
   Switch                   = disabled
   Hangup                   = disabled
```

Then use the NCP command SET LINE to change the number of receive buffers in the volatile database. In the following example, the number of receive buffers shown in the previous example (four buffers) is increased to six:

```
NCP>
NCP>
NCP>
```

To change the number of receive buffers in the permanent database as well, use the NCP command DEFINE LINE.

V5.0

On page 3-59, Section 3.6.2.2, the last paragraph should read as follows:

> This feature can be used to maximize performance over high-speed links such as Ethernet and the CI. To maximize performance on the Ethernet, one would use a large value for the BUFFER SIZE parameter, which would cause all logical links between adjacent nodes on the Ethernet to use that larger message size. This maximization would also work for a CI. However, on a CI the BUFFER SIZE parameter must be less than or equal to the SYSGEN parameter SCSMAXDG. Failure to do this will result in an unusable CI circuit.

5.20 *VMS Network Control Program Manual*

V5.0

On page NCP-10 of the *VMS Network Control Program Manual*, the description of the value of an object-name should read as follows:

> A string of up to 12 characters, consisting of alphanumeric characters, the dollar sign ($), or the underscore (_).

5.21 *VMS Version 5.2 New Features Manual*

V5.2

On page 3-8 of the *VMS Version 5.2 New Features Manual*, the format of the VMSINSTAL SET POSTINSTALL callback is incorrectly listed as follows:

```
$
```

The correct format is:

```
$
```

Use the option parameter (P3) to specify one of the following:

- YES—Use this option if you want to enable the postinstall phase.
- NO—Use this option if you want to disable the postinstall phase

If the option parameter is omitted the default is NO.

5.22 *Overview of VMS Documentation*

V5.2

In Table 4, "Order Numbers for Manuals in the System Management Subkit", the order number for the *Guide to VMS System Security* (Volume 3/Security) is incorrect. The correct order number is AA-LA40B-TE.

In Table 5, "Order Numbers for Manuals in the Programming Subkit", the order number for the *VMS Debugger Manual* (Volume 2A/Utilities) is incorrect. The correct order number is AA-LA59B-TE.

5.23 *VMS Record Management Services Manual*

V5.0

The *VMS Record Management Services Manual* contains the following error in the second example for the $PUTMSG system service:

```
NEWSIGARGS(ELEMENT) = 10
```

The correct example is as follows:

```
NEWSIGARGS(ELEMENT) = MIN(SIGARGS(1)-2,10)
```

5.24 *VAX RMS Journaling Manual*

V5.0

The following sections contain information concerning the *VAX RMS Journaling Manual*.

5.24.1 Additional VAX RMS Journaling Error Messages

V5.0

The following VAX RMS Journaling error messages do not appear in Appendix A of the *VAX RMS Journaling Manual*.

JND, journaling disabled on this file,

Facility: RMS, VMS Record Management Services

Explanation: This file is disabled for journaling by the Backup Utility. The Backup Utility disables all backup copies of files marked for after-image or before-image journaling to prevent conflict with the original data file.

User Action: Use the SET FILE command to mark the backup copy for after-image or before-image journaling if you want to use the backup copy rather than the original data file.

JNLNOTAUTH, RMS Journaling not authorized; operation not performed,

Facility: RMS, VMS Record Management Services

Explanation: An attempt was made to open a file marked for RMS journaling for write access on a node that is not authorized to perform RMS journaling. Access to the file has been denied. The secondary status value (STV) contains the error status from the License Management Facility (LMF).

User Action: If you want to create a journal file, RMS journaling must be authorized on the node from which you access the file. Either attempt to access the file on a node that is authorized to perform RMS journaling, or fix the error condition specified in the secondary status value (STV). See the *VMS License Management Utility Manual* for more information.

If you do not want to create a journal file, unmark the file for journaling by entering the following command:

```
$ SET FILE/NOAI_JOURNAL/NOBI_JOURNAL/NORU_JOURNAL filespec
```

JNLNOTAUTH, RMS Journaling not authorized; recovery not performed,

Facility: RMSREC, RMS Recovery Utility

Explanation: An attempt was made to recover a file on a node that is not authorized to perform RMS Journaling and recovery. The file has not been recovered. The associated error message describes the error status from the License Management Facility.

User Action: Either attempt to recover the file on a node that is authorized to perform RMS journaling and recovery, or fix the error condition specified by the License Management Facility. See the *VMS License Management Utility Manual* for more information.

JNLNOTAUTH, RMS Journaling not authorized; operation still performed,

Facility: SET, Set Utility

Explanation: One of the RMS Journaling qualifiers to the DCL command SET FILE was specified on a node that is not authorized to perform RMS journaling. The specified operation is performed. The associated error messages describes the error status from the License Management Facility.

User Action: If you mark a file for journaling and then attempt to journal the file from a node that is not authorized to perform RMS journaling, the program will receive an error status. To correct the warning message, fix the specific error condition specified by the License Management Facility. See the *VMS License Management Utility Manual* for more information.

OK_RULK, record locked in recovery unit,

Facility: RMS, VMS Record Management Services

Explanation: You relocked a record using the RMS routine $FIND or $GET. This record was previously locked and released within a recovery unit, but the release was deferred until the end of that recovery unit.

User Action: None. This message indicates that the record was locked successfully.

5.24.2 VAX RMS Journaling Revisions

The following descriptions supersede the descriptions provided in the *VAX RMS Journaling Manual*.

5.24.2.1 Recovery Unit Flags Field XAB$W_RU_FLAGS

V5.0

The recovery unit flags field XAB$W_RU_FLAGS is used to specify recovery unit information. This is an optional, input-only field. The XAB$W_RU_FLAGS field has only one bit that may be set: the XAB$V_NOJOIN bit. When this bit is set during the execution of the RMS services Open or Connect, the record stream does not join any recovery unit.

If a $XABRU is specified off the $FAB at the time of an $OPEN call and the XAB$V_NOJOIN bit is set, then no record streams associated with the file join any recovery unit, unless specifically overridden when the call to $CONNECT is made. If an $XABRU is specified off the $RAB at the time

of a $CONNECT call and the XAB$V_NOJOIN bit is set, then the record stream does not join any recovery unit.

If a record stream does not join any recovery unit, and the file is marked for recovery unit journaling, only RMS operations that do not modify the contents of the file can be used. Any attempt to modify the file will result in the error message "NRU, operation prohibited outside recovery-unit".

If there is no $XABRU off either the $FAB or the $RAB, then the record stream attempts to join the default recovery unit (that is, the most recently started recovery unit).

5.24.2.2 SET FILE/RU_ACTIVE

V5.0

The SET FILE/RU_ACTIVE command sets the RU_ACTIVE attribute on a file, corresponding to the recoverable facility that you specify. The RU_ACTIVE attribute designates the recoverable facility that controls active recovery units for the file. Alternatively, when used with the /RU_FACILITY qualifier, the SET FILE/RU_ACTIVE command lets you clear the designation that a recoverable facility controls active recovery units for the specified file.

You use the SET FILE/RU_ACTIVE command in conjunction with the SET FILE/RU_FACILITY command to modify the facility that controls any active recovery units or to clear the RU_ACTIVE attribute that may be set for a given file. This can be useful if a data file is unavailable due to active recovery units and an unavailable recovery unit journal.

Caution: When you clear the RU_ACTIVE attribute (using the command SET FILE /RU_ACTIVE=0/RU_FACILITY=1), the data in the file is likely to be in an inconsistent state. Do not use the data file unless you can ensure that the data is consistent. After clearing the RU_ACTIVE attribute, you can unmark the file for journaling, delete the file, and recreate a consistent file using a backup copy.

You can determine the recoverable facility that controls active recovery units (if any) for the file by entering the DCL command DIRECTORY/FULL or DUMP/HEADER. You can use the ANALYZE/RMS_FILE/RU_JOURNAL command to determine the state of any active recovery units.

SET FILE/[NO]RU_ACTIVE[=ru-facility] *data-filespec*

The **ru_facility** parameter is the number or name of a recoverable facility. It can be an integer from 0 through 255, or it can be the name of a Digital-registered recoverable facility.

Facility numbers 1 through 127 are reserved by Digital; facility numbers 128 through 255 are available for user-written recoverable facilities. RMS is recoverable facility 1; if you specify the number "1", that is equivalent to using the text "RMS". The number 0 corresponds to no recoverable facility and is equivalent to using the qualifier /NORU_ACTIVE. Currently, the only Digital-defined recoverable facility is 1 (RMS).

The **data-filespec** parameter identifies the file that is to be operated upon with the SET FILE command.

/LOG
/NOLOG(default)
Controls whether the SET FILE command displays the file specification and the type of facility that has been specified. By default, this information is not displayed.

Example

```
$
```

If the file WEEKLY.DAT were unavailable due to active recovery units and an unavailable recovery unit journal file, you could use this command to gain access to the file. In this example, the recoverable facility is identified as RMS by the /RU_FACILITY=1 qualifier. The RU active file attribute that indicates active RMS recovery units for the file WEEKLY.DAT is cleared by the RU_ACTIVE=0 qualifier.

Caution: Be aware that the data in the file might be inconsistent if there are active recovery units. Digital recommends that you not use the contents of the data file unless you can verify that the data is consistent.

5.24.2.3 SET FILE /RU_FACILITY

V5.0

The SET FILE/RU_FACILITY command allows you to identify the recoverable facility that controls active recovery units on the file. You can use any other SET FILE qualifier with the /RU_FACILITY qualifier.

When a data file has active recovery units and RMS journaling cannot resolve the recovery units (for example, if the recovery unit journal file is unavailable), the data file cannot be opened or deleted. The presence of active recovery units prevents you from unmarking (or marking) a file for journaling. With the SET FILE/RU_FACILITY/RU_ACTIVE command, you can clear the designation that a recoverable facility controls active recovery units for the data file.

Caution: When you clear the RU_ACTIVE attribute (using the command SET FILE /RU_ACTIVE=0/RU_FACILITY=1), the data in the file is likely to be in an inconsistent state. Do not use the data file unless you can ensure that the data is consistent. After clearing the RU_ACTIVE attribute, you can unmark the file for journaling, delete the file, and re-create a consistent file using a backup copy.

You can determine the recoverable facility that controls active recovery units (if any) for the file by entering the DCL command DIRECTORY/FULL or DUMP/HEADER. You can use the ANALYZE/RMS_FILE/RU_JOURNAL command to determine the state of any active recovery units.

SET FILE /RU_FACILITY=ru-facility *data-filespec*

The **ru_facility** parameter is the number or name of a recoverable facility. It can be an integer from 0 through 255, or it can be the name of a Digital-registered recoverable facility.

Facility numbers 1 through 127 are reserved by Digital; facility numbers 128 through 255 are available for user-written recoverable facilities. RMS is recoverable facility 1; if you specify the number "1", that is equivalent to using the text "RMS". The number 0 corresponds to no recoverable facility. Currently, the only Digital-defined recoverable facility is 1 (RMS).

The recoverable facility that you specify is an input parameter that is only used to open the file; it does not modify any file attributes.

The **data-filespec** parameter identifies the file that is to be operated upon with the SET FILE command.

Examples

1
```
$  SET FILE /RU_FACILITY=1 /NORU_JOURNAL /NOAI_JOURNAL /LOG SAVINGS.DAT
%SET-I-FILUNMARKAI, $DISK1:[PERSONAL]SAVINGS.DAT;1 unmarked for RMS
after-image journaling
%SET-I-FILUNMARKRU, $DISK1:[PERSONAL]SAVINGS.DAT;1 unmarked for RMS
recovery-unit journaling
%SET-I-MODIFIED, $DISK1:[PERSONAL]SAVINGS.DAT;1 modified

$ DELETE SAVINGS.DAT.
```

This example shows the use of the /RU_FACILITY qualifier to allow SET FILE access to a data file. The SET FILE command identifies the recoverable facility holding the file and it also unmarks the file for recovery unit and after-image journaling. After these steps, it is then possible to delete the data file.

Note: Note that if it becomes necessary to use the /RU_FACILITY qualifier because of active recovery units, the data in the file may be inconsistent, and the data in the file should not be used unless you can verify that it is valid and consistent.

2
```
$ SET FILE /RU_FACILITY=RMS /RU_ACTIVE=0  SALES.DAT
```

In this example, the recoverable facility for the file SALES.DAT is identified as RMS by the /RU_FACILITY=RMS qualifier, and the RU active file attribute (which indicates active RMS recovery units) is cleared by the RU_ACTIVE=0 qualifier. If the file SALES.DAT were unavailable due to active recovery units and an unavailable recovery unit journal file, you could use this command to gain access to the file.

As in the previous example, this operation leaves the data file in an inconsistent state. Generally, you would use this command in order to delete the data file and rebuild it from other sources.

5.24.3 STREAM Formats not Supported for Shared Sequential Files

V5.0

Chapter 4 and Appendix A of the *VAX RMS Journaling Manual* state that STREAM formats are not supported when using recovery unit journaling with shared sequential files. All three STREAM formats are not supported: STREAM, STREAM_CR, and STREAM_LF. These formats are indicated by the symbolic values FABC_STM, FABC_STMCR, and FAB$C_STMLF in the FAB$B_RFM field of the FAB.

5.25 *VMS RTL Library (LIB$) Manual*

The following sections contain information concerning the *VMS RTL Library (LIB$) Manual.*

5.25.1 LIB$ADAWI Routine

V5.0

On page LIB-3, the LIB$ADAWI routine description should read "Add Aligned Word with Interlock" instead of "Add Adjacent Word with Interlock".

The **result** argument description incorrectly implies that it is the sum. **Result** is actually –1, 0, or 1, denoting the sign of the **sum** argument.

5.25.2 LIB$CREATE_VM_ZONE Routine

V5.0

On pages LIB-44 and LIB-47, the description of the LIB$CREATE_VM_ ZONE routine lists the argument *number-of-areas*; however, *number-of-areas* does not exist. LIB$CREATE_VM_ZONE has thirteen, not fourteen, arguments.

5.25.3 LIB$SPAWN Routine

V5.0

On page LIB-382, the last argument to LIB$SPAWN is omitted. The last argument, **table**, is described in the following section.

table

```
VMS usage:      char_string
type:           character string
access:         read only
mechanism:      by descriptor
```

The **table** argument is the address of this file specification string's descriptor. The table specified must reside in SYS$SHARE with a file type of EXE, and it must be installed.

If omitted, the subprocess uses the same table as the parent process.

The following are general notes about LIB$SPAWN omitted from the *VMS RTL LIB$ (Library) Manual.*

Though the subprocess inherits the caller's process privileges as its own process privileges, the set of authorized privileges in the subprocess is inherited from the caller's current privileges.

If the calling image is installed with elevated privileges, it should disable those privileges around the call to LIB$SPAWN unless the environment of the subprocess is strictly controlled. Otherwise, there is a possibility of a security breach due to elevated privileges accidentally being made available to the user in the spawned process.

The **cli** argument must be specified in all uppercase characters.

5.25.4 LIB$SYS_TRNLOG Routine

V5.1

The RTL routine LIB$SYS_TRNLOG was removed from the VMS Version 5.0 *VMS RTL Library (LIB$) Manual* because the system service that it calls, $TRNLOG, is obsolete. However, the RTL routine LIB$SYS_TRNLOG is not obsolete. Following is the routine description for LIB$SYS_TRNLOG that should have appeared in the *VMS RTL Library (LIB$) Manual* for VMS Version 5.0. See the *VMS Obsolete Features Manual* for a description of the system service $TRNLOG.

LIB$SYS_TRNLOG
Invoke $TRNLOG System Service to Translate Logical Name

The Invoke $TRNLOG System Service to Translate Logical Name routine uses the system service $TRNLOG to translate a logical name. LIB$SYS_TRNLOG returns the logical name's translation using the semantics of the caller's string.

FORMAT

LIB$SYS_TRNLOG *logical-name*
,[word-integer-dest-length]
,destination-string
[,byte-integer-table]
[,access-mode]
[,byte-integer-disable-mask]

RETURNS

VMS usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

logical-name

VMS usage: **logical_name**
type: **character string**
access: **read only**
mechanism: **by descriptor**

Logical name. The **logical-name** argument contains the address of a descriptor pointing to this logical name string.

word-integer-dest-length

VMS usage: **word_unsigned**
type: **word (unsigned)**
access: **write only**
mechanism: **by reference**

Number of characters written into **destination-string**, not counting padding in the case of a fixed-length string. The **word-integer-dest-length** argument contains the address of an unsigned word integer that is this number.

If the input string is truncated to the size specified in the **destination-string** descriptor, **word-integer-dest-length** is set to this size. Therefore, **word-integer-dest-length** can always be used by the calling program to access a valid substring of **destination-string**.

destination-string

VMS usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**
Destination string into which LIB$SYS_TRNLOG writes the logical name translation. The **destination-string** argument contains the address of a descriptor pointing to this destination string.

byte-integer-table

VMS usage: **byte_signed**
type: **byte integer (signed)**
access: **write only**
mechanism: **by reference**
Logical name table number. The **byte-integer-table** argument contains the address of a signed byte integer that is this table number.

access-mode

VMS usage: **access_mode**
type: **byte integer (unsigned)**
access: **write only**
mechanism: **by reference**
Access mode of entry (process table only). The **access-mode** argument contains the address of a signed byte integer that is this access mode.

The access modes, their numeric values, and symbolic names are as follows:

Mode	Value	Symbolic Name
Kernel	0	PSL$C_KERNEL
Executive	1	PSL$C_EXEC
Supervisor	2	PSL$C_SUPER
User	3	PSL$C_USER

byte-integer-disable-mask

VMS usage: **mask_byte**
type: **byte (unsigned)**
access: **read only**
mechanism: **by reference**
Table search disable mask. The **byte-integer-disable-mask** argument contains the address of a mask byte that is this mask.

The argument **byte-integer-disable-mask** is passed to this routine by reference and is changed to value for use by $TRNLOG.

The mask-bit settings and their resultant actions are as follows:

Bit	Action
0	Do not search system logical name table
1	Do not search group logical name table
2	Do not search process logical name table

DESCRIPTION

See the *VMS System Services Reference Manual* for a complete description of $TRNLOG.

CONDITION VALUES RETURNED

SS$_NORMAL	Routine successfully completed.
SS$_NOTRAN	Successfully completed, but the input logical name string was placed in the output buffer because no equivalence name was found.
LIB$_STRTRU	Successfully completed, but the source string was truncated on copy.
SS$_ACCVIO	The caller cannot read the logical name string or the string descriptor, or cannot write the output length, output buffer, table, or access mode field.
SS$_IVLOGNAM	The specified logical name string has a length of zero or is greater than 255 characters. String was placed in the destination string buffer because no equivalence name was found.
SS$_RESULTOVF	The destination string buffer has a length of zero, or it is smaller than the resultant string.
LIB$_FATERRLIB	Fatal internal error. An internal consistency check has failed. This usually indicates an internal error in the Run-Time Library and should be reported to Digital in a Software Performance Report (SPR).
LIB$_INVSTRDES	Invalid string descriptor. A string descriptor has an invalid value in its DSC$B_CLASS field.
LIB$_INSVIRMEM	Insufficient virtual memory. A call to LIB$GET_VM has failed because your program has exceeded the image quota for virtual memory.

5.26 *VMS RTL Parallel Processing (PPL$) Manual*

The following sections contain information concerning the *VMS RTL Parallel Processing (PPL$) Manual.*

5.26.1 Flags Arguments

V5.1

All of the **flags** arguments in the RTL PPL$ routines are passed by reference, not by value. The following routines *incorrectly* show the **flags** argument as passed by value:

PPL$CREATE_SHARED_MEMORY
PPL$DELETE_SHARED_MEMORY
PPL$FLUSH_SHARED_MEMORY
PPL$TRIGGER_EVENT

5.26.2 PPL$CREATE_SHARED_MEMORY Routine

V5.0

PPL$CREATE_SHARED_MEMORY allows you to specify a file that will be mapped as the shared memory. The size of the resulting address space is the smaller of the following:

- The specified buffer size
- The size of the file being mapped

5.27 *VMS SYSMAN Utility Manual*

The following sections contain information concerning the *VMS SYSMAN Utility Manual.*

5.27.1 DISKQUOTA REMOVE Command — Correction

V5.1

The DISKQUOTA REMOVE function does not exist as documented in the *VMS SYSMAN Utility Manual.* Instead, enter the command DISKQUOTA DELETE to remove an entry from a quota file. DISKQUOTA DELETE works the same way that DISKQUOTA REMOVE is documented.

DISKQUOTA REMOVE will function in VMS Version 5.2. DISKQUOTA DELETE will continue to function in future releases as well.

5.27.2 PARAMETERS Commands — Privilege Requirements

V5.2

The descriptions of the PARAMETERS USE and PARAMETERS WRITE commands in the *VMS SYSMAN Utility Manual* contain incorrect information about the privileges required to use these commands.

On page SM-46, note the following corrections:

- The PARAMETERS USE ACTIVE command requires the CMEXEC privilege.

- The PARAMETERS USE CURRENT command requires the CMEXEC privilege and read access to SYS$SYSTEM:VAXVMSSYS.PAR.
- The PARAMETERS USE file-spec command requires read access to the file that you specify.

On page SM-47, note the follow corrections:

- The PARAMETERS WRITE command does not require the SYSLCK privilege.
- The PARAMETERS WRITE CURRENT command requires the CMEXEC privilege in addition to write access to the file.
- The PARAMETERS WRITE file-spec command requires only write access to the file.

5.27.3 PARAMETER SET /STARTUP Command

V5.2

On page SM-40, /STARTUP[=][file-spec] is listed as a qualifier to the PARAMETER SET command. The format of this qualifier is incorrect. The corrected format is as follows:

```
/STARTUP file-spec
```

The *file-spec* is not optional, and cannot be specified using an equal sign (=).

5.28 *VMS System Messages and Recovery Procedures Reference Manual: Part I*

V5.0

The following changes have been made to the user action of the INSVIRMEM system message:

> Increase the account's page file quota or the SYSGEN parameter VIRTUALPAGECNT. Also try to delete strings, ranges, markers, windows, and buffers that are not being used. You might also want to ask your system manager to increase the available memory.

5.29 *VMS System Messages and Recovery Procedures Reference Manual: Part II*

V5.0

In the *VMS System Messages and Recovery Procedures Reference Manual: Part II*, the description of the message TPU$_STACKOVER recommends that you submit an SPR if this message is returned. This recommendation is incorrect. The following paragraphs provides a better explanation and recovery procedure.

TPU$_STACKOVER is returned when your VAXTPU program is too complex for VAXTPU's parser. The VAXTPU parser currently allows a maximum stack depth of 1000 syntax tree nodes. When the parser first encounters a VAXTPU statement, the parser assigns each token in the statement to a syntax tree node. For example, the statement "a := 1" contains three tokens, each of which would occupy a syntax tree node. After the parser parses this statement, only the assignment statement

remains on the stack of nodes. The "a" and the "1" are sub-trees to the assignment syntax tree node.

The most common reason for a TPU$_STACKOVER condition is that your program is not modular enough—it contains one or more large procedures whose statements occupy too many syntax tree nodes. To make your program manageable by the parser, break the large procedures into smaller ones. The other possible reasons for a TPU$_STACKOVER condition are that you have too many small procedures (in which case you must consolidate them somewhat), or that you have too many statements that are not in procedures at all.

5.30 *Introduction to VMS System Routines*

V5.2

On page A-47 of the *Introduction to VMS System Routines*, footnotes 4 and 5 to Table A-11 are in reversed order.

5.31 *Introduction to VMS System Services*

V5.2

On page 3-33 of the *Introduction to VMS System Services*, the decision box ACL ON THIS OBJECT in the flowchart of $CHKPRO operation should not be connected to the box ACCESS DENIED. If the answer is YES, the arrow should point to the CHECK ACCESSOR FOR PRIVILEGES box. The box ACCESS DENIED at the right of the page should be deleted.

On page 11-14, there is an error in the $CRMPSC example. The UPI option must be specified if the file is to be write-shared.

Incorrect example:

```
SECFAB: $FAB FNM=<SECTION.TST>, -
             FOP=UFO, -
             FAC=PUT, -
             SHR=<GET,PUT>
               .
               .
               .
```

Corrected example:

```
SECFAB: $FAB FNM=<SECTION.TST>, -
             FOP=UFO, -
             FAC=PUT, -
             SHR=<GET,PUT,UPI>
               .
               .
               .
```

5.32 *VMS System Services Reference Manual*

The following sections describe corrections in the *VMS System Services Reference Manual*.

5.32.1 $CHKPRO Service

V5.2

On page SYS-57, it is not clear that when item codes are not specified, the routine uses default values. In all cases except the item code CHP$_ACMODE, the routine uses the value of the current process. If the CHP$_ACMODE item code is not specified, the routine uses the kernel mode value, which is 0. Therefore, if CHP$_ACMODE is not specified, the routine compares two default values, which are both 0, and the check succeeds.

On page SYS-61, the explanation of the item code CHP$_MATCHDACE states that the Access Control Element (ACE) returned from the objects's Access Control List (ACL) allows the accessor to access the object. It should state that the ACE returned from the objects's ACL can either allow or deny access to the object.

On page SYS-61, in the table explaining the item code CHP$_PRIVUSED, a V, which indicates a field offset symbol, was omitted from the following symbols:

- CHP$V_SYSPRV
- CHP$V_GRPPRV
- CHP$V_BYPASS
- CHP$V_READALL

On page SYS-62, there are two values to be added to the list *Condition Values Returned.* $CHKPRO can also return:

SS$_ACLFULL	More than 20 CHP$_ACL items were given.
SS$_RIGHTSFULL	More than 11 CHP$_ADDRIGHTS items were given.

5.32.2 $CMKRNL Service

V5.2

On page SYS-66, the description of $CMKRNL has changed. The VMS Version 5.2 documentation will note that programs should not use registers R2 through R11 to pass context between the calling and called procedures. Prior to VMS version 5.0, the system service dispatcher modified only R4. The system service dispatcher now modifies R0, R1, R2 and R4 before entry into the target routine.

The VAX calling standard states that registers R2 through R11 must be preserved across procedure calls. Called procedures can modify these registers as long as they are saved and restored using the procedure entry mask mechanism. VMS system services, including the $CMKRNL system service, are invoked as called procedures, and may choose to use R2 through R11 as permitted by the calling standard.

Because the system service dispatcher did not previously use some registers that are reserved for its use, some privileged user programs have used these registers to pass context to called procedures. Those programs that used the R2 register no longer execute correctly. Programs that use registers between R3 and R11 may also produce errors in the future.

Application programs should pass context by using argument blocks rather than by using registers. See the *Introduction to VMS System Routines* for more information.

5.32.3 $CRELNM Service

V5.2

On page SYS-70, the description of the **itmlst** argument erroneously states that the **itmlst** argument is required; it is not.

5.32.4 $CREPRC Service

V5.2

On page SYS-95, the description of the **baspri** argument does not specify that if the **baspri** argument is omitted when using the BLISS or MACRO predefined system service macros, a default of 2 is used. That paragraph of the **baspri** argument description now reads as follows:

> If the **baspri** argument is not specified, the priority defaults to 2 for VAX MACRO and VAX BLISS-32 and to 0 for all other languages. If you want a subprocess to have a higher priority than its creator, you must have the ALTPRI privilege to raise the priority level. If the caller does not have this privilege, the specified base priority is compared with the caller's priority and the lower of the two values is used.

5.32.5 $CRMPSC Service

V5.2

On page SYS-107, the last entry in the flag table that describes the flag SEC$M_NO_OVERMAP is misleading because it implies that setting the flag SEC$M_NO_OVERMAP allows sections to overmap existing address space. In fact, the opposite is true. The description of the flag SEC$M_NO_OVERMAP should be:

```
Pages cannot overmap existing address
space.  Note that, by default,
pages can overmap existing address space.
```

On page SYS-115, the following status value has been added to the list of Condition Values Returned:

```
SS$_VA_IN_USE       A page in the specified input address range is already
                    mapped and the flag SEC$M_NO_OVERMAP is set.
```

5.32.6 $DCLCHM Service

V5.2

On page SYS-123, the VMS usage, type and access of the **addres** argument are listed as follows:

addres
VMS usage: **procedure**
type: **procedure entry mask**
access: **call without stack unwinding**
mechanism: **by reference**

The VMS usage, type and access of the **addres** argument have been changed as follows:

addres
VMS usage: **address**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

5.32.7 $DISMOUNT Service

V5.2

On page SYS-355, the description of the MNT$_FLAGS option MNT$_NOMNTVER, erroneously states that MNT$M_NOMNTVER applies only to disks. As of VMS Version 5.0, mount verification applies to tapes as well as disks.

5.32.8 $ENQ Service

V5.2

On page SYS-148, the description of the **efn** argument states that the event flag is set when the lock request has been granted. The event flag is also set if the lock request is cancelled. The description of the **efn** argument has been changed to the following:

> Number of the event flag to be set when the request has been granted or canceled. Cancellation occurs if you use $DEQ with the cancel modifier or if the waiting request is chosen to break a deadlock.

On page SYS-154, the description of the **astadr** argument states that this parameter is an "AST service routine to be executed when the lock is either granted or converted." This AST routine is also called when the $ENQ request is aborted because of deadlock (and the lock status block contains the condition SS$_DEADLOCK). The description of the **astadr** argument has been changed to the following:

> AST service routine to be executed when the lock is either granted or converted. The **astadr** argument is the address of the entry mask of this routine.
>
> The AST is also delivered when the lock or conversion request is cancelled. Cancellation occurs if you use $DEQ with the cancel modifier or if the waiting request is chosen to break a deadlock.

5.32.9 $FAO Service

V5.2

On page SYS-167, the first sentence of the description of $FAO incorrectly states that "The $FAO_S macro form uses a PUSHL instruction for all parameters (**p1** through **pn**)." The $FAO_S macro actually accepts arguments **P1** to **P17**.

On page SYS-177, on lines 5 and 31 of Example 8, *!5>* should be *!5*>*.

The incorrect example appears as follows:

```
.ASCID  /!5> NOW IS: !%D/
```

The corrected example should appear as follows:

```
.ASCID  /!5*> NOW IS: !%D/
```

On page SYS-177, on line 5 of Example 9, */#_* should be */5*_*. The incorrect example appears as follows:

```
.ASCID  /DATE: !11%D!#_TIME: !5%T/
```

The corrected example should appear as follows:

```
.ASCID  /DATE: !11%D!5*_TIME: !5%T/
```

5.32.10 $FORMAT_ACL Service

V5.2

On page SYS-195, the description of the alarm ACE states that ACE$T_AUDITNAME contains a counted ASCII string. There is no requirement for the string to be counted.

On page SYS-200, in the description of the last table item ACE$L_KEY, the *identifier* field incorrectly states that the "The number of longwords is implied by ACE$B_LENGTH." The number of longwords is implied by ACE$B_SIZE.

5.32.11 $GETQUI Service

V5.2

On pages SYS-272 and SYS-273, the explanation of the item codes QUI$_INTERVENING_BLOCKS and QUI$_INTERVENING_JOBS has been changed to read as follows:

QUI$_INTERVENING_BLOCKS

When you specify QUI$_INTERVENING_BLOCKS, $GETQUI returns, as a longword integer value, the size (in blocks) of files associated with pending jobs in the queue that were skipped during the current call to $GETQUI. These jobs were not reported because they did not match the selection criterion in effect for the call to $GETQUI.

QUI$_INTERVENING_BLOCKS is zero when:

- The job is not a pending job
- The job that matches the selection criterion is the first pending job in the queue
- The preceding pending job in the queue was reported in the previous call to $GETQUI

QUI$_INTERVENING_JOBS

When you specify QUI$_INTERVENING_JOBS, $GETQUI returns, as a longword integer value, the number of pending jobs in the queue that were skipped during the current call to $GETQUI. These jobs were not reported because they did not match the selection criterion in effect for the call to $GETQUI.

QUI$_INTERVENING_JOBS is zero when:

- The job is not a pending job
- The job that matches the selection criterion is the first pending job in the queue
- The preceding pending job in the queue was reported in the previous call to $GETQUI

The item codes QUI$_INTERVENING_BLOCKS and QUI$_INTERVENING_JOBS were incorrectly documented in VMS Version 5.0 as being supported by the QUI$_DISPLAY_ENTRY function code and the QUI$_DISPLAY_JOB function code. They are only supported by the QUI$_DISPLAY_JOB function code.

5.32.12 $GETSYI Service

V5.2

On page SYS-301, the description of the item codes SYI$_ACTIVECPU_CNT and SYI$_AVAILCPU_CNT fails to note that the $GETSYI service returns this information only for the local VAX node.

The new wording for these item codes is as follows:

$GETSYI Item Codes

SYI$_ACTIVECPU_CNT
When you specify SYI$_ACTIVECPU_CNT, $GETSYI returns a count of CPUs actively participating in the current boot of the Symmetric MultiProcessing (SMP) system. The $GETSYI service returns this information for the local node only.

Because this number is a longword, the **buffer length** field in the item descriptor should specify 4 bytes.

SYI$_AVAILCPU_CNT

When you specify SYI$_AVAILCPU_CNT, $GETSYI returns the number of CPUs available in the current boot of the SMP system. The $GETSYI service returns this information for the local node only.

Because this number is a longword, the **buffer length** field in the item descriptor should specify 4 bytes.

5.32.13 $GETUAI Service

V5.2

On page SYS-316, the description of UAI$_ACCOUNT states that the buffer length field in the item descriptor should specify 9 bytes. It should specify 32 bytes. That description has been changed to the following:

When you specify UAI$_ACCOUNT, $GETUAI sets, as a blank-filled 32-character string, the account name of the user. An account name can include up to 8 characters. Because the account name is a blank-filled string, however, the buffer length field of the item descriptor should specify 32 bytes.

On page SYS-324, UAI$_USERNAME, a $GETUAI item code, has been eliminated. UAI$_USERNAME cannot return the username of the owner of a specified job. UAI$_USERNAME returns only the username that you enter as an argument. Use $GETJPI to return job information.

5.32.14 $MOD_IDENT Service

V5.2

The following status value has been added to the list of Condition Values Returned:

SS$_DUPLNAM	The specified identifier name already exists in the rights database.

5.32.15 $PUTMSG Service

V5.2

On page SYS-371, the description of $PUTMSG is incomplete and has been changed to the following:

> The Put Message service is a generalized message-formatting and output routine used by VMS to write informational and error messages to processes. These messages are written to SYS$ERROR and SYS$OUTPUT. Informational messages are written to SYS$OUTPUT only; error messages are written to SYS$ERROR. Error messages are also written to SYS$OUTPUT if it has a different definition from SYS$ERROR.

5.32.16 $QIO Service

V5.2

On page SYS-379, the **func** argument is described as both a word and a longword. The **func** argument is a word, not a longword, value.

5.32.17 $SETDDIR Service

V5.2

On page SYS-516, the following format statement implies that *length-addr* and *cur-dir-addr* are optional; they are not.

```
[new-dir-addr] [,length-addr] [,cur-dir-addr]
```

The correct format statement is as follows:

```
[new-dir-addr] ,[length-addr] ,[cur-dir-addr]
```

5.32.18 $SETEXV Service

V5.2

The description of the PRVHND argument for the $SETEXV system service uses the service name $SETEF rather than $SETEXV. The description should read as follows:

> Previous condition handler address contained by the specified exception vector. The **prvhnd** argument is the address of a longword into which $SETEXV writes the handler address.

5.32.19 $SETPRV Service

V5.2

On page SYS-419, the *prmflg* parameter to the $SETPRV system service is incorrectly labeled *get jobprmflg*.

The arguments **enbflg** and **prmflg** are longword values, not byte values.

5.32.20 $SETUAI Service

V5.2

On page SYS-432, the description of UAI$_ACCOUNT states that the buffer length field in the item descriptor should specify 9 bytes. It should specify 32 bytes. That description has been changed to the following:

> When you specify UAI$_ACCOUNT, $SETUAI sets, as a blank-filled 32-character string, the account name of the user. An account name can include up to 8 characters. Because the account name is a blank-filled string, however, the buffer length field of the item descriptor should specify 32 bytes.

On page SYS-439, UAI$_USERNAME, a $SETUAI item code, has been eliminated. The item code UAI$_USERNAME could not be used to set the username of the owner of a specified job.

5.32.21 $SNDJBC Service

V5.2

On page SYS-469, the SJC$_CREATE_JOB function code has the following new item codes:

- SJC$_FIRST_PAGE
- SJC$_NO_FIRST_PAGE
- SJC$_LAST_PAGE
- SJC$_NO_LAST_PAGE

On page SYS-491, JBC$_NOSUCHENT should be included as a valid Condition Value Returned in the I/O Status Block as follows:

```
JBC$_NOSUCHENT   There is no job with the specified entry number.
```

5.32.22 $SNDOPR Service

V5.2

The following status value has been added to the list of Condition Values Returned.

OPC$_NOPERATOR	The Operator Communciation Manager (OPCOM) is not running; the message will not be sent.

5.32.23 $WFLOR Service

V5.2

On page SYS-543, the description of $WFLOR states that if all the specified event flags have been set, the process resumes execution. All event flags do not have to be set; if any of the specified event flags have been set, the process resumes execution.

5.33 *VMS Utility Routines Manual*

V5.0

The description of the TPU$CONTROL routine in Chapter 13 of the *VMS Utility Routines Manual* does not mention that TPU$CONTROL optionally accepts one parameter. The integer is passed by reference. Specifying this optional parameter prevents VAXTPU from displaying the message "Editing session is not being journaled" when the calling program gives control to VAXTPU. Specify a true (odd) integer to preserve compatibility in future releases. If you omit the parameter, VAXTPU displays the message.

In Chapter 13 of the *VMS Utility Routines Manual*, Example 13-1 (Sample VAX BLISS Template for Callable VAXTPU) does not work. Example 5–1 should be substituted.

Example 5–1 Sample VAX BLISS Template for Callable VAXTPU

```
MODULE file_io_example (MAIN = top_level,
                        ADDRESSING_MODE (EXTERNAL = GENERAL)) =

BEGIN

FORWARD ROUTINE
    top_level,                  ! Main routine of this example
    tpu_init,                   ! Initialize TPU
    tpu_io;                     ! File I/O routine for TPU
!
! Declare the stream data structure passed to the file I/O routine
!
MACRO
    stream_file_id =  0, 0, 32, 0 % ,    ! File ID
    stream_rat =      6, 0,  8, 0 % ,    ! Record attributes
    stream_rfm =      7, 0,  8, 0 % ,    ! Record format
    stream_file_nm =  8, 0,  0, 0 % ;    ! File name descriptor
!
! Declare the routines that would actually do the I/O.  These must be supplied
! in another module
!
EXTERNAL ROUTINE
    my_io_open,                 ! Routine to open a file
    my_io_close,                ! Routine to close a file
    my_io_get_record,           ! Routine to read a record
    my_io_put_record;           ! Routine to write a record

!
! Declare the VAXTPU routines
!
EXTERNAL ROUTINE
    tpu$fileio,                 ! VAXTPU's internal file I/O routine
    tpu$handler,                ! VAXTPU's condition handler
    tpu$initialize,             ! Initialize VAXTPU
    tpu$execute_inifile,        ! Execute the initial procedures
    tpu$execute_command,        ! Execute a VAXTPU statement
    tpu$control,                ! Let user interact with VAXTPU
    tpu$cleanup;                ! Have VAXTPU cleanup after itself
```

(continued on next page)

Example 5–1 (Cont.) Sample VAX BLISS Template for Callable VAXTPU

```
!
! Declare the VAXTPU literals
!
EXTERNAL LITERAL
    tpu$k_close,                ! File I/O operation codes
    tpu$k_close_delete,
    tpu$k_open,
    tpu$k_get,
    tpu$k_put,

    tpu$k_access,               ! File access codes
    tpu$k_io,
    tpu$k_input,
    tpu$k_output,

    tpu$_calluser,             ! Item list entry codes
    tpu$_fileio,
    tpu$_outputfile,
    tpu$_sectionfile,
    tpu$_commandfile,
    tpu$_filename,
    tpu$_journalfile,
    tpu$_options,

    tpu$m_recover,              ! Mask for values in options bitvector
    tpu$m_journal,
    tpu$m_read,
    tpu$m_command,
    tpu$m_create,
    tpu$m_section,
    tpu$m_display,
    tpu$m_output,

    tpu$m_reset_terminal,       ! Masks for cleanup bitvector
    tpu$m_kill_processes,
    tpu$m_delete_exith,
    tpu$m_last_time,

    tpu$_nofileaccess,          ! VAXTPU status codes
    tpu$_openin,
    tpu$_inviocode,
    tpu$_failure,
    tpu$_closein,
    tpu$_closeout,
    tpu$_readerr,
    tpu$_writeerr,
    tpu$_success;

ROUTINE top_level =

    BEGIN
!++
! Main entry point of your program
!--
! Your_initialization_routine must be declared as a BPV
```

(continued on next page)

Example 5–1 (Cont.) Sample VAX BLISS Template for Callable VAXTPU

```
    LOCAL
        initialize_bpv: VECTOR [2],
        status,
        cleanup_flags;
    !
    ! First establish the condition handler
    !
    ENABLE
        tpu$handler ();
    !
    ! Initialize the editing session, passing TPU$INITIALIZE the address of
    ! the bound procedure value which defines the routine which VAXTPU is
    ! to call to return the initization item list
    !
    initialize_bpv [0] = tpu_init;
    initialize_bpv [1] = 0;
    tpu$initialize (initialize_bpv);
    !
    ! Call VAXTPU to execute the contents of the command file, the debug file
    ! or the TPU$INIT_PROCEDURE from the section file.
    !
    tpu$execute_inifile();
    !
    ! Let VAXTPU take over.
    !
    tpu$control();
    !
    ! Have VAXTPU cleanup after itself
    !
    cleanup_flags = tpu$m_reset_terminal OR     ! Reset the terminal
                    tpu$m_kill_processes OR     ! Delete Subprocesses
                    tpu$m_delete_exith OR       ! Delete the exit handler
                    tpu$m_last_time;            ! Last time calling the editor

    tpu$cleanup (cleanup_flags);

    RETURN tpu$_success;

    END;
ROUTINE tpu_init =

    BEGIN
```

(continued on next page)

Example 5–1 (Cont.) Sample VAX BLISS Template for Callable VAXTPU

```
!
! Allocate the storage block needed to pass the file I/O routine as a
! bound procedure variable as well as the bitvector for the initialization
! options
!
OWN
    file_io_bpv: VECTOR [2, LONG]
                 INITIAL (TPU_IO, 0),
    options;
!
! These macros define the file names passed to VAXTPU
!
MACRO
    out_file = 'OUTPUT.TPU' % ,
    com_file = 'TPU$COMMAND' % ,
    sec_file = 'TPU$SECTION' % ,
    inp_file = 'FILE.TPU' % ;
    !
! Create the item list to pass to VAXTPU.  Each item list entry consists of
! two words which specify the size of the item and its code, the address of
! the buffer containing the data, and a longword to receive a result (always
! zero, since VAXTPU does not return any result values in the item list)
!
!               +---------------------------------+
!               | Item Code      | Item Length    |
!               +----------------+----------------+
!               |        Buffer Address           |
!               +---------------------------------+
!               |    Return Address (always 0)    |
!               +---------------------------------+
!
! Remember that the item list is always terminated with a longword containing
! a zero
!
BIND
    item_list = UPLIT BYTE (
        WORD (4),                       ! Options bitvector
        WORD (tpu$_options),
        LONG (options),
        LONG (0),

        WORD (4),                       ! File I/O routine
        WORD (tpu$_fileio),
        LONG (file_io_bpv),
        LONG (0),

        WORD (%CHARCOUNT (out_file)),    ! Output file
        WORD (tpu$_outputfile),
        LONG (UPLIT (%ASCII out_file)),
        LONG (0),

        WORD (%CHARCOUNT (com_file)),    ! Command file
        WORD (tpu$_commandfile),
        LONG (UPLIT (%ASCII com_file)),
        LONG (0),
```

(continued on next page)

Example 5–1 (Cont.) Sample VAX BLISS Template for Callable VAXTPU

```
            WORD (%CHARCOUNT (sec_file)),   ! Section file
            WORD (tpu$_sectionfile),
            LONG (UPLIT (%ASCII sec_file)),
            LONG (0),

            WORD (%CHARCOUNT (inp_file)),    ! Input file
            WORD (tpu$_filename),
            LONG (UPLIT (%ASCII inp_file)),
            LONG (0),

            LONG (0));                      ! Terminating longword of 0
    !
    ! Initialize the options bitvector
    !
    options = tpu$m_display OR              ! We have a display
              tpu$m_section OR              ! We have a section file
              tpu$m_create OR               ! Create a new file if one does not
                                            !   exist
              tpu$m_command OR              ! We have a section file
              tpu$m_output;                 ! We supplied an output file spec

    !
    ! Return the item list as the value of this routine for VAXTPU to interpret
    !
    RETURN item_list;

    END;                                    ! End of routine tpu_init
ROUTINE tpu_io (p_opcode, stream: REF BLOCK [ ,byte], data) =
!
! This routine determines how to process a TPU I/O request
!
    BEGIN

    LOCAL
        status;
!
! Is this one of ours, or do we pass it to TPU's file I/O routines?
!
    IF (..p_opcode NEQ tpu$k_open) AND (.stream [stream_file_id] GTR 511)
    THEN
        RETURN tpu$fileio (.p_opcode, .stream, .data);

!
! Either we're opening the file, or we know it's one of ours
! Call the appropriate routine (not shown in this example)
!
    SELECTONE ..p_opcode OF
        SET

        [tpu$k_open]:
            status = my_io_open (.stream, .data);

        [tpu$k_close, tpu$k_close_delete]:
            status = my_io_close (.stream, .data);

        [tpu$k_get]:
            status = my_io_get_record (.stream, .data);

        [tpu$k_put]:
            status = my_io_put_record (.stream, .data);
```

(continued on next page)

Example 5–1 (Cont.) Sample VAX BLISS Template for Callable VAXTPU

```
        [OTHERWISE]:
            status = tpu$_failure;

        TES;

    RETURN .status;

    END;                                        ! End of routine TPU_IO

END                                             ! End Module file_io_example

ELUDOM
```

In Chapter 13 of the *VMS Utility Routines Manual*, Example 13-2 does not work. Example 5–2 should be substituted:

Example 5–2 Building a Callback Item List with VAX FORTRAN

```
       PROGRAM  TEST_TPU
C
       IMPLICIT NONE
C
C       Define the expected VAXTPU return statuses
C
       EXTERNAL        TPU$_SUCCESS
       EXTERNAL        TPU$_QUITTING
       EXTERNAL        TPU$_EXITING
C
C       Declare the VAXTPU routines and symbols used
C
       EXTERNAL        TPU$M_DELETE_CONTEXT
       EXTERNAL        TPU$HANDLER
       INTEGER*4       TPU$M_DELETE_CONTEXT
       INTEGER*4       TPU$INITIALIZE
       INTEGER*4       TPU$EXECUTE_INIFILE
       INTEGER*4       TPU$CONTROL
       INTEGER*4       TPU$CLEANUP
C
C      Use LIB$MATCH_COND to compare condition codes
C
       INTEGER*4       LIB$MATCH_COND
C
C       Declare the external callback routine
C
       EXTERNAL        TPU_STARTUP       ! the VAXTPU set-up function
       INTEGER*4       TPU_STARTUP

       INTEGER*4       BPV(2)            ! Set up a bound procedure value
C
C      Declare the functions used for working with the condition handler
C
       INTEGER*4       LIB$ESTABLISH
       INTEGER*4       LIB$REVERT
```

(continued on next page)

Example 5–2 (Cont.) Building a Callback Item List with VAX FORTRAN

```
C
C       Local Flags and Indices
C
        INTEGER*4       CLEANUP_FLAG       ! flag(s) for VAXTPU cleanup
        INTEGER*4       RET_STATUS
        INTEGER*4       MATCH_STATUS
C
C       Initializations
C
        RET_STATUS       = 0
        CLEANUP_FLAG     = %LOC(TPU$M_DELETE_CONTEXT)
C
C       Establish the default VAXTPU condition handler
C
        CALL LIB$ESTABLISH(%REF(TPU$HANDLER))
C
C       Set up the bound procedure value for the initialization callback
C
        BPV(1)  =  %LOC (TPU_STARTUP)
        BPV(2)  =  0
C
C       Call the VAXTPU procedure for initialization
C
        RET_STATUS = TPU$INITIALIZE(BPV)

        IF (RET_STATUS .NE. %LOC(TPU$_SUCCESS)) THEN
        CALL LIB$SIGNAL (%VAL(RET_STATUS))
        ENDIF
C
C       Execute the VAXTPU initialization file
C
        RET_STATUS = TPU$EXECUTE_INIFILE()

        IF (RET_STATUS .NE. %LOC(TPU$_SUCCESS)) THEN
        CALL LIB$SIGNAL (%VAL(RET_STATUS))
        ENDIF
C
C       Pass control to VAXTPU
C
        RET_STATUS = TPU$CONTROL()
C
C       Test for valid exit condition codes.  You must use LIB$MATCH_COND
C       because the severity of TPU$_QUITTING can be set by the TPU
C       application
C
        MATCH_STATUS = LIB$MATCH_COND (RET_STATUS, %LOC (TPU$_QUITTING),
        1                                          %LOC (TPU$_EXITING))
        IF (MATCH_STATUS .EQ. 0) THEN
        CALL LIB$SIGNAL (%VAL(RET_STATUS))
        ENDIF
C
C       Clean up after processing
C
        RET_STATUS = TPU$CLEANUP(%REF(CLEANUP_FLAG))

        IF (RET_STATUS .NE. %LOC(TPU$_SUCCESS)) THEN
        CALL LIB$SIGNAL (%VAL(RET_STATUS))
        ENDIF
```

(continued on next page)

Example 5–2 (Cont.) Building a Callback Item List with VAX FORTRAN

```
C
C       Set the condition handler back to the default
C
        RET_STATUS = LIB$REVERT()
        END

        INTEGER*4 FUNCTION TPU_STARTUP
        IMPLICIT NONE
        INTEGER*4       OPTION_MASK       ! temporary variable for VAXTPU
        CHARACTER*44       SECTION_NAME   ! temporary variable for VAXTPU
C
C       External VAXTPU routines and symbols
C
        EXTERNAL        TPU$K_OPTIONS
        EXTERNAL        TPU$M_READ
        EXTERNAL        TPU$M_SECTION
        EXTERNAL        TPU$M_DISPLAY
        EXTERNAL        TPU$K_SECTIONFILE
        EXTERNAL        TPU$K_FILEIO
        EXTERNAL        TPU$FILEIO
        INTEGER*4       TPU$FILEIO
C
C       The bound procedure value used for setting up the file I/O routine
C
        INTEGER*4       BPV(2)

C
C        Define the structure of the item list defined for the callback
C
        STRUCTURE /CALLBACK/
        INTEGER*2       BUFFER_LENGTH
        INTEGER*2       ITEM_CODE
        INTEGER*4       BUFFER_ADDRESS
        INTEGER*4       RETURN_ADDRESS
        END STRUCTURE
C
C       There are a total of four items in the item list
C
        RECORD /CALLBACK/ CALLBACK (4)
C
C       Make sure it is not optimized!
C
        VOLATILE /CALLBACK/
C
C        Define the options we want to use in the VAXTPU session
C
        OPTION_MASK = %LOC(TPU$M_SECTION) .OR. %LOC(TPU$M_READ)
        1       .OR. %LOC(TPU$M_DISPLAY)
C
C       Define the name of the initialization section file
C
        SECTION_NAME = 'TPU$SECTION'
```

(continued on next page)

Example 5–2 (Cont.) Building a Callback Item List with VAX FORTRAN

```
C
C       Set up the required I/O routine.  Use the VAXTPU default.
C
        BPV(1) = %LOC(TPU$FILEIO)
        BPV(2) = 0
C
C       Build the callback item list
C
C       Set up the edit session options
C
        CALLBACK(1).ITEM_CODE = %LOC(TPU$K_OPTIONS)
        CALLBACK(1).BUFFER_ADDRESS = %LOC(OPTION_MASK)
        CALLBACK(1).BUFFER_LENGTH = 4
        CALLBACK(1).RETURN_ADDRESS = 0
C
C       Identify the section file to be used
C
        CALLBACK(2).ITEM_CODE = %LOC(TPU$K_SECTIONFILE)
        CALLBACK(2).BUFFER_ADDRESS = %LOC(SECTION_NAME)
        CALLBACK(2).BUFFER_LENGTH = LEN(SECTION_NAME)
        CALLBACK(2).RETURN_ADDRESS = 0
C
C       Set up the I/O handler
C
        CALLBACK(3).ITEM_CODE = %LOC(TPU$K_FILEIO)
        CALLBACK(3).BUFFER_ADDRESS = %LOC(BPV)
        CALLBACK(3).BUFFER_LENGTH = 4
        CALLBACK(3).RETURN_ADDRESS = 0
C
C       End the item list with zeros to indicate we are finished
C
        CALLBACK(4).ITEM_CODE = 0
        CALLBACK(4).BUFFER_ADDRESS = 0
        CALLBACK(4).BUFFER_LENGTH = 0
        CALLBACK(4).RETURN_ADDRESS = 0
C
C       Return the address of the item list
C
        TPU_STARTUP = %LOC(CALLBACK)

        RETURN
        END
```

5.34 *VMS VAXcluster Manual*

The following sections contain information concerning the *VMS VAXcluster Manual*.

5.34.1 CLUSTER_CONFIG.COM

V5.0

Because CLUSTER_CONFIG.COM no longer sets EXPECTED_VOTES, text describing the ADD function near the beginning of Examples 3-1 and 3-2 should read as follows:

The ADD function adds a new node to the cluster.

If the node being added is a voting member, EXPECTED_VOTES in every cluster member's MODPARAMS.DAT must be adjusted, and the cluster must be rebooted.

Additionally, the EXPECTED_VOTES line at the end of Example 3-1 should be ignored and parameter settings should be shown as follows:

```
The following parameters have been set for SATURN:

                    VOTES = 1
                    QDSKVOTES = 1
```

The first sentence in Section 3.3.1 should read as follows:

Whenever you add or remove a voting cluster node, or when you enable or disable a quorum disk, you must edit MODPARAMS.DAT in every cluster member's SYSx.SYSEXE directory and adjust the value for the SYSGEN parameter EXPECTED_VOTES appropriately.

5.34.2 MOUNT/CLUSTER Command

V5.0-1

The following command example in Section 5.5.1 of the *VMS VAXcluster Manual* is incorrect:

```
$
```

The correct command is as follows:

```
$
```

A future revision of the *VMS VAXcluster Manual* will contain this correction.

5.35 *VMS Developer's Guide to VMSINSTAL*

V5.0-2

Developers of installation procedures using the VMSINSTAL callback SET STARTUP can specify only a file name for P3. The documentation incorrectly states that developers can specify SYSMAN for P3.

A DECwindows Ada Programming Interfaces

V5.1

DECwindows provides programming interface definitions for the Ada language. When you select Ada support at the time of the DECwindows kit installation, four Ada package source files are placed in the SYS$LIBRARY: directory of your system. These files are:

- CDA$CDA_.ADA: Package CDA—Compound Document Architecture
- DDIF$DDIF_.ADA: Package DDIF—Digital Document Interchange Format
- DECW$DWT_.ADA: Package DWT—DECwindows Toolkit
- DECW$X_.ADA: Package X—Xlib

These package source files can be individually compiled into your Ada program libraries or compiled into the systemwide Ada predefined library. To make the packages available systemwide, the command file SYS$UPDATE:DECW$COMPILE_ADA_UNITS.COM is provided.

This command procedure compiles all four packages into the predefined Ada library, and, if the VAX Source Code Analyzer (SCA) product is present, loads SCA analysis data for the packages into the SCA library for the predefined library. The command procedure should be run as a batch job and should have available a minimum of 2000 pages in the working set; however, 3000 pages is preferable. A page file quota of at least 30000 pages is suggested.

Once the units are compiled into the predefined Ada library, you must execute the following Ada program library manager command to make the units visible:

```
$
```

You need only to do this once. This step is also performed automatically for all Ada program libraries created after the DECwindows units are compiled into the predefined library.

Future installations of DECwindows might replace the Ada packages. If so, the new packages must be compiled as shown. If you have already entered the units into your own library, you must then execute the following command to make your library current:

```
$
```

Future installations of VAX Ada might replace the Ada predefined library and remove the DECwindows units. If this occurs, reexecute the DECW$COMPILE_ADA_UNITS.COM command procedure.

If you want to compile the units into your program libraries directly, you must execute the following commands after compilation of packages DWT and X:

```
$
$
```

This step is not necessary if the units are entered from the predefined library.

Once the units are entered into your program library, applications that use the DECwindows packages are linked in the normal manner using ACS LINK. It is not necessary to explicitly specify the shareable images when linking.

A.1 Using the Ada Packages

Each package, CDA, DDIF, DWT, and X, contains definitions of constants, structures, status codes, and routines for each facility. All four packages observe certain common conventions for naming and use; these conventions are outlined as follows:

- In each package, the facility prefix (CDA_, DDIF_, DWT_, X_) has been removed from all the symbols defined in that package. It is intended that the Ada USE clause not be used with these packages. This encourages clarity in the application source and also improves compiler efficiency. For example:

```
with DWT;
procedure CALLBACK (
WIDGET: in DWT.WIDGET_TYPE) is
...
ARGLIST: DWT.ARG_ARRAY_TYPE (0..0);
CSTRING: DWT.COMP_STRING_TYPE;
...
begin
DWT.LATIN1_STRING (.....);
```

  In some packages, symbols defined with other facility prefixes are present; these have not been removed from the symbol names. For example, routine XT$INITIALIZE is DWT.XT_INITIALIZE.

- When a symbol would conflict with an Ada reserved word or predefined identifier, the last letter of the symbol name is removed. For example, the routine DWT$STRING is DWT.STRIN. See the individual package descriptions for a list of affected identifiers.
- Unconstrained array types are defined as name_ARRAY_TYPE for an array of name_TYPE elements. The DECwindows documentation sometimes uses name_LIST for such arrays; in the Ada packages, these names are used when the address of an array is desired, most commonly as an element of a structure.
- All functions are defined as "valued procedures." The function return value is usually named STATUS or RESULT, depending on the type of value returned.
- The null-terminated strings required by some procedures can be created by concatenating the string with ASCII.NUL. Further information about the interfaces can be found by examining the package sources provided in SYS$LIBRARY:, as described above.

Usage information for the specific packages is given in following sections.

A.1.1 Package CDA

This package defines constants and types for the Compound Document Architecture facility. There are no package-specific usage comments for package CDA.

A.1.2 Package DDIF

This package defines constants and types for the Digital Document Interchange Format facility. There are no package-specific usage comments for package DDIF.

A.1.3 Package DWT

This package defines constants, types, and procedures for the XUI Toolkit facility. The following usage comments are specific to package DWT:

- The procedure STRING is renamed STRIN to avoid conflict with the predefined type.
- The parameter ADDRESS of procedure XT_FREE is renamed ADDRES to avoid conflict with the predefined type.
- The subtype definitions shown in Table A–1 that rename types from package DWT are provided:

Table A–1 Subtype Definitions—Package DWT

Subtype	Definition
DISPLAY_TYPE	X.DISPLAY_TYPE
EVENT_TYPE	X.EVENT_TYPE
GC_TYPE	X.GC_ID_TYPE
PIXMAP_TYPE	X.PIXMAP_ID_TYPE
TIME_TYPE	X.TIME_TYPE
SCREEN_TYPE	X.SCREEN_ID_TYPE
WINDOW_TYPE	X.WINDOW_ID_TYPE
XRMDATABASE_TYPE	X.DATABASE_ID_TYPE

- The types INTEGER_ARRAY and ADDRESS_ARRAY are defined for use with procedures in package DWT, being unconstrained arrays of INTEGER and ADDRESS, respectively.
- The type DESCRIPTOR_TYPE is defined for constructing string descriptors required by certain procedures.

A.1.4 Package X

This package defines types, structures, and procedures for the Xlib facility. The following usage comments are specific to package X:

- The subtype definitions listed in Table A–2 are provided.

Table A–2 Subtype Definitions—Package X

Subtype	Definition
ATOM_ID_TYPE	SYSTEM.UNSIGNED_LONGWORD
BITMAP_ID_TYPE	SYSTEM.UNSIGNED_LONGWORD
CLASS_LIST_ID_ TYPE	SYSTEM.UNSIGNED_LONGWORD
COLORMAP_ID_ TYPE	SYSTEM.UNSIGNED_LONGWORD
CURSOR_ID_TYPE	SYSTEM.UNSIGNED_LONGWORD
DATABASE_ID_TYPE	SYSTEM.UNSIGNED_LONGWORD
DISPLAY_ID_TYPE	SYSTEM.ADDRESS
DISPLAY_TYPE	SYSTEM.ADDRESS
DRAWABLE_ID_ TYPE	SYSTEM.UNSIGNED_LONGWORD
FONT_ID_TYPE	SYSTEM.UNSIGNED_LONGWORD
GC_ID_TYPE	SYSTEM.UNSIGNED_LONGWORD
KEYSYM_ID_TYPE	SYSTEM.UNSIGNED_LONGWORD
NAME_LIST_ID_ TYPE	SYSTEM.UNSIGNED_LONGWORD
PIXMAP_ID_TYPE	SYSTEM.UNSIGNED_LONGWORD
PROPERTY_ID_ TYPE	SYSTEM.UNSIGNED_LONGWORD
REGION_ID_TYPE	SYSTEM.UNSIGNED_LONGWORD
SEARCH_LIST_ID_ TYPE	SYSTEM.UNSIGNED_LONGWORD
SELECTION_ID_ TYPE	SYSTEM.UNSIGNED_LONGWORD
SCREEN_ID_TYPE	SYSTEM.UNSIGNED_LONGWORD
TARGET_ID_TYPE	SYSTEM.UNSIGNED_LONGWORD
TIME_TYPE	SYSTEM.UNSIGNED_LONGWORD
TYPE_ID_TYPE	SYSTEM.UNSIGNED_LONGWORD
WINDOW_ID_TYPE	SYSTEM.UNSIGNED_LONGWORD

- The argument EVENT_TYPE of procedures CHECK_TYPED_EVENT and CHECK_TYPED_WINDOW_EVENT has been renamed to EVENT_TYP to avoid conflict with the EVENT_TYPE type definition.

- EVENT_TYPE has been defined as a variant record; subtypes for specific event types are also defined as specific instances of the variant record. The discriminant for EVENT_TYPE is the field EVNT_TYPE; each variant uses its own prefixes for the field names, for example, KYEV_DISPLAY for a key event.

  When you declare a variable as being type EVENT_TYPE, Ada automatically allocates the maximum possible event size for the variable. When examining event variables, be sure to use only the correct fields for the variant defined by EVNT_TYPE; otherwise an Ada constraint error can be generated. For example, the following code is correct:

```
if EVENT.EVNT_TYPE = X.C_EXPOSE then
if EVENT.EXEV_WINDOW = WINDOW_2 then
 .
 .
 .
```

  while the following is incorrect:

```
if EVENT.EVNT_TYPE = X.C_EXPOSE and
EVENT.EXEV_WINDOW = WINDOW_2 then
 .
 .
 .
 .
```

  The second code fragment would raise a constraint error on the reference to EVENT.EXEV_WINDOW if the value of the discriminant (EVNT_TYPE) was not C_EXPOSE.

A.2 Callbacks

Ada procedures that are to be used as callback routines must be made visible by means of the EXPORT_PROCEDURE pragma. This requires that the procedure be a library unit or be declared in the outermost declarative part of a library package. See the section on exporting subprograms in the *VAX Ada Language Reference Manual* for more details.

Be aware that EXPORT_PROCEDURE implicitly declares the procedure name as a global symbol. If the same procedure name is used in multiple packages, you should specify an "external designator" as the second argument of pragma EXPORT_PROCEDURE to give the procedure a unique external name.

Callback routines used in tasking applications must also specify pragma SUPPRESS_ALL. This suppresses the task stack check that might fail for routines called from outside the context of an Ada task.

A.3 Tasking Considerations

Ada programs that use tasking can call DECwindows routines, but applications designers should be aware that the DECwindows design philosophy is oriented towards event polling and not asynchronous notification of events. A mechanism is available to queue an AST when certain events occur, but this is specific to VMS and should be used cautiously in applications intended to be portable.

An important consideration is that the routines that wait for an event, such as X.NEXT_EVENT, block the process until the event occurs. In a tasking program, this means that all tasks are blocked, even if a task of a higher priority is eligible for execution. However, if time slicing is enabled with pragma TIME_SLICE, other tasks of equal priority will run at the end of each time slice; but when the stalled task is again scheduled, it will block until its time slice has expired. Tasks of lower priority will not run. However, tasks of higher priority that become runnable by means of an AST completion (such as by using one of the routines in package TASKING_SERVICES) will run immediately.

A.4 Examples

There are three Ada language examples provided in the DECW$EXAMPLES: directory:

1. HELLOWORLD.ADA is a simple example of using the DECwindows Toolkit and Resource Manager.
2. DECBURGER.ADA is a more complex example of using various predefined widgets in the DECwindows Toolkit and demonstrates the use of callbacks, as well as more intensive use of the Resource Manager and access to UIL definitions.
3. XLIBINTRO.ADA demonstrates the use of the Xlib interface and responding to events.

The first two example programs require that the appropriate UIL file from the directory DECW$EXAMPLES be compiled using the UIL compiler before running the programs. See the command procedure DECW$EXAMPLES:DEMO_BUILD.COM for details on compiling and linking the example applications.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-DEC-DEMO (800-332-3366) using a 1200- or 2400-baud modem. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local DIGITAL subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	————	Local DIGITAL subsidiary or approved distributor
Internal[1]	————	SDC Order Processing - WMO/E15 *or* Software Distribution Center Digital Equipment Corporation Westminster, Massachusetts 01473

[1]For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

Reader's Comments

VMS Version 5.2 Release Notes
AA–LB22B-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

_____

What I like best about this manual is _____

_____

_____

What I like least about this manual is _____

_____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

--- 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

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

--- Do Not Tear - Fold Here ---

Reader's Comments

VMS Version 5.2 Release Notes
AA–LB22B-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

_____

What I like best about this manual is _____

_____

_____

What I like least about this manual is _____

_____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

--- 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

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

--- Do Not Tear - Fold Here ---