VMS Version 5.0 Release Notes

Order Number: AA–LB22A–TE

April 1988

This document describes changed software features, problems and restrictions to software, changes to documentation, and upgrade information for VMS Version 5.0.

Revision/Update Information:	This manual supersedes previous VAX/VMS Release Notes.
Operating System and Version:	VMS Version 5.0
Software Version:	VMS Version 5.0

digital equipment corporation
maynard, massachusetts

April 1988

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

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem–10	PDP	VT
DECSYSTEM–20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital™

ZK4524

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire
03061

CANADA

Digital Equipment
of Canada Ltd.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

Digital Equipment Corporation
PSG Business Manager
c/o Digital's local subsidiary
or approved distributor

In Continental USA, Alaska, and Hawaii call 800-DIGITAL.
In Canada call 800-267-6215.
* Any order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).
Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

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.

® PostScript is a registered trademark of Adobe Systems, Inc.

Contents

PREFACE xxiii

CHAPTER 1 UPGRADING TO VMS VERSION 5.0 1–1

1.1 UPGRADE SUMMARY 1–2

1.2 UPGRADE CONSIDERATIONS 1–2

1.3 LAYERED PRODUCTS CAUTION 1–4

CHAPTER 2 PRE-UPGRADE PROCEDURE 2–1

CHAPTER 3 UPGRADE PROCEDURE FOR A SINGLE SYSTEM 3–1

3.1 BEGINNING THE UPGRADE PROCEDURE 3–1

3.2 UPGRADE PHASE 1 3–4
3.2.1 Upgrade Phase 1—for VAX–11/750 Computer 3–4
3.2.2 Upgrade Phase 1—for VAX 8200, 8250, 8300, and 8350 Computers 3–6
3.2.3 Upgrade Phase 1—for VAX 8530, 8550, 8700, or 8800 Computers 3–8
3.2.4 Upgrade Phase 1—for VAX–11/725, VAX–11/730, VAX–11/780, VAX–11/785, VAX 8600, and VAX 8650 Computers 3–10
3.2.5 Upgrade Phase 1—for MicroVAX and VAXstation Computers 3–11

3.3 UPGRADE PHASE 2 3–12

3.4 UPGRADE PHASE 3 3–13

3.5	UPGRADE PHASE 4	3–13
3.5.1	Upgrade Phase 4—VAX 8530, 8550, 8700, 8800 Computers	3–14
3.5.2	Upgrade Phase 4—for MicroVAX and VAXstation Computers	3–14
3.6	UPGRADE PHASE 5	3–15
3.7	UPGRADE PHASE 6	3–15
CHAPTER 4	UPGRADING CLUSTERS	4–1
4.1	UPGRADING A VAXCLUSTER: CONCURRENT UPGRADE	4–1
4.2	UPGRADING A VAXCLUSTER: ROLLING UPGRADE	4–2
4.2.1	VMS Version 5.0 Rolling Upgrade Procedure—Overview	4–3
4.2.2	Limitations of Mixed-Version Clusters	4–4
4.3	PERFORMING THE VERSION 5.0 ROLLING UPGRADE	4–6
4.3.1	Upgrading the Cluster to A5.0	4–7
4.3.2	Converting the Cluster to B5.0	4–9
4.3.3	Converting the Cluster to VMS Version 5.0	4–10
CHAPTER 5	POST-UPGRADE PROCEDURES	5–1
CHAPTER 6	LICENSE MANAGEMENT	6–1
6.1	REGISTERING YOUR LICENSES	6–1
6.2	REGISTERING A LICENSE WITH VMSLICENSE.COM	6–3
6.3	REGISTERING A VMS LICENSE IN A VAXCLUSTER ENVIRONMENT	6–7

6.4	REGISTERING A VMS LICENSE ON A STANDALONE VAX	6–13
6.5	REGISTERING A SYSTEM INTEGRATED PRODUCT LICENSE	6–19
6.6	MANAGING VMS W-KIT LICENSES FOR SERVICE CUSTOMERS	6–24
6.7	MODIFYING LICENSE UNITS WITH THE LICENSE MANAGEMENT UTILITY	6–25
6.8	LICENSE MANAGEMENT FACILITY (LMF) NOTES	6–28
6.9	VMS LICENSE TYPES	6–29
6.10	LICENSE ACTIVITY USE DEFINITION FOR VMS LICENSES	6–30
6.11	VAXCLUSTER LICENSE NOTES	6–30
6.12	DECNET–VAX LICENSE NOTES	6–31
6.13	VAX RMS JOURNALING LICENSE NOTES	6–31
6.14	VAX VOLUME SHADOWING LICENSE NOTES	6–31
6.15	NEW DCL LICENSE COMMAND	6–31
6.16	NEW DCL SHOW LICENSE COMMAND	6–32
6.17	LICENSE LIST COMMAND PROBLEM	6–33

CHAPTER 7	**GENERAL USER RELEASE NOTES**	**7–1**
7.1	DIGITAL COMMAND LANGUAGE (DCL) — NOTES	7–1
7.1.1	DCL Command DEFINE/FORM Qualifier /SHEET_FEED — Restriction	7–1
7.1.2	DCL Command Statement IF Level Clarification	7–1
7.1.3	DCL Command OPEN — Problem	7–1
7.1.4	DCL Command RENAME Requires Delete Access	7–2
7.1.5	DCL Command SET PROCESS/CPU=[NO]ATTACHED Removed	7–2
7.1.6	Using the DCL Command SET TIME	7–2
7.1.7	Using the DCL Command SET TIME/CLUSTER	7–2
7.1.8	DCL Command SHOW QUEUE/SUMMARY Displays Incorrect Job Counts	7–2
7.1.9	Obsolete Commands	7–3
7.2	EDT — PROBLEM	7–3
7.3	EVE — NOTES	7–3
7.3.1	Incompatibilities with Previous Versions	7–4
7.3.2	Incompatibility with Future Versions	7–9
7.3.3	Problems in EVE	7–10
7.3.4	Restrictions in EVE	7–14
7.3.5	Problems in EVE Documentation	7–15
7.4	COMMAND PROCEDURES — RESTRICTION	7–15
7.5	EXTENDED FILE NAMES OR FILE TYPES — CAUTION	7–16
7.6	PRIMARY AS OPPOSED TO ALIAS DIRECTORY ENTRIES	7–17
7.7	FILE DEFINITION LANGUAGE FACILITY — RESTRICTION	7–17
7.8	SYMBOL NAMES — CAUTION	7–18
7.9	SYMBOL SUBSTITUTION FOR F$VERIFY LEXICAL FUNCTION	7–18

CHAPTER 8 SYSTEM MANAGER RELEASE NOTES 8–1

8.1 INSTALLATION AND UPGRADE INFORMATION 8–1
8.1.1 Installing VMS on a VAX 8200 from an HSC Disk Drive 8–1
8.1.1.1 How to Proceed • 8–1
8.1.1.2 Installing the VMS Operating System • 8–1
8.1.1.3 Editing CIBOO.CMD • 8–3
8.1.2 Tape Device Names for MicroVAX, VAXstation, and VAXserver 3600 Series Computers Installations 8–5
8.1.2.1 Device Names for Tape Cartridge and Magnetic Tape Drives • 8–5
8.1.3 RK07 Installation Distribution Kit 8–5
8.1.4 Error Message Displayed During AUTOGEN 8–6
8.1.5 Upgrade Procedure Driver Version Mismatch Errors 8–6
8.1.6 ANALYZE/ERROR_LOG Restriction for Rolling Upgrades 8–6

8.2 ASYMMETRIC MULTIPROCESSING (ASMP) AND VAX–11/782 — SUPPORT DISCONTINUED 8–6

8.3 SYSTEM STARTUP PROCEDURE — NOTES 8–7
8.3.1 The STARTUP.COM Command Procedure 8–7
8.3.2 The SYS$STARTUP Directory 8–8
8.3.3 The Startup Data Files 8–8
8.3.4 New Name for the Site-Specific Startup Procedure 8–9
8.3.5 SYSMAN Utility STARTUP Subfunction 8–9

8.4 SYENVIRON.COM OBSOLETE 8–10

8.5 AUTHORIZE UTILITY — NOTES 8–10
8.5.1 Network Proxy Authorization File Changes 8–10
8.5.2 Changed UAF Limits for SYSTEM and DEFAULT Records 8–11

8.6 AUTOGEN COMMAND PROCEDURE — NOTES 8–12
8.6.1 AUTOGEN Aborts During GENFILES Phase 8–12
8.6.2 Feedback Mechanism Added 8–13
8.6.3 Additional Data Files Used with Feedback Mechanism 8–13
8.6.4 Oldsite Mechanism Is Obsolete 8–13
8.6.5 Mechanism to Control MSCP Server Buffer Size Is Obsolete 8–14
8.6.6 System Files Not Marked for NOBACKUP 8–14
8.6.7 Swap File Size Changes 8–15
8.6.8 Dump File Size Changes 8–15
8.6.9 Selective Crash Dump Files — Caution 8–15

8.6.10	Interaction Between the SAVEDUMP Parameter and PAGEFILE.SYS Size — Caution	8–16
8.6.11	Hexadecimal Values Processed Correctly in MODPARAMS.DAT	8–16
8.7	ADDITIONAL PRIVILEGE REQUIRED TO EXECUTE STABACKIT.COM	8–16
8.8	BATCH JOBS SUBMITTED USING THE DCL COMMAND SUBMIT/USER	8–16
8.9	CIBCA — NEW DEVICE SUPPORT	8–17
8.10	SUPPORTED COMPUTER INTERCONNECT PORT MICROCODE	8–17
8.11	DEBUGGER — MAKING IMAGES SHAREABLE FOR BETTER PERFORMANCE	8–17
8.12	DEBNA VAXBI ETHERNET CONTROLLER — NOTES	8–18
8.12.1	Tuning the VMS Operating System for DEBNA Controllers	8–18
8.12.2	Configuration and Startup	8–19
8.13	DECNET–VAX NOTES	8–19
8.13.1	Support for X.25 Virtual Circuits Requirement	8–19
8.13.2	Constraints on Passive Maintenance Functions Relaxed	8–19
8.13.3	Proxy Access Parameters — Changes	8–20
8.13.4	MAXIMUM PATH SPLITS Default Value	8–20
8.14	DEQNA ETHERNET ADAPTER MAY RECEIVE CORRUPT DATA	8–20
8.15	DMB32 PRODUCT SOFTWARE REQUIRED FOR DMB32 COMMUNICATIONS CONTROLLER	8–20
8.16	DIGITAL STORAGE ARCHITECTURE REQUIREMENT FOR DISK PATH FAILOVER	8–20

8.17	HIGHWATER MARKING ON SYSTEM VOLUMES MAY CAUSE SYSTEM FAILURES	8–21
8.18	FORCED ERROR HANDLING	8–22
8.19	INITIALIZE COMMAND — DEFINING VOLUME SERIAL NUMBERS	8–22
8.20	LAT SOFTWARE — NOTES	8–23
8.20.1	Delay in Process Disconnect	8–23
8.20.2	Solicit Connection QIO	8–23
8.20.3	LAT PASSALL Session	8–23
8.20.4	LAT Control Program (LATCP) Changes and Restrictions	8–23
8.21	LIBRARIAN ROUTINES — CAUTION WHEN USING LOCATE MODE	8–24
8.22	MODEM SIGNAL REQUIREMENTS NOW ENFORCED	8–24
8.23	MONITOR UTILITY — NOTES	8–24
8.23.1	MONITOR Display Error	8–24
8.23.2	MONITOR RMS Bucket and Multibucket Split Rates Invalid	8–25
8.24	MOUNT UTILITY — NOTES	8–25
8.24.1	MOUNT Qualifier /MULTI_VOLUME — Requirement	8–25
8.24.2	Changes to Mount Verification	8–25
8.25	MODULAR EXECUTIVE — NOTES	8–26
8.25.1	Introduction to the Modular Executive	8–26
8.25.2	Effects on Privileged Code	8–27
8.25.3	Version Numbers and Version Checking	8–27
8.26	MODULAR EXECUTIVE — EFFECTS UPON SYSTEM MANAGEMENT	8–28
8.26.1	SYS$LOADABLE_IMAGES Directory on the System Disk	8–28
8.26.2	Version Checking	8–28
8.26.3	System Failure	8–29

8.27	MODULAR EXECUTIVE — EFFECTS UPON SYSGEN	8–29
8.28	MSCP SERVER AND DISKETTE DEVICES	8–29
8.29	NETWORK CONTROL PROGRAM — CHANGES	8–29
8.30	RESTRICTION FOR RQDX3 CONTROLLERS	8–30
8.31	DEVICE UNIT NUMBER CHANGED WITH RQDX3 CONTROLLERS	8–30
8.32	VAX RMS JOURNALING — NOTES	8–30
8.32.1	STREAM Formats not Supported for Shared Sequential Files	8–31
8.32.2	CONTROL Access Required to Modify a File Within a Recovery Unit	8–31
8.32.3	New File Access Block Field for VAX RMS Journaling (FAB$B_JOURNAL)	8–31
8.32.4	Running the VAX RMS Journaling Installation Verification Procedure	8–32
8.32.5	CONTROL Access Required for After-Image or Before-Image Recovery	8–34
8.32.6	Additional VAX RMS Journaling Error Messages	8–34
8.32.7	Backup Utility Errors	8–36
8.32.8	The Backup Utility and the SET FILE Command	8–36
8.32.9	Errors During the Execution of Recovery Unit Service	8–37
8.32.10	WRTJNL_BIJ Error Message	8–37
8.32.11	Files DIGITAL Recommends Not Marking for Journaling	8–37
8.32.12	Handling RMS I/O Errors when Journaling	8–38
8.32.13	Accessing Indexed Files on Systems Without VAX RMS Journaling Installed	8–38
8.32.14	Restriction on Exclusive Access to Recovery Unit Journaled Files	8–39
8.32.15	SET FILE/AI_JOURNAL or SET FILE/BI_JOURNAL Command	8–39
8.32.16	VFC Format Sequential Files Partially Supported for Before-Image or Recovery Unit Journaling	8–39
8.32.17	Mount Utility Creates the Logical Name DISK$volume_label	8–40
8.33	SYSTEM BOOTSTRAP — HANDLING OF THE SPECIAL FILES	8–40

8.34	SHOW CLUSTER UTILITY — NOTES	8–41
8.34.1	HW_TYPE Field Changes	8–41
8.34.2	WRITE/ALL Method for Determining Page Size	8–42
8.35	NEW FIELD IN SHOW MEMORY DISPLAY—RESERVABLE PAGES	8–42
8.36	SYSGEN — PARAMETER CHANGES	8–43
8.36.1	SYSGEN Parameter RECNXINTERVAL	8–43
8.36.2	SYSGEN Parameter MULTIPROCESSING Default Value	8–44
8.37	SET TIME/CLUSTER COMMAND SUPERSEDED	8–44
8.38	SYSTEM MANAGEMENT UTILITY (SYSMAN) — NOTES	8–44
8.39	SYS$LOADABLE_IMAGES [SYS$LDR] DIRECTORY	8–45
8.40	TAILORING PROGRAM CHANGED	8–47
8.41	QUORUM DISKS — NOTES	8–47
8.41.1	New Configuration Support	8–47
8.41.2	Disk Behavior	8–48
8.42	UIS WORKSTATIONS — SETTING OF MULTIPROCESSING SYSGEN PARAMETER	8–48
8.43	SET HOST/DTE COMMAND CAUSES SYSTEM FAILURES ON WORKSTATIONS	8–48
8.44	UETP CHANGES	8–49
8.44.1	SYSUAF Quotas for SYSTEST and SYSTEST_CLIG Accounts	8–49
8.44.2	Support for Q-bus Devices	8–49
8.44.3	Cluster Integration Test Phase	8–49
8.44.4	Device Support Removed	8–49
8.45	INSTALLING A BUA BOARD ON A VAX 8820, 8830, OR 8840	8–50

8.46	STOPPING A CPU ON A VAX 8530, 8550, 8700, 8800, 8820, 8830, AND 8840	8–50
8.47	UNSYNCHRONIZED CLUSTER TIME AFFECTS SUBMIT/AFTER COMMAND	8–50
8.48	VAXCLUSTER ETHERNET ADAPTER RESTRICTION	8–51
8.49	UNIBUS DEVICES ON VAX 8800 SYSTEMS RUNNING SMP — KNOWN PROBLEM	8–51
8.50	UNIBUS FLOATING INTERRUPT VECTOR CHANGE	8–51
8.51	REDUCTION IN VAXCLUSTER STATE TRANSITION TIME	8–53
8.52	COMMAND PROCEDURE FOR THE VAX–11/750	8–54
8.53	UNDOCUMENTED NEW VAX COMPUTERS	8–56
8.54	VAX 6210/6220 SYSTEMS — LP11 PRINTER NOT SUPPORTED WHEN STANDALONE BACKUP IS BOOTED FROM A TK50	8–56
8.55	VAX 8800 — STOP/CPU COMMAND FAILS	8–56
8.56	LARGE SYSTEM BOOTSTRAP FAILURE	8–57
8.57	COLOR VAXSTATIONS IN CLUSTERED ENVIRONMENTS	8–57
8.58	VAXSTATION 2000/MICROVAX 2000 HARD DISK GEOMETRY CHANGE	8–58
8.59	VAX 6200 SERIES COMPUTER SYSTEMS	8–58
8.60	VAX 8820, VAX 8830, AND VAX 8840 COMPUTER SYSTEMS	8–58

8.61	RESTRICTION ON A VAXBI 5	8–59
8.62	HALTING A VAX 8530, 8550, 8700, 8800	8–59
8.63	FEWER CLUSTER MESSAGES DISPLAYED ON THE CONSOLE TERMINAL AND SENT TO OPCOM	8–59
8.64	BOOTING A SATELLITE NODE (CLUSTER_CONFIG.COM ADD PHASE)	8–60
8.65	REBOOTING A SATELLITE NODE WITH AN OPERATING SYSTEM ON A LOCAL DISK	8–60
8.66	VMS BOOTSTRAPPING PROBLEM AND SOLUTION	8–60
8.67	SHUTDOWN NOTIFICATION ON CLUSTERS — NOTE	8–61
8.68	VMSINSTAL OPTION N — COMPATIBILITY PROBLEM	8–61
8.69	VMS/MICROVMS — NOTES RELATED TO MERGE	8–62
8.69.1	Format and Installation Procedures	8–62
8.69.2	ERRFMT and OPCOM	8–62
8.69.3	STARLET.OLB	8–62
8.69.4	SYSALF.DAT	8–62
8.69.5	HELPLIB.HLB	8–62
8.69.6	Tailoring	8–62
8.69.7	Installing System Images	8–63
8.69.8	Logical Names	8–63
8.69.9	Accounts	8–63
8.69.10	UVDEFAULT.PAR	8–63
8.69.11	VMSTAILOR.EXE and VMSKITBLD.COM	8–63
8.69.12	Template File	8–63
8.69.13	MicroVMS Files Included for VMS Version 5.0	8–64
8.70	VMS EXECUTIVE — CHANGES	8–64
8.70.1	Modified Page Writing now Multithreaded	8–64
8.70.2	SYSGEN Parameters Added	8–64
8.70.3	Changes to Process Paging File Assignment	8–65
8.70.4	Paging File Recommendation	8–65
8.70.5	PAGEFILE.SYS — Changes	8–65
8.70.6	Extended Working Set Sizes	8–66

8.71	SYSTEM SECURITY INFORMATION	8–66
8.71.1	Directory and File Protection — Changes	8–66
8.71.2	Enhanced LAT Security Auditing and Break-In Detection	8–67
8.71.3	Protection of Security Auditing Information	8–67
8.71.4	Security Alarm ACEs—Restriction	8–69
8.71.5	SET ACL Command—Restriction	8–69
8.71.6	$SETUAI System Service—Restriction	8–69
8.71.7	DECnet Security	8–69

CHAPTER 9 PROGRAMMER RELEASE NOTES 9–1

9.1	VAX ADA VERSION 1.5 RUN-TIME LIBRARY — NOTES	9–1
9.1.1	Change in Use of Standard Input and Output Files	9–1
9.1.2	Closing and Re-opening Default Files	9–1
9.1.3	Temporary Files Have No Name	9–2
9.1.4	Asynchronous File Operations Supported	9–2
9.1.5	VAX Ada Restrictions	9–2
9.1.6	VAX BASIC RTL — Changes	9–3
9.1.7	DIBOL Routines Added	9–3
9.2	ALL-IN-1 VERSION 2.2 — RESTRICTIONS	9–4
9.3	VAX BASIC VERSION 3.2 INSTALLATION NOTE	9–5
9.4	DEBUGGER — NOTES	9–5
9.4.1	Debugging SMG Programs — Restriction	9–5
9.4.2	Linking Shareable Images for Debugging — Problem	9–6
9.4.3	MACRO Support — Change from Previous Versions	9–6
9.4.4	Making Images Shareable for Better Performance	9–6
9.4.5	Obsolete Commands	9–6
9.4.6	Screen-Mode Note — Change from Previous Versions	9–7
9.4.7	SET IMAGE Command — Problem Corrected	9–7
9.4.8	SET SCOPE Command — Problem Corrected	9–8
9.4.9	Using the Debugger on a VAXstation — Problem	9–8
9.4.10	VAXstation Support — Change from Previous Versions	9–8
9.4.11	VMS Version 4.6 and Version 4.7 Debugger Notes	9–8
9.4.11.1	CALL Command — Restriction Removed • 9–9	

9.5	LINKING WORKSTATION APPLICATIONS ON NONWORKSTATION SYSTEMS	9–9
9.6	MESSAGE ROUTER VERSION 3.0 INSTALLATION NOTES	9–9
9.7	VAX PASCAL VERSION 3.7 INSTALLATION NOTE	9–10
9.8	RECORD MANAGEMENT SERVICES — NOTES	9–11
9.8.1	Extended Asynchronous Interface	9–11
9.8.2	Block Mode Support for RMS Copy Operations	9–11
9.8.3	RMS Transparent Task-to-Task Network Operations	9–11
9.8.4	XAB$V_NUL Option — Clarification	9–11
9.8.5	Appending to Shared Sequential Files	9–11
9.8.6	$FREE Restriction	9–11
9.9	RUN-TIME LIBRARY — NOTES	9–12
9.9.1	Restriction on Using PPL$ENABLE_EVENT_SIGNAL	9–12
9.9.2	Omission from the PPL$CREATE_SHARED_MEMORY Routine Description	9–12
9.9.3	Creating Multiple Copies of a Program Through PPL$SPAWN	9–12
9.9.4	Corrections to the LIB$ADAWI Routine Description	9–13
9.9.5	Omissions from the LIB$SPAWN Routine Description	9–13
9.9.6	Correction to LIB$CREATE_VM_ZONE	9–13
9.9.7	String Procedures Performance Improvements	9–14
9.9.8	LIB$ Routine Changes	9–14
9.9.9	LIB$ Translation Tables Added	9–15
9.9.10	Significant LIB$ Updates	9–15
9.9.11	RTL Routine LIB$SYS_TRNLOG — Notes	9–16
9.9.12	MTH$ Math Routines — Changes	9–16
9.9.13	OTS$ Routines — Changes	9–17
9.9.14	SMG$ — Enhancements	9–17
9.9.15	Using SMG$CREATE_PASTEBOARD and SMG$CREATE_VIRTUAL_KEYBOARD — Restriction	9–17
9.9.16	Additional SMG$ Features	9–18
9.9.17	SMG$ Internal Changes	9–18
9.9.18	Obsolete SMG$ Routines	9–18
9.9.19	RTL Language Support Enhancements	9–19

9.10	NCS DIACRITICALS COLLATE IN OPPOSITE ORDER	9–19
9.11	VAX C RUN-TIME LIBRARY — CHANGES	9–20
9.12	VAX MACRO — NOTES	9–20
9.12.1	Restrictions	9–21
9.12.2	Fixed Problems	9–21
9.12.3	Known Problems	9–21
9.13	VAXTPU — NOTES	9–23
9.13.1	Incompatibilities with Previous Versions of VAXTPU	9–23
9.13.2	Problems in VAXTPU	9–25
9.13.3	Incompatibility with Future Versions	9–26
9.13.4	VAXTPU — Restrictions	9–26
9.13.5	VAXTPU Documentation — Notes	9–27
9.14	I/O SYNCHRONIZATION USING SYMMETRIC MULTIPROCESSING	9–41
9.14.1	Description of the SMP Change	9–41
9.14.2	General User Visibility	9–42
9.14.3	Programmer Responsibility	9–42
9.15	DRIVER MANIPULATION OF I/O POSTPROCESSING QUEUES	9–43
9.16	DELTA/XDELTA — USING XDELTA ON SMP SYSTEMS	9–45
9.17	EXE$BUFFRQUOTA AND EXE$BUFQUOPRC ROUTINES REPLACED	9–45
9.18	LAT — LTDRIVER CHANGES	9–45
9.18.1	Problem Corrected	9–45
9.18.2	New QIO Support	9–45
9.19	LPA11–K DRIVER (LADRIVER) — CHANGING TIMEOUTS ALLOWED	9–45

9.20	MODULAR EXECUTIVE — EFFECT ON SYSTEM SERVICES	9–46
9.20.1	New Description for the $MTACCESS	9–46
9.20.2	Instructions for Loading a Site-Specific Executive Loaded Image	9–46
9.21	MODULAR EXECUTIVE — EFFECT ON DEVICE DRIVERS	9–48
9.21.1	Preparing a Driver for Loading into the Operating System	9–49
9.22	MODULAR EXECUTIVE — EFFECT ON SYSTEM DUMP ANALYZER	9–49
9.22.1	SHOW EXECUTIVE Command	9–49
9.22.2	READ /EXECUTIVE Qualifier	9–49
9.23	MODULAR EXECUTIVE — EFFECT ON DELTA/XDELTA	9–49
9.23.1	XDELTA ;L Command	9–50
9.24	CAUTION ON USE OF NOP INSTRUCTION AS A DELAY MECHANISM	9–50
9.25	PROCESSOR REGISTER DEFINITION SYMBOLS	9–50
9.26	SET HOST/DTE/DIAL COMMAND — PROBLEM AND SOLUTION	9–51
CHAPTER 10	DOCUMENTATION INFORMATION	10–1
10.1	*VMS DCL DICTIONARY*	10–1
10.2	*VMS VAXCLUSTER MANUAL*	10–1
10.3	*VMS DEVICE SUPPORT MANUAL*	10–2
10.4	*VMS DEBUGGER MANUAL*	10–2
10.5	*VMS I/O USER'S REFERENCE MANUAL: PART II*	10–2

10.6	*VMS SYSTEM MESSAGES AND RECOVERY PROCEDURES REFERENCE MANUAL: PART I*	10–3
10.7	*VMS MAIL UTILITY MANUAL*	10–3
10.8	*VMS DECNET TEST SENDER/DECNET TEST RECEIVER UTILITY MANUAL*	10–3
10.9	*VMS NETWORKING MANUAL*	10–4
10.10	*GUIDE TO VMS SOFTWARE INSTALLATION*	10–4
10.11	*VMS MONITOR UTILITY MANUAL*	10–4
10.12	*VMS NETWORK CONTROL PROGRAM MANUAL*	10–4
10.13	*VMS CONVERT AND CONVERT/RECLAIM UTILITY MANUAL*	10–5
10.14	*VMS LINKER UTILITY MANUAL*	10–5
10.15	*VMS RECORD MANAGEMENT SERVICES MANUAL*	10–5
10.16	*VMS SYSTEM SERVICES VOLUME*	10–5
10.17	*VMS RUN-TIME LIBRARY ROUTINES VOLUME*	10–5
10.17.1	**RTL LIB$ (Library) Facility**	10–5
10.17.1.1	Corrections to the LIB$ADAWI Routine Description • **10–6**	
10.17.1.2	Omissions from the LIB$SPAWN Routine Description • **10–6**	
10.17.1.3	Correction to LIB$CREATE_VM_ZONE • **10–6**	
10.18	*VAX RMS JOURNALING MANUAL*	10–6
10.18.1	**Recovery Unit Flags Field XAB$W_RU_FLAGS**	10–7
10.18.2	**SET FILE/RU_ACTIVE**	10–7
10.18.3	**SET FILE /RU_FACILITY**	10–8

APPENDIX A RUN-TIME LIBRARY ARGUMENT NOTATION A–1

A.1 NAME A–1

A.2 ACCESS TYPE A–2

A.3 DATA TYPE A–2

A.4 PASSING MECHANISM A–4

A.5 ARGUMENT FORM A–4

A.6 EXAMPLES OF DOT NOTATION A–6

INDEX

EXAMPLES

9–1 Sample VAX BLISS Template for Callable VAXTPU 9–28

9–2 Building a Callback Item List with VAX FORTRAN 9–33

FIGURES

6–1 Example of a Product Authorization Key (PAK) 6–2

6–2 Example of a VMS Availability Product Authorization Key (PAK) 6–8

6–3 Example of a VMS Activity Product Authorization Key (PAK) 6–14

6–4 Example of a VMS Volume Shadowing Product Authorization Key (PAK) 6–19

TABLES

1–1 Layered Products Currently Available for VMS Version 5.0 1–5

1–2 Layered Products To Be Supported for VMS Version 5.0 1–8

2–1 Restart Settings for VAX Computers 2–5

6–1 License Unit Requirement Tables (LURT) 6–27

7–1	Obsolete Commands	7–3
8–1	New Initial Values for SYSTEM and DEFAULT Records in AUTHORIZE	8–12
8–2	FAB$B_JOURNAL Bit Settings	8–31

Preface

This manual supersedes all release note documentation for previous versions of the VMS operating system. It includes, or updates, any previous release notes that are still pertinent to the Version 5.0 release.

For information about the new features included in VMS Version 5.0, see the *New Features Manual*.

Intended Audience

This manual is intended for all system users. Read this manual before you install, upgrade, or use VMS Version 5.0.

Document Structure

This manual contains the following chapters:

- Chapter 1 contains pertinent information about the upgrade procedure, including cautions, restrictions, notes, and suggestions about upgrading your VAX computer system.
- Chapter 2 contains procedures you must perform before beginning the VMS upgrade.
- Chapter 3 describes the procedures for upgrading a single computer system.
- Chapter 4 describes the procedures for upgrading a VAXcluster environment with a single common system root.
- Chapter 5 provides information about mandatory and optional procedures you perform after the upgrade.
- Chapter 6 contains information about the VMS License Management Facility (LMF).
- Chapter 7 contains the VMS Version 5.0 release notes of interest to the general user.
- Chapter 8 contains the VMS Version 5.0 release notes of interest to the system manager.
- Chapter 9 contains the VMS Version 5.0 release notes of interest to the programmer.
- Chapter 10 lists additional information about the VMS Version 5.0 documentation set.
- Appendix A provides information about dot notation, which is used to describe the arguments of a Run-Time Library routine in a VMS source file.

Conventions

The following conventions are observed in this manual:

Convention	Meaning
RET	In examples, a key name (usually abbreviated) shown within a box indicates that you press a key on the keyboard; in text, a key name is not enclosed in a box. In this example, the key is the RETURN key. (Note that the RETURN key is not usually shown in syntax statements or in all examples; however, assume that you must press the RETURN key after entering a command or responding to a prompt.)
CTRL/C	A key combination, shown in uppercase with a slash separating two key names, indicates that you hold down the first key while you press the second key. For example, the key combination CTRL/C indicates that you hold down the key labeled CTRL while you press the key labeled C. In examples, a key combination is enclosed in a box.
$ SHOW TIME 05-JUN-1988 11:55:22	In examples, system output (what the system displays) is shown in black. User input (what you enter) is shown in red.
$ TYPE MYFILE.DAT . . .	In examples, a vertical series of periods, or ellipsis, means either that not all the data that the system would display in response to a command is shown or that not all the data a user would enter is shown.
input-file, . . .	In examples, a horizontal ellipsis indicates that additional parameters, values, or other information can be entered, that preceding items can be repeated one or more times, or that optional arguments in a statement have been omitted.
[logical-name]	Brackets indicate that the enclosed item is optional. (Brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
quotation marks apostrophes	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe is used to refer to a single quotation mark (').

This manual often refers to the following products by their abbreviated names:

- The VAX 6200 Series computer is referred to as the VAX 6200 series.
- The VAX 8200, VAX 8250, VAX 8300, and VAX 8350 computers are referred to collectively as the VAX 8200, 8250, 8300, 8350.

- The VAX 8200 computer is referred to as the VAX 8200.
- The VAX 8250 computer is referred to as the VAX 8250.
- The VAX 8300 computer is referred to as the VAX 8300.
- The VAX 8350 computer is referred to as the VAX 8350.
- The VAX 8820, VAX 8830, and VAX 8840 computers are referred to collectively as the VAX 8820, 8830, and 8840.
- The VAX 8820 computer is referred to as the VAX 8820.
- The VAX 8830 computer is referred to as the VAX 8830.
- The VAX 8840 computer is referred to as the VAX 8840.
- The VAX 8530, VAX 8550, VAX 8700, and VAX 8800 computers are referred to collectively as the VAX 8530, 8550, 8700, and 8800.
- The VAX 8530 computer is referred to as the VAX 8530.
- The VAX 8550 computer is referred to as the VAX 8550.
- The VAX 8700 computer is referred to as the VAX 8700.
- The VAX 8800 computer is referred to as the VAX 8800.
- The VAX 8600 and VAX 8650 computers are referred to collectively as the VAX 8600 and 8650.
- The VAX 8600 computer is referred to as the VAX 8600.
- The VAX 8650 computer is referred to as the VAX 8650.
- The VAX–11/780 and VAX–11/785 computers are referred to collectively as the VAX–11/780 and 11/785.
- The VAX–11/780 computer is referred to as the VAX–11/780.
- The VAX–11/785 computer is referred to as the VAX–11/785.
- The VAX–11/750 computer is referred to as the VAX–11/750.
- The VAX–11/725 and VAX–11/730 computers are referred to collectively as the VAX–11/725 and 11/730.
- The VAX–11/725 computer is referred to as the VAX–11/725.
- The VAX–11/730 computer is referred to as the VAX–11/730.
- The VAXstation 8000 computer is referred to as the VAXstation 8000.
- The MicroVAX, VAXstation, and VAXserver 3600 Series computers are referred to collectively as the MicroVAX, VAXstation, and VAXserver 3600 Series.
- The MicroVAX computer is referred to as the MicroVAX.
- The VAXstation computer is referred to as the VAXstation.
- The VAXserver 3600 Series computer is referred to as the VAXserver 3600 Series.
- The VAXstation I, II, II/GPX, MicroVAX I and II computers are referred to collectively as the VAXstation I, II, II/GPX, MicroVAX I and II.
- The VAXstation I computer is referred to as the VAXstation I.

- The VAXstation II computer is referred to as the VAXstation II.
- The VAXstation II/GPX computer is referred to as the VAXstation II/GPX.
- The MicroVAX I computer is referred to as the MicroVAX I.
- The MicroVAX II computer is referred to as the MicroVAX II.
- The VAXstation 2000 and MicroVAX 2000 computers are referred to collectively as the VAXstation 2000 and MicroVAX 2000.
- The VAXstation 2000 computer is referred to as the VAXstation 2000.
- The MicroVAX 2000 computer is referred to as the MicroVAX 2000.
- A VAXcluster configuration is referred to as a VAXcluster.
- A Local Area VAXcluster configuration is referred to as a Local Area VAXcluster.

1 Upgrading to VMS Version 5.0

Upgrades are performed on existing operating systems to bring them up to the latest major release version from the most recent maintenance version. For example, if you want to bring a Version 4.7 system up to Version 5.0, you perform the Version 5.0 upgrade.

Chapters 1 through 5 cover the following topics on the VMS Version 5.0 upgrade procedure for VAX computers and clusters:

Topic	Description
Upgrade considerations	A summary of the upgrade procedure and upgrade cautions, restrictions, notes, and suggestions (Chapter 1).
Upgrade preparations	Procedures to perform before beginning the upgrade (Chapter 2).
System upgrade	Procedures to upgrade a single system (Chapter 3).
Concurrent upgrade	Procedures to upgrade a VAXcluster while the cluster is shut down (Chapter 4).
Rolling upgrade	Procedures to upgrade a VAXcluster while the cluster remains available (Chapter 4).
Post-upgrade procedures	Mandatory and optional procedures to perform after the upgrade (Chapter 5).

To perform the VMS Version 5.0 upgrade procedure successfully, you must understand the basic operations of the system that you are upgrading. Refer to the VAX computer-specific installation and operations guide as well as the *Introduction to VMS System Management*.

Before you begin the upgrade procedure, read Section 1.2, "Upgrade Considerations," that contains important information you need to know prior to performing an upgrade of your computer. DIGITAL recommends that you read the *entire* section before beginning. Also, read Section 1.3 for information on how layered products are handled.

After reading Sections 1.2 and 1.3, refer to the sections that pertain to your system's configuration and follow the numbered steps.

1.1 Upgrade Summary

An upgrade consists of the following basic steps:

1. Backing up your current system disk
2. Creating a backup copy of your console media
3. Booting from the backup copy of your system disk
4. Moving any files that may affect the upgrade, to a user directory
5. Making sure you have enough space on the system disk to do the upgrade
6. Making sure the CPU is set up for automatic restart
7. Preventing users from logging into the system, stopping all queues, and, if applicable, shutting down the network
8. Invoking the VMSINSTAL command procedure and following VMSINSTAL instructions through six upgrade phases
9. Booting from the upgraded system disk when the upgrade is completed and registering all your Product Authorization Keys (PAKs)
10. Converting queue and MAIL database files to the VMS Version 5.0 format and making any required changes in system procedures before resuming normal operations

1.2 Upgrade Considerations

You must be aware of the following considerations, cautions, and restrictions before you begin the upgrade:

- Your system must be running VMS (or MicroVMS) Version 4.6 or Version 4.7 to complete the VMS Version 5.0 upgrade. If your system is not running Version 4.6 or Version 4.7, you must upgrade to at least VMS (or MicroVMS) Version 4.6 before you begin the VMS Version 5.0 upgrade procedure.
- For a rolling upgrade, all systems in the cluster must be running VMS Version 4.7.
- The upgrade procedure does *not* work across the network.
- If current system directories have been altered in any way, the upgrade procedure might not work correctly. You must restore your operating system to a standard system before performing the upgrade.
- The upgrade cannot be applied to a tailored system. If you have a tailored system disk, you must *install* the VMS operating system. Refer to the installation and operations guide for your VAX computer.
- The VMS Version 5.0 upgrade procedure does not support RC25 or RK07 system disks. However, you can *install* VMS Version 5.0 on RC25 or RK07 system disks.
- If you are using an RD52 disk for the system disk, the upgrade procedure does not restore certain VMS Version 5.0 files (full library and optional files). After the upgrade is completed, move the files you need onto the disk.

- You must not move the system disk or the distribution volume from one device to another during the upgrade.
- The upgrade procedure automatically converts all private system disks to the common system disk format.
- The upgrade procedure purges the paging, swapping, dump, and authorization files.
- The upgrade procedure deletes everything in the [SYSERR] directory.
- The upgrade procedure deletes all operator and accounting logs. To save these files, move them to a user directory before starting the upgrade.
- The upgrade procedure preserves the following files:

  ```
  [SYSEXE]NOTICE.TXT
  [SYSEXE]RIGHTSLIST.DAT
  [SYSEXE]SYSUAF.DAT
  ```

- The name of the system startup command procedure changes to SYSTARTUP_V5.COM.
- The name of the shared MAIL database changes to VMSMAIL_PROFILE.DATA.
- The name of the DECnet proxy database changes to NETPROXY.DAT.
- Before starting the upgrade procedure in local area VAXcluster configurations, you must shut down all satellite nodes.
- The queue and MAIL database files must be converted to the VMS Version 5.0 format before the full capabilities of Version 5.0 are available.
- You can only log in under the system manager's account on the console until you register your VMS license using the License Management Facility (LMF). If you are currently under service, this will be automatically done for you as part of the mandatory update. See Section 6.6.
- If you are running the VMS operating system in a VAXcluster configuration, you must separately register your VAXcluster license using LMF. Until you register the license for each node, the following messages will be displayed when you attempt to boot a CPU into a VAXcluster environment:

  ```
  %LICENSE-E-NOAUTH, DEC VAXCLUSTER use is not authorized on this node
  -LICENSE-F-NOLICNESE, no license is active for this software product
  -LICENSE-I-SYSMGR, please see your system manager
  Startup processing continuing....
  ```

  If you do not have your VAXcluster license, contact your DIGITAL sales representative.

Suggestions for the Upgrade

The following list contains additional suggestions to help you perform the upgrade:

- A major part of the upgrade is automated and does not require an operator. Therefore, you should perform the upgrade from a hardcopy terminal to record all returned data. If your VAX computer does not have a hardcopy terminal, attach a printer to the console terminal. Refer to the installation and operations guide for your VAX computer. Note, however,

that if you are upgrading a VAX 8530, 8550, 8700 or 8800, you must perform the upgrade from the console (PRO 380).

- If the upgrade is interrupted for any reason, you can resume from the point at which the system was most recently booted. The upgrade reboots the system several times during the upgrade.

Material Needed for the Upgrade

The VMS Version 5.0 upgrade requires the following materials:

- A VMS Version 5.0 software distribution kit.
- A blank console volume of the following type:
  - RX01 floppy diskette for the VAX–11/780 or VAX–11/785 computers
  - RL02 for the VAX 8600 and VAX 8650 computers
  - RX50 floppy diskette for the VAX 8200, VAX 8250, VAX 8300, and VAX 8350 computers
  - TU58 cartridge for the VAX–11/725, VAX–11/730, and VAX–11/750 VAX computers
- The installation and operations guide for your VAX computer has the instructions for several upgrade procedures.

NOTE: You do not need a blank console volume to upgrade a VAX 8530, 8550, 8700, or 8800; or any of the MicroVAX or VAXstation systems.

1.3 Layered Products Caution

Because of the way the VMS Version 5.0 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 (refer to Table 1–1), yet exhibits unexpected behavior once Version 5.0 is running, check the sections of this document that describe layered products restrictions. If problems persist, contact your DIGITAL support representative.

Table 1–1 lists the layered products that are currently supported for VMS Version 5.0. Table 1–2 lists the layered products that will be supported soon after the release of VMS Version 5.0. Contact your DIGITAL representative for specific release dates.

Table 1–1 Layered Products Currently Available for VMS Version 5.0

Layered Product	Version Number
ALL-IN-1[1]	2.1
ALL-IN-1/BEV[1]	2.2
BASEVIEW	1.2
CMR21 Host Utility	1.1
Courseware Design System	1.0
DECnet/SNA Application Programming Interface	2.2
DECnet/SNA Gateway VMS DISOSS Document Exchange Facility	1.4
DECnet/SNA Gateway VMS Distributed Host Command Facility	1.1
DECnet/SNA VMS 3270 Data Stream Programming Interface	1.3
DECnet/SNA VMS 3270 Terminal Emulator	1.4
DECnet/SNA VMS APPC/LU6.2	2.0
DECnet/SNA VMS Printer Emulator	1.1
DECnet/SNA VMS Remote Job Entry	1.3
DECnet/SNA Gateway	1.5
DECnet/SNA Gateway for DEC MicroServer	2.1
DECnet–VAX	5.0
DECserver 100	1.3
DECserver 200	2.0
DECserver 500[2]	1.0
DRB32 VMS Drivers	2.0
LCG01 Software	1.4
MicroPower/Pascal-VMS	2.4
PDP–11 FORTRAN–77/VAX to RSX	5.2
PDP–11 Symbolic Debugger/VAX to RSX	2.0
PLXY-11/VAX	1.4
ReGIS Software	1.1
Session Support Utility	1.1
Terminal Server Manager	1.2
VAX 2780/3780 Protocol Emulator	1.7
VAX Ada	1.5
VAX ADE	2.4A
VAX BASIC[3]	3.2

[1]Some restrictions apply. Refer to Section 9.2.

[2]Refer to the cover letters for supporting VAX 8800 computers and the VAX 6200. Contact your DIGITAL representative for copies.

[3]Some restrictions apply. Refer to Section 9.3.

Table 1–1 (Cont.) Layered Products Currently Available for VMS Version 5.0

Layered Product	Version Number
VAX BCP	1.1
VAX BLISS-32	4.4
VAX C	2.4
VAX COBOL	4.0
VAX COBOL Generator	1.2
VAX Common Data Dictionary	4.0
VAX DATATRIEVE	4.1
VAX DEC/CMS	3.0
VAX DEC/MMS	2.3
VAX DEC/Shell	2.1
VAX DEC/Test Manager	2.3
VAX DECalc-Plus	3.0A
VAX DECalc	3.0A
VAX DECgraph	1.5
VAX DECSCAN MicroVMS and ELN Bitbus Software Drivers	2.0
VAX DECslide	1.3
VAX DIBOL	4.0
VAX DOCUMENT	1.0
VAX DT07	2.2
VAX EDCS	1.1A
VAX FMS	2.3
VAX FORTRAN	4.8
VAX GKS	3.1
VAX Language Sensitive Editor	2.2
VAX LIMS/SM	1.3
VAX LISP/VMS	2.2
VAX Message Router[4]	3.0
VAX Notes	1.3
VAX OPS5	2.2
VAX Pascal[5]	3.7
VAX Performance Advisor	1.2
VAX Performance and Coverage Analyzer	2.0
VAX PHIGS	1.0
VAX PL/I	3.1
VAX Printserver 40 Client Software	2.0

[4]Some restrictions apply. Refer to Section 9.6.

[5]Some restrictions apply. Refer to Section 9.7.

Table 1–1 (Cont.) Layered Products Currently Available for VMS Version 5.0

Layered Product	Version Number
VAX Printserver 40 Supporting Host Software	2.0
VAX Rdb/ELN	2.1
VAX ReGIS to Sixels Converter	1.0
VAX RMS Journaling	5.0
VAX Scan	1.1
VAX Scriptprinter Software	1.1
VAX Software Project Manager	1.1
VAX Source Code Analyzer	1.2
VAX SPM	3.2
VAX TDMS	1.7
VAX TU70/72 Device Driver	1.2
VAX VALU	2.1
VAX Volume Shadowing	5.0
VAX VTX	3.0
VAX XWAY	1.1A
VAX–11 RSX	2.4
VAX–11 RTEM[6]	2.3
VAXcluster Software	5.0
VAXELN Ada	1.2
VAXELN Toolkit	3.0
VAXset	Release 6
VIDA with IDMS/R	2.0
VMS Workstation Software	4.0
VS11-VAX Driver	2.4
X25Router	1.0

[6]Refer to the cover letter for RTEM, Version 2.3.

Table 1–2 Layered Products To Be Supported for VMS Version 5.0

Layered Product	Version Number
A-to-Z Base System	2.6
A-to-Z Business Graphics	2.6
A-to-Z Database Manager	2.6
A-to-Z Developer's Kit	2.6
A-to-Z Electronic Mail	2.6
A-to-Z Word Processing	2.6
AAF01/VMS Subroutine Library	1.3
ADF01/VMS Subroutine Library	3.2
ALL-IN-1 System for Sales and Marketing	1.2
C.A.S Delivery System	1.6
Courseware Authoring System	1.6
DECnet/SNA Gateway VMS Data Transfer Facility	2.0
EDE-W Document Exchange	2.0
KMV1A MicroVMS Development Tools	2.0
KMV1A MicroVAX Driver	2.0
KMV1A MicroVMS X.25 Link Level Software	2.0
MIRA Switch Control	2.0
MUXserver 100 Remote Terminal Server	2.2
NMCC/DECnet ETHERNIM	2.1
NMCC/DECnet Monitor	2.1
Remote System Manager	2.1
Spatial/II	1.2
WPS-PLUS/VMS	3.0
VAX 3271 Protocol Emulator	2.5
VAX ACMS	3.0
VAX APL	3.1
VAX DAL	1.6
VAX Data Distributor	2.0
VAX DBMS	4.0
VAX DECrad	4.0
VAX DECReporter	2.1
VAX DEC/Map	2.0
VAX DECScan VMS Software Tool Kit	1.1
VAX DST32 VMS Device Driver	1.1
VAX DY32	3.0
VAX DSM	4.1
VAX High Performance Workstation Software	1.1
VAX KCT32	2.0

Table 1–2 (Cont.) Layered Products To Be Supported for VMS Version 5.0

Layered Product	Version Number
VAX Key Distribution Center	1.1
VAX KMS11-BD/BE HDLC/BSC Framing Software	2.0
VAX KMS11-BD/BE X.25 Link Level Software	2.0
VAX Message Router X.400 Gateway	2.1
VAX Message Router/P Gateway	1.1
VAX Message Router/S Gateway	1.1
VAX OSI Transport Service	2.0
VAX PSI	4.2
VAX PSI Access	4.2
VAX Public Access Communications	1.2
VAX Rally	2.0
VAX Rdb/VMS	3.0
VAX Real-Time Accelerator Software	2.0
VAX TEAMDATA	1.3
VAXcluster Console System	1.2
VAXinfo I	1.4
VAXinfo II	1.4
VAXinfo II	1.4
VAXLAB Software Library	1.3
VMS/SNA	1.3
VAX/VMS Services for MS-DOS	2.0
VNXSET	Release 5
VSV21 VMS Support Software	3.0

2 Pre-Upgrade Procedure

This chapter explains how to prepare the system for the upgrade procedure. Complete the tasks in this chapter before you begin the upgrade procedure. When you have finished the pre-upgrade procedure, go to Chapter 3.

NOTE: Log in to the system manager's account, SYSTEM, for all pre-upgrade tasks.

The pre-upgrade procedure is divided into numbered tasks. Follow these steps in the indicated order.

NOTE: If you are using volume-shadowed system disks, you must perform specific pre-upgrade procedures and post-upgrade procedures to upgrade successfully to VMS Version 5.0. The *VAX Volume Shadowing Manual* describes these procedures.

The following procedure describes each pre-upgrade step in detail:

1 Backing Up and Restoring the System Disk

Where possible, you should perform the upgrade using a backup version of the system disk, not the current, working system disk. Use standalone BACKUP to back up the current system disk. Refer to the installation and operations guide for your VAX computer for a step-by-step procedure for backing up the system disk.

2 Backing Up and Restoring the Console Media

If you are upgrading a VAX-11/725, 11/730, 11/750, 11/780, 11/785, 8200, 8250, 8300, 8350, 8600, or 8650, back up the console media. For any other VAX computer, go to step 3.

Use the CONSCOPY command procedure to save and restore the console media.

The CONSCOPY command procedure does either a save or a restore operation on the console medium. First, do the save operation. Then, invoke the procedure again to do a restore operation.

a. To invoke the procedure, enter the following command:

```
$ @SYS$UPDATE:CONSCOPY
```

b. The CONSCOPY procedure prompts for the type of VAX computer. Enter the type of the computer that you are upgrading.

c. The procedure asks whether you want to save or restore the medium. First, enter SAVE to make the backup copy.

d. The procedure asks for the name of the console device. Enter the device name in the *ddcu* format.

e. The procedure asks if you want to log messages. Enter YES or NO.

f. The procedure asks if the device is ready. Answer YES when the console device is ready.

g. Follow the instructions provided by the procedure to complete the backup.

h. Repeat the procedure to restore the console media. In step 2b, enter RESTORE.

For more information on using CONSCOPY, refer to the installation and operations guide for your VAX computer.

3 Booting from the Backup System Disk

To boot from the backup copy of the system disk, perform the following steps (unless the system disk is fixed):

a. Complete an orderly shutdown:

```
$ @SYS$SYSTEM:SHUTDOWN
```

The shutdown procedure asks you a series of questions about the system shutdown. Answer each question appropriately for your system.

b. Halt the VAX computer according to the procedure described in the installation and operations guide for your VAX computer. If appropriate, mount the backup copy of the system disk.

c. Boot the system according to the procedure described in the installation and operations guide for your VAX computer.

d. Log in to the system manager's account, SYSTEM.

4 Copying SYSUAF.DAT

The current copy of SYSUAF.DAT must be on the system disk in SYS$SYSTEM. Otherwise, you will not be able to proceed with the upgrade. If you have moved your copy of SYSUAF.DAT to another location, copy it to SYS$SYSTEM for the duration of the upgrade.

5 Analyzing the System Disk

Analyze the system disk for inconsistencies and errors in the file structure. If there are errors, you can correct them and recover free disk space.

To analyze the system disk you are using for the upgrade, enter the following command:

```
$ ANALYZE/DISK_STRUCTURE  SYS$SYSDEVICE
```

If you find any errors on the system disk, fix them using the following command:

```
$ ANALYZE/DISK_STRUCTURE/REPAIR  SYS$SYSDEVICE
```

6 Checking and Setting Page File Size

Check the system page file to make sure there are at least 4600 blocks.

a. List the number of blocks in the system page file, using the following command:

```
$ @SYS$UPDATE:SWAPFILES
```

The SWAPFILES.COM procedure displays the current sizes of the page, swap, and dump files. The procedure prompts you to enter new values.

```
Enter new size for paging file:
```

b. If the current page file size (PAGEFILE.SYS) is greater than 4600 blocks, press RETURN. If the value is less than 4600 blocks, enter 4600 and press RETURN.

c. If the swap and dump files are on the system disk, the procedure displays the current sizes for each file. To keep the current values, press RETURN after each prompt.

If the swap file is not on the system disk, the procedure asks if you want to create one. You do not need a swap file on the system disk for the upgrade procedure. Therefore, enter the appropriate response for your system.

d. If you changed the size of the page file or created a swap file, a prompt appears telling you to reboot the system. Do not reboot now. You will reboot in step 14.

7 Checking Free Blocks on the System Disk

Check the system disk to make sure there is sufficient free disk space for the upgrade. You need 27,000 free blocks, plus the number of blocks contained in SYS$HELP:HELPLIB.HLB or SYS$LIBRARY:STARLET.OLB, whichever is larger, or a minimum of 31,000 blocks.

If the system disk is an RD52, you need only 23,000 blocks.

To determine whether you have enough free disk space, use the following procedure:

a. To determine the size of the HELPLIB and STARLET files, enter the following commands:

```
$ DIRECTORY/SIZE SYS$HELP:HELPLIB.HLB
$ DIRECTORY/SIZE SYS$LIBRARY:STARLET.OLB
```

b. Add the number of blocks for the largest file to the 27,000 base value. For example, if STARLET.OLB is the largest file, with 4100 blocks, you need a total of 31,100 blocks for the upgrade.

c. To determine whether you have enough free disk space, enter the following command:

```
$ SHOW DEVICE SYS$SYSDEVICE
```

d. If the number of free blocks is less than 27,000 plus the size of HELPLIB.HLP or STARLET.OLB (or the 31,000 minimum), delete and purge files to create space.

8 Checking SYSGEN Parameters

If you have not modified any SYSGEN parameters for your system, skip this step.

If you have modified SYSGEN parameters, and you want to retain them after the upgrade, make sure that you have entered the changed parameter in MODPARAMS.DAT. The value in MODPARAMS.DAT will be retained after the upgrade because AUTOGEN calculates new values based on MODPARAMS.DAT.

Enter the name of the parameter that you changed and the value that AUTOGEN needs to add to the default minimum value. For example, if you modified GBLPAGES by 128 pages above the default, enter 128 into MODPARAMS.DAT, as follows:

```
ADD_GBLPAGES=128
```

If you modified the SYSGEN parameter INTSTKPAGES, check its current value in MODPARAMS.DAT. If this parameter has a value lower than 4, the system might crash during a reboot. If INTSTKPAGES is lower than 4, either edit it or delete it from MODPARAMS.DAT.

The value for MSCP_BUFFER currently represents a number of bytes. For VMS Version 5.0, the value for MSCP_BUFFER represents a number of pages. If you modified MSCP_BUFFER, you must change its value to represent pages.

9 Shutting Down the Network

If you are not running DECnet–VAX, go to step 10.

To shut down the network, enter the following commands:

```
$  RUN SYS$SYSTEM:NCP
NCP> SET EXECUTOR STATE OFF
NCP> EXIT
```

10 Stopping Queues

Stop all batch and print activities, as follows:

```
$  STOP/QUEUE/MANAGER
```

11 Setting SYSGEN Parameters

If you are upgrading from MicroVMS, go to step 12.

Check the current value of the STARTUP_P1 parameter, as follows:

a. Invoke SYSGEN:

```
$ RUN SYS$SYSTEM:SYSGEN
```

b. Set for "CURRENT":

```
SYSGEN>  USE CURRENT
```

c. Show the STARTUP_P1 parameter and set to "MIN":

```
SYSGEN>  SHOW STARTUP_P1

Parameter Name            Current   Default   Minimum   Maximum Unit  Dynamic
--------------            -------   -------   -------   ------- ----  -------
STARTUP_P1             "        "    "    "    "    "    "ZZZZ" Ascii

SYSGEN> SET STARTUP_P1 "MIN"
```

d. Save the new parameter settings and exit SYSGEN:

```
SYSGEN>  WRITE CURRENT
SYSGEN>  EXIT
```

12 Checking on Completion of Pre-Upgrade

Once you invoke the upgrade procedure, it verifies that the pre-upgrade procedures have been completed, as follows:

- Sufficient free blocks on the system disk are available.
- The system disk is not tailored.
- Volume shadowing is disabled on the system disk.
- The STARTUP_P1 SYSGEN parameter is set to "MIN" (VMS only).
- DECnet is shut down.
- Queue manager is not running.
- SYSUAF.DAT is in SYS$SYSTEM.
- PAGEFILE.SYS is in SYS$SPECIFIC:[SYSEXE].

All of these conditions must be met for the upgrade procedure to continue. Check each item to make sure your system meets these conditions.

13 Setting Automatic Restart

Make sure automatic restart is enabled to allow the upgrade to continue automatically. To determine how to enable auto restart, refer to Table 2–1 and the installation and operations guide for your VAX computer.

Table 2–1 Restart Settings for VAX Computers

VAX Computer	Restart	Required Setting
VAX–11/725, 11/730	AUTO/RESTART BOOT switch	ON
VAX–11/750	POWER ON ACTION switch	(RESTART)
VAX–11/780, 11/785	AUTO/RESTART switch	ON
VAX 8600, 8650	RESTART/BOOT switch	RESTART/BOOT
VAX 8200, 8250, 8300, 8350, 8530, 8550, 8700, 8800		AUTO RESTART and AUTO BOOT
MicroVAX and VAXstation	AUTO/RESTART switch	ON

14 Rebooting the System

If you modified a SYSGEN parameter (including STARTUP_P1), if you changed the page file size, or if you created or modified the swap file, you *must* reboot the system before continuing the upgrade procedure.

Shut down the system as follows:

```
$ @SYS$SYSTEM:SHUTDOWN
```

The shutdown procedure asks you a series of questions about the system shutdown. Answer each question appropriately for your system. Be sure to answer YES when asked if an automatic system reboot should be performed.

15 Configuring System Devices

Skip this step if you are upgrading from the MicroVMS operating system, and go to step 16. Run SYSGEN to reconfigure all available system devices and run STARTUP CONFIGURE:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> AUTOCONFIGURE ALL
SYSGEN> EXIT
$ @SYS$SYSTEM:STARTUP CONFIGURE
```

NOTE: You must run STARTUP CONFIGURE to enable the system to recognize all HSC-based disks and tapes after the reboot.

16 Isolating the System from Users

To prevent users from logging in to the system, enter the following command:

```
$ SET LOGINS/INTERACTIVE=0
```

17 Modifying Boot Command Procedures

Perform this step only if you are upgrading a VAX 8530, 8550, 8700, or 8800 computer. If you are not upgrading one of these VAX computers, go to Chapter 3.

Before beginning the upgrade procedure, you must modify the boot command procedures, DEFBOO and *ddd*GEN. Before making these modifications, make copies of the files. You will need the original versions for the upgrade procedure.

The installation and operations guide for your VAX computer describes the procedures for copying and modifying DEFBOO and *ddd*GEN. Use these procedures to make one change in DEFBOO and one change in *ddd*GEN. Both procedures must specify SYSF as the root directory from which to boot.

In DEFBOO, edit the BOOT command, as follows:

```
BOOT dddn/R5:F0000000
```

- where *ddd* is BCI, BDA, or UDA
- where *n* is the unit number of the drive holding the system disk

In *ddd*GEN, edit the command that deposits a value in register 5 (R5), as follows:

```
DEPOSIT R5 F0000001
```

Go to Chapter 3.

3 Upgrade Procedure for a Single System

This chapter explains the upgrade procedure for a single VMS system. Before you begin the upgrade procedure, you must complete the entire pre-upgrade procedure described in Chapter 2.

The upgrade procedure is divided into six phases, preceded by a step to get the devices ready and to invoke, VMSINSTAL. You must complete each phase of the upgrade procedure.

3.1 Beginning the Upgrade Procedure

This section describes how to invoke VMSINSTAL and respond to its prompts. You can enter a question mark (?) for help at any time while running VMSINSTAL. For additional information about VMSINSTAL, refer to the *VMS Install Utility Manual*.

NOTE: Use the console terminal to complete this part of the upgrade.

1 Mounting the Distribution Volume

If you are using a tape drive, be sure that the tape is write-protected. Then, thread the tape and put the drive on line.

If you are using a disk drive, physically mount the disk and write-protect it.

2 Invoking VMSINSTAL

Invoke VMSINSTAL as follows:

```
$ SET DEFAULT SYS$UPDATE
$ @VMSINSTAL
```

The following VMSINSTAL message is displayed:

```
        VMS Software Product Installation Procedure Vnnn

It is (date) at (time).
Enter a question mark (?) at any time for help.
```

3 Backing Up the Disk

The first prompt asks you about disk backup:

```
*Are you satisfied with the backup of your system disk [YES]?
```

If you have backed up the system disk already, press RETURN to continue the upgrade and go to step 4.

If you have not backed up the system disk or the console media, or you are not satisfied with the backup, complete the rest of this step.

a. Enter NO and press RETURN. VMSINSTAL returns to the DCL level.

b. For instructions on backing up the system disk and console media, refer to the installation and operations guide for your VAX computer.

c. When the backup operation is completed, start the upgrade procedure again from the beginning of Section 3.1.

4 Specifying the Device Name

VMSINSTAL requests the name of the drive holding the distribution volume:

```
*Where will the distribution volumes be mounted:
```

For non-HSC devices, enter the device name using the format *ddcu* and press RETURN.

For HSC devices, use the format *hsc_name$ddcu*, where

- *hsc_name* identifies the HSC device.
- *dd* specifies the type of device (load device).
- *c* refers to the controller number.
- *u* refers to the device unit number.

For example, you might enter the following:

```
MUTT$DJA2
```

After you enter the device name successfully (VMSINSTAL error messages are not displayed), go to step 5.

If VMSINSTAL returns an error message, reenter the device name. If another error message is returned, check the device name to make sure you are entering the correct name. Check the device to make sure it is properly connected and set up. Enter the name again.

If you still get an error message, verify the device name as follows:

a. Enter CTRL/Y.

b. To verify the device name and status, enter the SHOW DEVICE command.

c. Begin the upgrade procedure over again from the beginning of Section 3.1.

5 Specifying the Product

When VMSINSTAL prompts for the products, enter the VMS Version 5.0 product name, as follows:

```
*Products:  VMS050
```

6 Readying the Distribution Volume

VMSINSTAL prompts you to place the first distribution volume (if there is more than one piece of medium). Enter Y and press RETURN.

```
*Are you ready?  Y
```

After several minutes, the following messages are displayed:

```
%MOUNT-I-MOUNTED, [volume_label]       mounted on _ddcu:
The following products will be processed:
  VMS V5.0
```

After several more minutes, the following messages are displayed:

```
        Beginning installation of VMS 5.0 at [date hh:mm]
%VMSINSTAL-I-RESTORE, Restoring product saveset A...

                        VAX/VMS V5.0 Upgrade Procedure
.
.
.
.
```

7 Reading Information Messages

The upgrade procedures display several messages that provide you with the following information:

- What VMSINSTAL is doing.
- Notes, suggestions, and restrictions about various parts of the upgrade.
- Status of the upgrade.
- Licensing. Once you have registered your license during the post-upgrade procedure, the licensing messages will no longer be displayed.

Read these messages carefully to decide whether or not you need to interrupt the upgrade procedure.

8 Interrupting the Upgrade

An interruption is allowed before proceeding to Phase 1. The following message is displayed:

```
Do you want to continue? (Y/N):
```

To interrupt the upgrade, follow this procedure:

a. Enter N and press RETURN. The * *Products:* prompt is displayed.

b. To return to the DCL level, enter CTRL/Z.

c. To restart the upgrade, invoke VMSINSTAL again. The configuration of your system determines the point at which the upgrade resumes.

If you want to continue, follow this procedure:

a. Enter Y and press RETURN. The current date and time are displayed.

b. If the date and time are correct, enter YES to the prompt. If they are incorrect, enter NO. Then enter the correct date and time.

c. Once the date and time are verified, additional questions are displayed. Answer each one. Then begin Phase 1 (see Section 3.2).

3.2 Upgrade Phase 1

Separate sets of instructions are provided for Phase 1 procedures, depending on the type of VAX computer that you are upgrading. Use only one set of instructions for the VAX computer that you are upgrading. Refer to the following table for the correct section for your system.

VAX Computer	Section
VAX–11/750	Section 3.2.1
VAX 8200, 8250, 8300, 8350	Section 3.2.2
VAX 8520, 8550, 8700, 8800	Section 3.2.3
VAX–11/725, VAX–11/730, VAX–11/780, VAX–11/785, VAX 8600, or VAX 8650	Section 3.2.4
MicroVAX and VAXstation systems	Section 3.2.5

3.2.1 Upgrade Phase 1—for VAX–11/750 Computer

Follow this Phase 1 procedure for a VAX–11/750 computer.

1 The upgrade procedure asks you about the type of upgrade you are performing.

```
Are you performing a Rolling Upgrade? (Y/N):
```

If you are doing a standalone or concurrent upgrade, enter N and press RETURN.

If you are doing a rolling upgrade, enter Y and press RETURN. For more information on rolling upgrades, refer to Section 4.2.

2 To ensure system security, the upgrade procedure requires you to change the passwords for the SYSTEM, SYSTEST, and FIELD accounts before continuing. Passwords must be at least eight characters. Use unique passwords. Do not use passwords that are the same as the user name, that are printed in a document, or that might be commonly used. For example, do not use MANAGER as the password for SYSTEM. The upgrade procedure checks the passwords and prompts you to change them. You have the option of changing the passwords, disabling the accounts, or both.

3 The upgrade procedure turns off the disk quotas on the system disk and removes directory entries that point to nonexistent files.

You can boot from the TU58 or directly from the disk. If you are using a CI750, answer YES in response to the prompt. If you are booting directly from a local system disk, skip to step 5.

4 The procedure changes the default bootstrap command procedure (DEFBOO.CMD) on the console volume to boot from [SYSF].

a. Position the BOOT DEVICE switch on the VAX computer control panel to position A.

NOTE: Be sure the RECORD switch on the console TU58 cassette is set to allow writing of the cassette.

b. When prompted, insert the backup copy of the console TU58 in the console drive. The procedure modifies some of the files on the console TU58.

c. The procedure prompts you to set DEFBOO.CMD to boot the backup copy of the system disk (assuming you have not done so previously). Enter the name of the boot file to copy to DEFBOO.CMD, using the *ddu*BOO.CMD format.

For example, if the system disk is in DBA1, enter DB1BOO.CMD. If the system disk is controlled by either an HSC or a UDA, you must use the revised version of CIBOO.CMD or DUABOO.CMD that boots the disk you want to upgrade. Select a command procedure that can boot the system disk without any operator intervention (for example, registers R0 through R5 must be correctly initialized by the command procedure).

Do not specify a conversational boot command file. The upgrade kit is set with parameters that boot any system. The procedure builds a conversational boot file (UPGGEN.CMD) that boots from [SYSF]. Use the conversational boot command file only for the situations prescribed by the procedure.

NOTE: Leave the console TU58 in its drive for the remainder of the upgrade procedure.

5 The upgrade procedure builds the upgraded system in system root [SYSF] so that the current system is available if needed. Then the upgrade procedure cleans up directories on the system disk, removes installed images, updates the console volume, and displays a series of messages that indicate the state of the upgrade.

6 After several minutes, the upgrade procedure indicates that it will shut down to reboot the partially installed VMS Version 5.0 system. The procedure should reboot the system automatically. If necessary, you can reboot the system manually, as follows:

- If booting from the console TU58, press CTRL/P to put the system in console mode and enter the following command:

```
>>> B
```

- If booting from the system disk, press CTRL/P to enter console mode and enter the following command:

```
>>> B/F0000000 ddcu
```

where:

ddcu represents the system disk.

7 If the system fails to boot in a CI-only VAXcluster environment because of insufficient nonpaged dynamic memory, use the conversational boot described in this step. Otherwise, go to step 8.

To invoke UPGGEN.CMD when booting from a disk, press CTRL/P to enter console mode and enter the following command:

```
>>> B/F0000001 ddcu
```

where *ddcu* represents the system disk.

If booting from the TU58, enter the following commands:

```
>>> B/800 DDA0
BOOT58> @UPGGEN.CMD
```

After a short wait, SYSBOOT prompts for parameter changes. Enter the following commands:

```
SYSBOOT> SET NPAGEDYN 300000
SYSBOOT> CONTINUE
```

8 During the shutdown, the following two messages are displayed:

```
%SHUTDOWN-I-STOPQUEMAN, the queue manager will now be stopped.

%SYSTEM-F-DEVOFFLINE, device is not in configuration or not available.
```

The first message is an informational message. Ignore the second error message. When the system reboots, a message similiar to the following is displayed:

```
VAX/VMS Version nnn
```

Phase 1 of the upgrade for VAX–11/750 computers is complete. The rest of the upgrade does not require an operator if you booted from a TU58. However, if you boot directly from the system disk, you must reboot manually each time the system shuts down for Phases 2 through 4.

To continue the upgrade procedure, go to Section 3.3.

3.2.2 Upgrade Phase 1—for VAX 8200, 8250, 8300, and 8350 Computers

Complete this Phase 1 procedure for a VAX 8200, 8250, 8300, and 8350 computer.

1 The upgrade procedure asks you about the type of upgrade you are performing.

```
Are you performing a Rolling Upgrade? (Y/N):
```

If you are doing a standalone or concurrent upgrade, enter N and press RETURN.

If you are doing a rolling upgrade, enter Y and press RETURN. For more information on rolling upgrades, refer to Section 4.2.

2 To ensure system security, the upgrade procedure requires you to change the passwords for the SYSTEM, SYSTEST, and FIELD accounts before continuing. Passwords must be at least eight characters. Use unexpected and unique passwords. Do not use passwords that are the same as the user name, that are printed in a document, or that might be commonly used. For example, do not use MANAGER as the password for SYSTEM. The upgrade procedure checks the passwords and prompts you to change them. You have the option of changing the passwords, disabling the accounts, or both.

3 The upgrade procedure turns off the disk quotas on the system disk and removes directory entries that point to nonexistent files.

You can boot from the RX50 or directly from the disk. If using a CI-only device, answer YES to this question. If you are booting directly from a local disk, go to step 5.

4 The procedure changes the default bootstrap command procedure (DEFBOO.CMD) on the console volume to boot from [SYSF].

   a. When prompted, insert the console RX50 in the console drive. The procedure modifies some of the files on the console RX50.

   b. The procedure prompts you to set DEFBOO.CMD to boot the backup copy of the system disk (assuming you have not done so previously). Enter the name of the boot file to copy to DEFBOO.CMD, using the *ddd*BOO.CMD format.

      If the system disk is controlled by an HSC disk, you must use the revised version of CIBOO.CMD that you built when you first installed the VMS operating system. Select a command procedure that can boot the system disk without any operator intervention (for example, registers R0 through R5 must be correctly initialized by the command procedure).

      Do not specify a conversational boot command file. The upgrade kit is set with parameters that boot any system. The procedure builds a conversational boot file (UPGGEN.CMD) that boots from [SYSF]. Use the conversational boot command file only for the situations prescribed by the procedure.

NOTE: Leave the console RX50 in its drive for the remainder of the upgrade procedure.

5 The upgrade procedure builds the upgraded system in system root [SYSF] so that the current system is available if needed. Then the procedure updates the console volume, cleans up directories on the system disk, removes installed images, and displays a series of messages indicating the state of the upgrade.

6 After several minutes, the upgrade procedure indicates that it will shut down to reboot the partially installed VMS Version 5.0 system. The procedure should reboot automatically. If necessary, you can manually reboot the upgraded system as follows:

- If booting from the console RX50, press CTRL/P to put the system in console mode and enter the following command:

  ```
  >>> B
  ```

- If booting from the disk, press CTRL/P to enter console mode and enter the following command:

  ```
  >>> B/R5:F0000000 ddnu
  ```

  — *dd* refers to the device name.

  — *n* refers to the VAXBI node number n.

  — *u* refers to the device unit number.

7 If the system fails to boot in a CI-only environment because of insufficient nonpaged dynamic memory, use the conversational boot described in this step. Otherwise, go to step 8.

If booting from a disk, press CTRL/P and enter the following command:

```
>>> B/R5:F0000001 ddnu
```

- *dd* refers to the device name.

- *n* refers to the VAXBI node number.
- *d* refers to the device unit number.

If booting from the console RX50, enter the following commands:

```
>>> B/R5:800 CSA1
BOOT58> @UPGGEN.CMD
```

After a short wait, SYSBOOT prompts for parameter changes. Enter the following commands:

```
SYSBOOT> SET NPAGEDYN 300000
SYSBOOT> CONTINUE
```

8 During the shutdown, the following two messages are displayed:

```
%SHUTDOWN-I-STOPQUEMAN, the queue manager will now be stopped.

%SYSTEM-F-DEVOFFLINE, device is not in configuration or not available.
```

The first message is an informational message. Ignore the second message. When the system reboots, a message similar to the following is displayed:

```
VAX/VMS Version nnn
```

Phase 1 of the upgrade for the VAX 8200, 8250, 8300, and 8350 computers is complete. The rest of the upgrade does not require an operator. Make sure that DIGITAL Field Service has set the default boot device in EEPROM to your current system disk. Then set the lower key switch to autostart. Rebooting is automatic and does not require user intervention.

To continue the upgrade procedure, go to Section 3.3.

3.2.3 Upgrade Phase 1—for VAX 8530, 8550, 8700, or 8800 Computers

This section describes the first phase of the upgrade for users who are upgrading a VAX 8530, 8550, 8700, or 8800 computer.

1 The upgrade procedure asks you about the type of upgrade you are performing.

```
Are you performing a Rolling Upgrade? (Y/N):
```

If you are doing a standalone or concurrent upgrade, enter N and press RETURN.

If you are doing a rolling upgrade, enter Y and press RETURN. For more information on rolling upgrades, refer to Section 4.2.

2 To ensure system security, the upgrade procedure requires that you change the passwords for the SYSTEM, SYSTEST, and FIELD accounts before continuing. Passwords must be at least eight characters. Use unexpected and unique passwords. Do not use passwords that are the same as the user name, that are printed in a document, or that might be commonly used. For example, do not use MANAGER as the password for SYSTEM. The upgrade procedure checks the passwords and prompts you to change them. You have the option of changing the passwords, disabling the accounts, or both.

3 The upgrade procedure turns off the disk quotas on the system disk and removes directory entries that point to nonexistent files.

4 The procedure builds the upgraded system in system root [SYSF] so that the current system is available if needed.

5 The upgrade procedure cleans up directories on the system disk, removes installed images, and displays a series of messages indicating the state of the upgrade.

6 After several minutes, the upgrade procedure indicates that it will shut down to reboot the partially installed VMS Version 5.0 system. The procedure should reboot the system automatically. If necessary, you can reboot the system manually, as follows:

   a. Press CTRL/P.

   b. Enter the following commands:

```
>>> H
>>> CLEAR RESTART_FLAGS
>>> B
```

7 If the system fails to boot in a CI-only environment because of insufficient nonpaged dynamic memory, use the conversational boot described in this step. Otherwise, go to step 8.

   To invoke UPGGEN.COM, enter the following commands:

```
$  CTRL/P
>>> H
>>> CLEAR RESTART_FLAGS
>>> @dddGEN.COM

SYSBOOT> SET NPAGEDYN 300000
SYSBOOT> CONTINUE
```

8 During the shutdown, the following two messages are displayed:

```
%SHUTDOWN-I-STOPQUEMAN, the queue manager will now be stopped.

%SYSTEM-F-DEVOFFLINE, device is not in configuration or not available.
```

   The first message is an informational message. Ignore the second message.

   When the system reboots, a message similar to the following is displayed:

```
VAX/VMS Version nnn
```

Phase 1 of the upgrade for VAX 8530, 8550, 8700, and 8800 computers is complete. The upgrade procedure does not require an operator until the completion of Phase 4.

To continue the upgrade procedure, go to Section 3.3.

3.2.4 Upgrade Phase 1—for VAX–11/725, VAX–11/730, VAX–11/780, VAX–11/785, VAX 8600, and VAX 8650 Computers

This section describes the first phase of the upgrade for users who are upgrading a VAX–11/725, VAX–11/730, VAX–11/780, VAX–11/785, VAX 8600, or 8650 computer.

1 The upgrade procedure asks you about the type of upgrade you are performing.

```
Are you performing a Rolling Upgrade? (Y/N):
```

If you are doing a standalone or concurrent upgrade, enter N and press RETURN.

If you are doing a rolling upgrade, enter Y and press RETURN. For more information on rolling upgrades, refer to Section 4.2.

2 To ensure system security, the upgrade procedure requires that you change the passwords for the SYSTEM, SYSTEST, and FIELD accounts before continuing. Passwords must be at least eight characters. Use unique passwords. Do not use passwords that are the same as the user name, that are printed in a document, or that might be commonly used. For example, do not use MANAGER as the password for SYSTEM. The upgrade procedure checks the passwords and prompts you to change them. You have the option of changing the passwords, disabling the accounts, or both.

3 The upgrade procedure now turns off the disk quotas on the system disk and removes directory entries that point to nonexistent files.

NOTE: If you are upgrading a VAX 8600 or VAX 8650, note that all console command procedures referenced in the following steps have COM file types, not CMD file types.

4 The procedure builds the upgraded system in system root [SYSF], so that the current system is available if needed. The procedure changes the default bootstrap command procedure (DEFBOO.CMD) on the console volume to boot from [SYSF].

NOTE: If the console volume is a TU58, be sure that the RECORD switch on the cassette is set to allow writing of the TU58.

5 When prompted, insert the console volume in the console drive. The procedure modifies some of the files on the console volume.

NOTE: Leave the console volume in its drive for the remainder of the upgrade procedure.

6 The procedure prompts you to set DEFBOO.CMD to boot the backup copy of the system disk. Enter the name of the boot file to copy to DEFBOO.CMD, using the *ddd*BOO.CMD format.

For example, if the system disk is in DBA1, enter DB1BOO.CMD. If the system disk is controlled by either an HSC or a UDA, you must use the revised version of CIBOO.CMD or DUABOO.CMD that you built when you first installed VMS on the HSC- or the UDA-controlled system disk. Select a command procedure that can boot the system disk without any operator intervention (for example, registers R0 through R5 must be correctly initialized by the command procedure).

Do not specify a conversational boot command file. The upgrade kit is set with parameters that boot any system. The procedure builds a conversational boot file (UPGGEN.CMD) that boots from [SYSF]. Use the conversational boot file only for situations prescribed by the procedure.

7 The procedure updates the console volume and displays messages that describe the state of the build. After several messages, the upgrade describes the correct method for handling a shutdown or reboot failure. Note the following information about these failures:

- On a VAX–11/730 system, the microcode can be reloaded. If the system fails to boot correctly, turn the system off and then on again, so that the console reloads the microcode from the TU58.
- If the system fails to boot in a CI-only environment because of insufficient nonpaged dynamic memory, use the conversational boot command procedure UPGGEN.CMD to increase NPAGEDYN as follows:

```
>>> @UPGGEN.CMD
SYSBOOT>  SET NPAGEDYN 300000
SYSBOOT>  CONTINUE
```

8 When the system reboots, a message similiar to the following is displayed:

```
VAX/VMS Version nnn
```

During the shutdown, the following two messages are displayed:

```
%SHUTDOWN-I-STOPQUEMAN, the queue manager will now be stopped.

%SYSTEM-F-DEVOFFLINE, device is not in configuration or not available.
```

The first message is an informational message. Ignore the second message.

To continue the upgrade, go to Section 3.3.

3.2.5 Upgrade Phase 1—for MicroVAX and VAXstation Computers

This section describes the first phase of the upgrade for users who are upgrading a MicroVAX and VAXstation computer.

1 The upgrade procedure asks you about the type of upgrade you are performing.

```
Are you performing a Rolling Upgrade? (Y/N):
```

Enter N and press RETURN. You cannot perform a rolling upgrade on a MicroVAX or VAXstation computer.

2 To ensure system security, the upgrade procedure requires that you change the passwords for the SYSTEM, SYSTEST, and FIELD accounts before continuing. Passwords must be at least eight characters. Use unique passwords. Do not use passwords that are the same as the user name, that are printed in a document, or that might be commonly used. For example, do not use MANAGER as the password for SYSTEM. The upgrade procedure checks the passwords and prompts you to change them. You have the option of changing the passwords, disabling the accounts, or both.

3 The procedure automatically converts a private system disk to a cluster common system disk.

4. The procedure turns off disk quotas on the system disk and removes directory entries that point to nonexistent files. Ignore any messages about disk quotas being disabled.
5. The procedure stops OPCOM and the error formatter (ERRFMT).
6. The procedure cleans up directories and removes installed images.
7. The procedure purges all accounting data files, operator logs, and the directory SYSERR.
8. The procedure deletes all JNL files in the root directory and in all its subdirectories.
9. The procedure builds the directory tree [SYSF] and deletes all the old operating system files that will not be needed if the system needs to be rebooted during Phase I of the upgrade.
10. The upgrade procedure restores the VMS Version 5.0 required save set.
11. The procedure purges the page, swap, dump, and authorization files.
12. The procedure displays messages stating that it is shutting down the system, that the system disk and distribution medium must not be moved during the upgrade, and that SYSGEN parameters must not be changed while the system reboots.

    When the system shutdown has completed, you must halt the system. If your VAX computer has a HALT button, press it once to halt the system. Then press it again to make sure it is no longer halted. If your VAX computer has no HALT button, press BREAK on your console terminal. Then reboot the system by entering the following command:

    ```
    >>> B/F0000000 ddcu
    ```

    ddcu is the name of the system disk.
13. If your system is a MicroVAX I, it prompts you to enter the date and time, and continues automatically with Phase 2. To continue with the upgrade procedure, go to Section 3.3.

If a MicroVAX or VAXstation fails to reboot in Phases 2 through 4, halt the system. Enter the following command, where *ddcu* is the name of the system disk:

```
>>> B/F0000000 ddcu
```

3.3 Upgrade Phase 2

At the beginning of Phase 2, messages about licenses are displayed. You can ignore these messages, because they disappear once you register your license (refer to Chapter 6).

Phase 2 restores the rest of the VMS Version 5.0 files in the LIBRARY and OPTIONAL save sets. First the LIBRARY save set is restored, then the OPTIONAL save set is restored.

If you are upgrading an RD52 system disk, the LIBRARY and OPTIONAL save sets are not restored.

If you are using either RL02, RC25, or RK07 distribution media, the save sets will be restored from a multivolume set, requiring that you change disks. In a standard VMS configuration with RA60, CDROM, or magnetic tape distribution media, no media change is required.

You can remove the distribution kit at the end of Phase 2, if you prefer.

3.4 Upgrade Phase 3

During Phase 3, the upgrade procedure completes the following tasks:

- Merges the VMS-distributed files that are commonly edited by system managers with new VMS files.
- The upgrade procedure merges all the miscellaneous user files in the old system directories into a new set of system directories, temporarily called SYSF.SYSEXE, SYSF.SYSMGR, SYSF.SYSLIB, and so on. The amount of time the merge takes depends on the number of user files.
- Removes the directory entries for page, swap, dump, and authorization files from the old directory tree.
- Deletes all the remaining accounting data files, operator logs, and all files in the SYSERR directory.

3.5 Upgrade Phase 4

If you are upgrading a VAX 8530, 8550, 8700, or 8800 computer, refer to Section 3.5.1. If you are upgrading a MicroVAX or VAXstation computer, refer to Section 3.5.2. If you are upgrading any other VAX computer, read this section.

During this phase of the upgrade, the procedure modifies the site-specific console volume to restore the original boot files. The original boot files will allow reboot of the complete VMS Version 5.0 system. Be sure that the site-specific console volume modified in Phase 1 is still in the console drive (CSA1: for VAX–11/780, 11/785, VAX 8600, 8650 (original RL02), 8200, 8250, 8300, 8350 and 11/750 systems; CSA2: for VAX–11/730 and 11/725 systems).

When the modification is completed, the following message is displayed:

```
System shutting down to boot the complete VMS Version 5.0 system.
```

Leave the site-specific console volume in the console drive. The system disk must also remain where it is for the next phase of the upgrade to proceed. The system attempts an automatic reboot after the shutdown.

If the system fails to boot because of insufficient nonpaged dynamic memory, you must use a standard conversational bootstrap to increase the NPAGEDYN (nonpaged dynamic memory) system parameter as described in previous sections.

If needed, invoke the conversational boot command procedure for your system disk. For example, if the system disk is on the first MASSBUS device, invoke DB0GEN; if the system is on the first UDA device, DU0GEN, and so forth.

When the boot is completed, a message similar to the following is displayed:

```
VAX/VMS Version nnn
```

3.5.1 Upgrade Phase 4—VAX 8530, 8550, 8700, 8800 Computers

The Phase 4 upgrade procedure for a VAX 8530, 8550, 8700, or 8800 computer is as follows:

During this phase, the procedure fixes back links for system directories and shuts down the system. After the procedure shuts down the system, restore the original copies of DEFBOO and *ddd*GEN to boot from your original system root.

In step 15 of the pre-upgrade procedure (Chapter 2), you made copies of the original boot files, DEFBOO.SAV and *ddd*GEN.SAV.

Rename the files to their original names, as follows:

```
>>> EXIT
$ RENAME DEFBOO.SAV DEFBOO.COM
$ RENAME dddGEN.SAV dddGEN.COM
```

After restoring the original files, reboot the system. When the boot completes, the system displays a message similar to the following:

```
VAX/VMS Version nnn
```

3.5.2 Upgrade Phase 4—for MicroVAX and VAXstation Computers

The Phase 4 upgrade procedure for a MicroVAX and VAXstation system is as follows:

1 The following system message is displayed on your console device:

```
Continuing with VAX/VMS V5.0 Upgrade Procedure.

    Upgrade Phase 4  10-FEB-1988  16:00
```

2 The upgrade procedure corrects the back links for the system directories. This step requires only a few seconds. The procedure displays a message when it has completed.

3 The procedure shuts down the system.

When the shutdown has completed, halt the system. Then reboot the system by entering the following command:

```
>>> B ddcu
```

ddcu is the name of the system disk.

3.6 Upgrade Phase 5

During Phase 5, the upgrade procedure performs the following:

1. Deletes the temporary [SYSF] directory tree.
2. Cleans up files used only during the upgrade procedure.
3. Converts the DECnet proxy database to VMS Version 5.0 format, if it exists. The conversion is not done if the file SYS$SYSTEM:NETUAF.DAT does not exist or if NETUAF.DAT is not in SYS$SYSTEM. If the conversion is not done here, you must run it manually during the post-upgrade procedure. See Chapter 5, "Converting DECnet Proxy Database".
4. Converts all VAXVMSSYS.PAR parameter files found in each private system root (for a shared system disk) to VMS Version 5.0 format.
5. Displays messages listing tasks that you might want to perform after the upgrade procedure is completed. These tasks include the following:
   - Decompressing the system libraries (refer to step 12 in Chapter 5).
   - Building standalone BACKUP (refer to the installation and operations guide for your VAX computer).
   - Editing system files (refer to step 5 in Chapter 5).
   - Purging files (refer to step 11 in Chapter 5).

3.7 Upgrade Phase 6

During Phase 6, the upgrade procedure configures all devices on the system. It also applies the mandatory update.

The mandatory update is provided on separate media except for CDROMs, where it is on the same media.

The mandatory update on separate media has one of the following labels:

- VMS V5.0 BIN RX01 Mandatory Update
- VMS V5.0 BIN 16MT9 Mandatory Update
- VMS V5.0 BIN TU58 Mandatory Update
- VMS V5.0 BIN TK50 Mandatory Update
- VMS V5.0 BIN RX50 Mandatory Update

Follow this procedure in Phase 6:

1. The procedure prompts you to mount the distribution volume. Make sure the media is write-protected and place the mandatory update volume in the device.
2. The procedure prompts you for the name of the device. Enter the device name in the *ddcu* format.
3. The procedure asks you if you are ready. Enter Y for YES.

4 The procedure asks if you want to purge files replaced by the upgrade. Enter Y for YES.

5 Once the mandatory update is applied for a standalone system or concurrent upgrade, you will be running B5.0, an intermediary version of the full VMS Version 5.0. To complete the upgrade to full Version 5.0, you must convert the queue and MAIL database files to the Version 5.0 format.

Once the mandatory update is applied in a rolling upgrade, A5.0, an intermediary version of VMS Version 5.0, is running. To complete the rolling upgrade, you must convert to B5.0 and then to Version 5.0. You cannot complete the rolling upgrade until the mandatory update is applied.

If you are installing Version 5.0 on a new VAX computer, the mandatory update displays a message about running in a mixed-version cluster and whether you want to continue. See Section 4.2.

Note: When upgrading an RD52-based system, the LIBRARY and OPTIONAL save sets are not restored. Because the mandatory update may repair files contained in these save sets, you should do the following:

- **Ignore any "file not found" messages that appear during the application of the mandatory update**
- **If you restore files from one of these save sets at another time, reapply the mandatory update after restoring the files. You can do this by entering the following command:**

```
$ @SYS$UPDATE:VMSINSTAL VMSMUP
```

6 The procedure runs AUTOGEN to calculate the new SYSGEN parameters for your configuration. After AUTOGEN is complete, the system automatically reboots.

To complete the upgrade, go to Chapter 5.

4 Upgrading Clusters

The cluster upgrade you use depends on whether or not you want to maintain availability of the cluster. The two types of cluster upgrades are as follows:

- Concurrent upgrade—Shuts down the entire cluster and applies the upgrade to each system disk. The cluster does not become operational again until all system disks are upgraded and the nodes are rebooted. Refer to Section 4.1.
- Rolling upgrade—Applies the upgrade to each system disk while maintaining operation of the cluster. The cluster runs with mixed versions of VMS. Full VMS Version 5.0 capabilities are not available until the entire cluster is upgraded. Refer to Section 4.2.

NOTE: Clusters using multiple system disks require special licensing once you have completed the upgrade. Refer to Chapter 6.

4.1 Upgrading a VAXcluster: Concurrent Upgrade

A concurrent upgrade is performed by shutting down the entire cluster and applying the upgrade to each system disk. Then each node in the cluster is booted to start running the upgraded version of the VMS operating system. All systems in the cluster are unavailable while a concurrent upgrade is being performed.

This section describes the concurrent upgrade procedure.

1 Noting Current SYSGEN Parameter

Note the current value for the VOTES parameter on each node. You will restore the VOTES value after you complete the upgrade.

Enter the following commands:

```
$RUN SYS$SYSTEM:SYSGEN
SYSGEN> USE CURRENT
SYSGEN> SHOW VOTES
SYSGEN> EXIT
```

2 Shutting Down the Cluster

Shut down the entire cluster, using your site's standard shutdown procedure, as follows:

```
$ @SYS$SYSTEM:SHUTDOWN
```

3 Selecting a System Disk

If you have only one system disk for your cluster, go to step 4.

If you have more than one system disk, select one to upgrade.

4 Booting a Single System

Perform a conversational boot of a single VAX system from the system disk you will be upgrading. Refer to the installation and operations guide for your VAX computer for a discussion of the conversational boot procedure.

Set the votes and quorum values to 1, as follows:

```
SYSBOOT> USE CURRENT
SYSBOOT> SET VOTES 1
SYSBOOT> SET QUORUM 1
SYSBOOT> CONTINUE
```

5 Preparing for the Upgrade

Perform the complete pre-upgrade procedure as described in Chapter 2.

NOTE: If you are upgrading a shared system disk (SYS$COMMON), the new files will be the most recent versions in SYS$COMMON, they will not be in the system specific root.

6 Upgrading the System Disk

Perform a complete single system upgrade. Follow the procedure described in Chapter 3. You also complete the post-upgrade procedure, described in Chapter 5.

7 Upgrading Other System Disks

If you have only one system disk, skip this step and go to step 8.

If you have more than one system disk, shut down the system that you just used for the upgrade. Then repeat steps 3 through 6 for each system disk in the cluster. When you have upgraded each system disk, go to step 8.

8 Rebooting Nodes

Reboot each node in the VAXcluster.

If one of the nodes is a VAX 8530, 8550, 8700 or 8800 computer, you must update the console media for each one. Follow step 9 in Chapter 5.

If one of the nodes is a VAX–11/750, 11/780, 11/785, 8200, 8250, 8300, 8350, 8600, or 8650 computer, you must update the console media for each one. Follow step 10 in Chapter 5.

4.2 Upgrading a VAXcluster: Rolling Upgrade

For a rolling upgrade, you upgrade each system disk individually, allowing old and new versions of the VMS operating system to exist together temporarily in the same cluster. Clusters running two versions of VMS are called mixed-version clusters. Because rolling upgrades allow mixed-version clusters, you maintain availability of the cluster during the upgrade. To accommodate mixed-version clusters, however, the rolling upgrade requires upgrading system disks to two intermediate versions of Version 5.0, called A5.0 and B5.0.

DIGITAL recommends that all nodes in a cluster run the same version of the VMS operating system. Mixed-version clusters are supported *only* for the purpose of incrementally upgrading the entire cluster to Version 5.0.

NOTE: Rolling upgrades are supported for CI-only clusters.

A rolling upgrade is not applicable when all systems boot from a single system disk. When all systems boot from a single system disk, perform a concurrent upgrade. Refer to Section 4.1.

4.2.1 VMS Version 5.0 Rolling Upgrade Procedure—Overview

VMS Version 5.0 includes several major changes that affect the rolling upgrade procedure. These changes include the following:

- Changes in communications protocols of various components.
- A new format for the batch/print queue file (JBCSYSQUE.DAT).
- A new format for the shared MAIL database (VMSMAIL.DAT). Also, the name of the shared MAIL database has changed to VMSMAIL_PROFILE.DATA.
- A new format for the DECnet proxy database. Also, the name of the proxy database has changed to NETPROXY.DAT.

A rolling upgrade covers several states:

a. Before the upgrade—All nodes must be running Version 4.7.

b. Mixed Version 4.7/A5.0 state—Upgrade nodes from Version 4.7 to an intermediary of Version 5.0, called A5.0. The cluster can run with mixed versions of 4.7 and A5.0. While running both Version 4.7 and A5.0, the cluster might operate below optimum performance, and without full Version 5.0 capabilities.

The A5.0 version is essentially a full Version 5.0, except that it does not include most of the enhancements to the batch/print facility and the Mail Utility.

c. Complete A5.0 state—All nodes in the cluster are running A5.0.

d. Mixed A5.0/B5.0—Convert all A5.0 nodes to an intermediary of Version 5.0, called B5.0. Before converting to B5.0, all nodes must be running A5.0. B5.0 nodes can communicate with A5.0 nodes, but not with Version 4.7 nodes.

e. Complete B5.0 state—All nodes are running B5.0.

f. Mixed B5.0/Version 5.0 state—Before converting to Version 5.0, all nodes must be running B5.0. Convert the queue file and MAIL database on B5.0 nodes to the Version 5.0 format.

g. Complete Version 5.0 state—All nodes running Version 5.0. The cluster runs with the full capabilities of VMS Version 5.0 at optimum performance.

NOTE: Version 4.7 cannot coexist in a cluster with B5.0 or Version 5.0. Version A5.0 cannot coexist in a cluster with Version 5.0.

4.2.2 Limitations of Mixed-Version Clusters

Although the rolling upgrade allows mixed versions of the VMS operating system, there are drawbacks to the mixed-version cluster. Mixed-version clusters might operate at less than optimum performance and they do not have full Version 5.0 capabilities. Limited functions include the following:

- The queue file remains in Version 4.7 format. Most of the Version 5.0 enhancements to the job controller are not available. For A5.0, entry number references (DELETE/ENTRY=*n queue_name*) require the queue name until the queue file is converted. For A5.0 and B5.0, the SHOW ENTRY command is not available.
- The shared MAIL database remains in Version 4.7 format. Some Version 5.0 enhancements to MAIL are not available, including these new commands: SET CC_PROMPT, SET QUEUE, SET FORM.
- Version 4.7 nodes use the Version 4.7 format of the DECnet proxy database. A5.0 nodes use the Version 5.0 format. As a result, you must set up proxy accounts on both Version 4.7 and A5.0 nodes.
- Version 4.7 MSCP server and the Version 5.0 disk class driver are not compatible. As a result, mixed-version clusters are prevented from serving disks on any Version 4.7 system to systems in the cluster running A5.0.
- Version 4.7 nodes use the old format for TPU section files. A5.0 nodes use a new format. The file name for the Version 5.0 default section file is TPU$SECTION.TPU$SECTION.

  You must define logical names so that old logical names identify Version 4.7 files and new logical names identify Version 5.0 files. The following table provides the logical names for Version 4.7 and Version 5.0 TPU files:

File	Version 4.7 Name	Version 5.0 Name
Command File	TPUINI	TPU$COMMAND
Section File	TPUSECINI	TPU$SECTION

  You must recompile customized section files on A5.0 systems.

  If a node running Version 4.7 crashes while you are editing a file and the terminal server failover moves you to an A5.0 node, TPU produces a series of messages about Version 5.0 changes to the section file format. You can recover the file only by invoking the editor from a Version 4.7 node. Similarly, if you are editing a file on an A5.0 node and a terminal server failover moves you to a Version 4.7 node, you can recover the file only from an A5.0 node.

  Version 5.0 does not supply a copy of EDTSECINI. If you want to use this editor on A5.0 nodes, you must compile a new copy using Version 4.7 source files.
- On Version 4.7 systems, the SHOW CLUSTER/CONTINUOUS command truncates the hardware type (HW_TYPE) field. As a result, you might not be able to determine the hardware type of other nodes in the cluster.
- A problem with the $GETLKI system service occurs on Version 4.7 nodes, and sometimes occurs on A5.0 nodes.

The $GETLKI system service might not report user buffer overflow for item codes LKI$_LOCKS, LKI$_BLOCKEDBY, and LKI$_BLOCKING as documented in the *VMS System Services Reference Manual*. When $GETLKI is used with an item descriptor specifying any of these item codes, it might not set bit 31 of the **return length address** in the item descriptor when the user-supplied buffer is too small to hold the requested data.

To determine whether bit 31 of the returned length address is not indicating when the user-supplied buffer is too small, check the low and high words of the returned length address. If the sum of the two is greater than the size of the user buffer supplied, bit 31 is not indicating an overflow.

This problem disappears when all nodes run Version 5.0.

- The method of calculating LAT service ratings for nodes in a cluster is different in VMS Version 5.0 from Version 4.7. You will notice that Version 5.0 nodes in your cluster have substantially lower ratings than nodes that are running Version 4.7. If all nodes in your cluster offer a common cluster LAT service name (for example, CLUSTER), and most of your terminal server users connect to the cluster by using the "CONNECT CLUSTER" command, most LAT connections are going to the Version 4.7 nodes.
- The following layered products are not supported for mixed clusters:
  - A-to-Z Base System for MicroVAX, Version 2.6
  - A-to-Z Business Graphics for MicroVAX, Version 2.6
  - A-to-Z Database Manager for MicroVAX, Version 2.6
  - A-to-Z Developer's Kit for MicroVAX, Version 2.6
  - A-to-Z Electronic MAIL for MicroVAX, Version 2.6
  - A-to-Z Word Processing for MicroVAX, Version 2.6
  - ALL-IN-1, Version 2.2
  - DECnet/SNA VMS Distributed Host Command Facility, Version 1.1
  - High Performance Workstation Software, Version 1.1
  - Personal Computing Systems Architecture Software, Version 2.0
  - Remote Bridge Management Software, Version 1.0
  - VAX 3271 Emulator, Version 2.5
  - VAX APL, Version 3.1
  - VAX DEC/SHELL, Version 2.1
  - VAX DECrad, Version 4.0
  - VAX DECscan MicroVMS and ELN Bitbus Drivers, Version 2.0
  - VAX DSM, Version 4.1
  - VAX EDCS, Version 1.1A
  - VAX FORTRAN, Version 5.0
  - VAX Message Router, Version 3.0

- VAX Message Router X.400 Gateway, Version 2.1
- VAX Message Router/P Gateway, Version 1.1
- VAX Message Router/S Gateway, Version 1.1
- VAX OSI Application Kernel, Version 2.2
- VAX OSI Transport Service, Version 2.0
- VAX Pascal, Version 3.8
- VAX Performance and Coverage Analyzer, Version 2.0
- VAX Rdb/ELN, Version 2.1
- VAX SPM, Version 3.2, see documentation for restrictions
- VAX Storage Library System, Version 1.0
- VAX TEAMDATA, Version 1.2
- VAX-11 RSX, Version 2.4
- VAXcluster Console System, Version 1.2
- VAXLAB Software Library, Version 1.3
- VMS Workstation Software, Version 4.0
- VMS/SNA, Version 1.3

• Because Version 5.0 contains major functional changes, some customer applications might not work in the mixed-version cluster. Applications that contain privileged code linked against Version 4.7 systems will most likely fail.

4.3 Performing the Version 5.0 Rolling Upgrade

To complete a rolling upgrade, follow the procedures in these sections:

1. Upgrade the cluster to A5.0—refer to Section 4.3.1.
2. Convert the cluster to B5.0—refer to Section 4.3.2.
3. Convert the cluster to full VMS Version 5.0—refer to Section 4.3.3.

Before you begin the rolling upgrade, all nodes in the cluster must run Version 4.7. For instructions on upgrading to Version 4.7, see the *VMS Release Notes, Version 4.7*.

4.3.1 Upgrading the Cluster to A5.0

After all nodes in the cluster run Version 4.7, you can begin upgrading the cluster to A5.0. To upgrade to A5.0, apply the upgrade to each system disk (private and shared), until all nodes in the cluster run A5.0. Until all nodes run A5.0, you operate a mixed-version cluster.

To upgrade to A5.0, perform the following steps for each shared system disk and private system disk in the cluster:

1 Checking the Votes

Check the votes and make adjustments to maintain the proper quorum that allows the cluster to continue operating throughout this process. (The *VMS VAXcluster Manual* describes this procedure in detail.)

2 Backing Up the System Disk

Back up each system disk. Refer to the installation and operations guide for your VAX computer.

Make sure that the volume label on the backup copy of your system disk is different from any other system disk label in the cluster. All system disks throughout the cluster must have unique volume labels. Use the SET VOLUME/LABEL command to change the volume label, if necessary.

If you attempt to boot a backup copy that does not have a unique volume label, one of the following error messages is displayed:

```
%SYSINIT-E- error mounting system device, status = 007280B4
%SYSINIT-E- error opening or mapping F11BXQP, status = 00018272
%SYSINIT-E- message file not found, or insuffcient SPT to map it
```

3 Preparing for the Upgrade

Select a system disk to upgrade.

Complete steps 2 through 16 of the pre-upgrade procedure. Refer to Chapter 2.

4 Recovering from a Hang During Shutdown

If proper quorum is not maintained at any time during the rest of this upgrade procedure, the shutdown procedure hangs the cluster. To free the cluster, enter the following commands:

```
$ CTRL/P
>>> H
>>> D/I 14 C
>>> C
IPC> Q
IPC> CTRL/Z
```

5 Shutting Down Nodes

If you are upgrading a private system disk, go to step 6 of this section.

If you are upgrading a shared system disk perform a complete upgrade from one of the nodes that shares that system disk. For all systems on a

shared system disk except the one from which you apply the upgrade, do the following:

a. Shut down the system, using your site's standard shutdown procedure SYS$SYSTEM:SHUTDOWN.COM.

b. After you shut down a system, enter the following command on one of the remaining nodes:

```
$SET CLUSTER/QUORUM
```

This procedure allows one node to continue running from the system disk (assuming other nodes running from different system disks supply enough votes to sustain cluster quorum).

6 Upgrading System Disk

Upgrade the system disk by following the procedure in Chapter 3. Once the upgrade is complete, A5.0 is running. You also complete the post-upgrade procedure, described in Chapter 5.

NOTE: When the upgrade procedure asks if you are performing a rolling upgrade in Phase 1, enter YES. If you enter NO, you will be unable to run a mixed-version cluster.

7 Rebooting Nodes

If you are upgrading a private system disk, skip this step and go to step 8.

Reboot each node that boots from the shared system disk just upgraded. You must register a license for each node. Refer to Chapter 6 for the procedure on registering licenses.

If one of the nodes is a VAX 8530, 8550, 8700 or 8800 computer, you must update the console media for each one. Follow step 9 in Chapter 5.

If one of the nodes is a VAX–11/750, 11/780, 11/785, 8200, 8250, 8300, 8350, 8600, or 8650 computer, you must update the console media for each one. Follow step 10 in Chapter 5.

8 Upgrading Other System Disks

At this point, the cluster runs Version 4.7 and A5.0. You should now test and verify the new version before upgrading other system disks.

Repeat steps 3 through 7 as appropriate for each system disk, until each system disk is running A5.0.

Once the entire cluster is running A5.0, go to Section 4.3.2.

4.3.2 Converting the Cluster to B5.0

After all nodes in the cluster run A5.0, you can begin the conversion to B5.0. To convert the cluster to B5.0, you must apply the conversion program to each system disk in the cluster, then reboot all nodes that boot from the system disks. The conversion program updates the system version number from A5.0 to B5.0. As a result, nodes that have been converted to B5.0 can no longer communicate with Version 4.7 nodes.

Start the conversion to B5.0 only when you are committed to completing the rolling upgrade.

1 Backing Up the System Disk

Select a system disk to convert to B5.0.

Back up the A5.0 system disk. Refer to the installation and operations guide for your VAX computer for instructions.

2 Checking Votes

Check the votes and make adjustments to maintain the proper quorum that allows the cluster to continue operating throughout this process. (The *VMS VAXcluster Manual* describes this procedure.)

3 Running Conversion Program

Invoke and run the conversion program, as follows:

a. Log in to the system manager's account, SYSTEM, on the node from which you are performing the conversion. This node must boot from the system disk that you are converting.

b. To invoke the conversion program, enter the following command:

```
$ @SYS$UPDATE:VMSINSTAL VMS_ATOB050 SYS$UPDATE
```

c. VMSINSTAL displays a series of messages and prompts. If DECnet is running on your system, a message states that the network is up and running. A prompt asks you whether you want to continue. Enter Y.

d. The conversion program displays a message about running in a mixed-version cluster and asks you whether you want to continue. If all the system disks are running A5.0 or B5.0, enter Y to continue. If one of the system disks is still running Version 4.7, enter N to stop. Upgrade the system disk(s) to A5.0 (refer to Section 4.3.1). Then start from the beginning of this procedure.

The conversion program changes the VMS version number to B5.0 and shuts down the system.

4 Rebooting the System

Reboot the system from the converted system disk as follows:

```
$ CTRL/P
>>> H
>>> B
```

5 Rebooting Nodes

If you are converting a private system disk, go to step 6.

If you are converting a shared system disk, shut down and reboot each node that boots from the system disk that you converted.

6 Completing Conversion

Repeat steps 1 through 5 for each system disk in the cluster. Once you complete the conversion for all system disks, the cluster is running B5.0. Go to Section 4.3.3.

4.3.3 Converting the Cluster to VMS Version 5.0

After all nodes in the cluster run B5.0, you can convert the cluster to Version 5.0. You must apply the conversion program to each system disk in the cluster.

NOTE: **Ensure that all user disks are mounted before invoking the conversion procedure.**

While the conversion program runs, the job controller cannot process batch and print jobs.

The conversion procedure converts the primary queue and MAIL database files. If your cluster uses more than one queue or MAIL database file, you must convert the secondary queue or MAIL database files once the primary files are converted.

The queue file conversion program does the following:

- Stops the queue managers on all nodes of the cluster.
- Purges the queue file and renames it to V46_*name*.DAT.
- Creates a new queue file in the Version 5.0 format using the old queue file name.
- Restores all form and characteristic definitions to the new queue file.
- Resubmits jobs that were in the queue file when you ran the program.
  - The conversion program changes entry numbers and submission times. Job names remain the same.
  - The conversion program resubmits jobs with retained status with a holding status.
  - The conversion program resubmits with the /RESTART qualifier jobs that were submitted with the /RESTART qualifier. If a job submitted with /RESTART is executing when you run the conversion program, the job executes from the beginning when it is resubmitted; the $RESTART value is set to false. (See the SET RESTART_VALUE in the *VMS DCL Dictionary* for more information about $RESTART.)

While running the queue file conversion program, the following messages might be displayed:

- %JBCUPGRAD-W-NOTSTOPQUE, queue (queue_name) was not in stopped state.

  This message is displayed if a queue remains running during the file conversion. The conversion continues, but all queue job data may not be saved and restored.

- %JBCUPGRAD-E-JOBNOTSAVED, job (job_name) for user (user name) was in an unexpected state and will not be saved.

This message is displayed when a queue remains running during the file conversion, and a job on the queue was not submitted with the /RESTART qualifier. The conversion continues, but the job might not be saved.

The MAIL database file conversion program converts a MAIL database file to Version 5.0 format and names the file VMSMAIL_PROFILE.DATA. The converted file will be located in the same directory as the old MAIL database file. Define the logical name VMSMAIL_PROFILE in the system logical name table to specify an alternate disk and directory.

For each shared and private system disk in the cluster, perform the following steps:

1 Terminating MAIL Sessions

DIGITAL recommends that you terminate all MAIL sessions before starting the conversion program. If you are using MAIL when the conversion program runs and you enter a command that accesses the MAIL database file (for example, the SHOW or SEND command), the MAIL session is terminated, and the following message is displayed:

```
%MAIL-F-UPGRADE, mail version upgrade in progress
```

2 Invoking the Conversion Program

a. Log in to the system manager's account, SYSTEM, on the node from which you are performing the conversion. This node must boot from the system disk that you are converting.

b. Enter the following command:

```
$ @SYS$UPDATE:VMSINSTAL VMS_BTOC050 SYS$UPDATE
```

c. VMSINSTAL displays a series of messages and prompts. If DECnet is running on your system, a message is displayed telling you that the network is up and running, and asking you whether you want to continue. Enter Y.

d. The procedure displays a message about user disks and asks whether you want to continue. If all user disks are spun up and mounted, enter Y to continue. If one or more user disks are not spun up or mounted, enter N to stop. Spin up the disk(s), mount the disk(s), and start this procedure over again.

3 Specifying the Default Queue File

The conversion program needs to know the location of your queue file, if you have one. Answer the queue file prompt, as follows:

a. If the default queue file, SYS$SYSROOT:[SYSEXE]JBCSYSQUE.DAT, is the active queue file for your cluster, answer YES to this prompt:

```
Is SYS$SYSROOT:[SYSEXE]JBCSYSQUE.DAT your active queue file? YES
```

b. If the default queue file, SYS$SYSROOT:[SYSEXE]JBCSYSQUE.DAT, is not the active queue file for your cluster, perform these steps to enter the queue file for your cluster:

```
Is SYS$SYSROOT:[SYSEXE]JBCSYSQUE.DAT your active queue file? NO
What is the name of your active QUEUE file: disk:[directory]queuefile.DAT
```

Make sure you enter the disk and directory when you enter the queue file specification.

If you do not have an active queue file (you are not using batch queues or print queues), enter NONE instead of the queue file.

c. If the conversion program cannot find the default queue file, SYS$SYSROOT:[SYSEXE]JBCSYSQUE.DAT, enter the name of your queue file:

```
What is the name of your active QUEUE file:disk:[directory]queuefile.DAT
```

Include the disk and directory when you enter the queue file. If you do not have a queue file, enter NONE.

The conversion program now converts the queue file to the Version 5.0 format. Repeat this step for each system disk in the cluster.

4 Specifying the Default MAIL Database File

The conversion program needs to know the location of your MAIL database file, if you have one. Answer the MAIL database file prompt, as follows:

a. If the default MAIL database file SYS$SYSROOT:[SYSEXE]VMSMAIL.DAT, is the active MAIL database file for your cluster, enter Y:

```
Is SYS$SYSROOT:[SYSEXE]VMSMAIL.DAT your active VMSmail database (Y/N?) Y
```

b. If the default MAIL database file SYS$SYSROOT:[SYSEXE]VMSMAIL.DAT, is not the active file for your cluster, perform these steps to enter the active file for your cluster:

```
Is SYS$SYSROOT:[SYSEXE]VMSMAIL.DAT your active VMSmail database (Y/N?) N
What is the name of your VMSmail database: disk:[directory]maildatabase.DAT
```

Make sure you enter the disk and directory when you enter the MAIL database file. If you do not have a MAIL database (you are not using the Mail Utility), enter NONE.

c. If the conversion program cannot find the default database file, SYS$SYSROOT:[SYSEXE]VMSMAIL.DAT, enter the name of your database file, as follows:

```
What is the name of your VMSmail database:disk:[directory]maildatabase.DAT
```

Include the disk and directory when you enter the database file name. If you do not have a MAIL database file, enter NONE.

The conversion program now converts the MAIL database file to the Version 5.0 format. Repeat this step for each system disk in the cluster.

5 Converting Secondary Queue Files

Skip this step if you do not have secondary queue files.

From the system manager's account, SYSTEM, invoke and run the queue file conversion program, as follows:

```
$ @SYS$UPDATE:JBC$UPGRADE disk:[directory]queuefile
```

Include the disk and directory in the queue file specification.

Repeat this step for each secondary queue file in the cluster.

If the conversion program cannot complete the conversion, error messages are displayed describing the information that cannot be saved or restored, along with the reasons why.

If your terminal hangs during the conversion (program failure due to a corrupted queue file), enter CTRL/C or CTRL/Y to exit the conversion program. The conversion program sets a flag that must be reset using SYSGEN; otherwise, you cannot create a new queue file. To recover from the failure, enter the following commands:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> SET JOBCTLD 0
SYSGEN> WRITE ACTIVE
SYSGEN> EXIT
```

6 Converting Secondary MAIL Database Files

Skip this step if you do not have secondary MAIL database files.

From the system manager's account, SYSTEM, invoke and run the MAIL database file conversion program, as follows:

```
$ @SYS$UPDATE:MAIL$UPGRADE disk:[directory]mailfile
```

Include the disk and directory in the mail file specification.

The program displays a message when conversion is complete. The message tells you the number of MAIL records that were converted.

Repeat this step for each secondary MAIL database file in the cluster.

7 Restarting the Queues

Once all system disks are converted, you should reboot all nodes that share each system disk. If you do not reboot, you must restart all queue managers and queues on all the nodes. For more information on the START/QUEUE/MANAGER and START/QUEUE commands, refer to the *VMS DCL Dictionary*.

8 Running AUTOGEN

DIGITAL recommends running AUTOGEN on all nodes converted to Version 5.0.

You have completed the rolling upgrade.

5 Post-Upgrade Procedures

This chapter describes mandatory and optional procedures to perform once the upgrade is completed.

Complete the following steps after you finish the upgrade.

1 Applying Mandatory Update

If you chose to interrupt the upgrade before applying the mandatory update in Phase 6, you must apply it now. Invoke VMSINSTAL and specify VMSMUP050 as the product. Then reboot the system.

2 Registering Product Authorization Keys (PAKs)

You must register your VMS Product Authorization Keys; otherwise, you can only log in using the operator's console terminal. To register your PAK, refer to Chapter 6.

If you are currently under service, a VMS PAK will automatically be registered as part of the mandatory update. See Section 6.6 for more information.

If you have other PAKs or Service Update PAKs (SUPs), including VAXcluster PAKs, you must register them as well. Refer to Chapter 6.

3 Replacing SYSUAF.DAT

If you permanently store SYSUAF.DAT on the system disk, skip this step and go to step 4.

If you do not permanently store SYSUAF.DAT on the system disk, you copied it to the system disk during the pre-upgrade procedure. The upgrade procedure updates SYSUAF.DAT. Therefore, replace the version you usually use with the updated version by copying the updated SYSUAF.DAT from the system disk to the location you use for your system.

4 Converting Queue and MAIL Files

If you are doing a rolling upgrade, skip this step and go to step 5.

The queue and MAIL database files must be converted to the Version 5.0 format. Complete steps 2 through 6 in Section 4.3.3.

5 Modifying Command Procedures

The upgrade procedure places the latest versions of the following command procedures on your new system disk:

```
[SYSMGR]LTLOAD.COM
[SYSMGR]RTTLOAD.COM
[SYSMGR]STARTNET.COM
[SYSMGR]SYLOGIN.COM
[SYSMGR]SYSTARTUP_V5.COM
[SYSMGR]SYCONFIG.COM
[SYSMGR]SYSHUTDWN.COM
[SYSMGR]SYPAGSWPFILES.COM
[SYSMGR]SYLOGICALS.COM
```

Examine these files; your original versions may have site-specific changes that will be lost if you purge them. Edit the new versions as appropriate to your system.

NOTE: With VMS Version 5.0, the site-specific system startup command procedure is called SYSTARTUP_V5.COM.

If you are upgrading a shared system disk as part of a cluster upgrade, the new files will be the most recent versions in SYS$COMMON; they will not be in the system-specific root. In addition, you must restart all your system queues.

6 Changing MODPARAMS.DAT

If you are upgrading a standalone system, review the file SYS$SYSTEM:MODPARAMS.DAT. The upgrade procedure created a new version of this file. Modify it, if required, for your system. Then go to step 7.

If you are upgrading a system as part of a concurrent or rolling upgrade, follow this procedure.

In all VAXcluster configurations, you must update the MODPARAMS.DAT file for each node that boots from the system disk. QUORUM has been superseded by EXPECTED_VOTES. The Local Area VAXcluster parameters PE3 and PE6 have been renamed NISCS_CONV_BOOT and NISCS_LOAD_PEA0 respectively.

a. Edit MODPARAMS.DAT in each cluster node's [SYSn.SYSEXE] root. Delete the QUORUM parameter.

b. Check to see if the EXPECTED_VOTES parameter is present. If it is not present, add it.

c. Make sure the EXPECTED_VOTES value is correct for your system. The value is the sum of all VOTES in the cluster. For example, if there are five nodes in the cluster, and each has one VOTE, the value is 5.

d. Change the names of PE3 and PE6 to NISCS_CONV_BOOT and NISCS_LOAD_PEA0 respectively.

As you reboot each node, AUTOGEN runs automatically. The cluster forms when enough nodes have been booted to attain cluster quorum.

7 Converting DECnet Proxy Database

Skip this step if your DECnet proxy database is in SYS$SYSTEM (it was converted during Phase 5 of the upgrade). Go to step 8.

Skip this step if you are doing a rolling upgrade. Go to step 8.

If your DECnet proxy database is not in SYS$SYSTEM, you must now convert it to the Version 5.0 format. From the system manager's account, SYSTEM, invoke and run the conversion program as follows:

```
$ RUN SYS$SYSTEM:CVTNAFV5
```

Enter disk and directory information when prompted for the database file specification.

8 Sharing the DECnet Permanent Database

Skip this step if you are upgrading a standalone system and go to step 9.

You can share the components of the DECnet–VAX permanent database among some or all of the nodes in a VAXcluster. The permanent database comprises several separate files. By default, these files are located in SYS$SPECIFIC:[SYSEXE]. For example, after initializing a node, the permanent object database is located in SYS$SPECIFIC:[SYSEXE]NETOBJECT.DAT. If the object database is identical on some nodes in the cluster, the nodes can be configured to share one copy of the file. The following commands establish a shared permanent object database:

a. Copy the permanent object database from one node in the cluster to the shared system disk. For example:

```
$COPY SYS$SPECIFIC:[SYSEXE]NETOBJECT.DAT -
_$SYS$COMMON:[SYSEXE]NETOBJECT.DAT
```

b. For each node in the cluster that should share the database, delete the permanent object database from the private system disk. For example:

```
$ DELETE SYS$SPECIFIC:[SYSEXE]NETOBJECT.DAT;*
```

The remote node database (NETNODE_REMOTE.DAT) is another component of the permanent database that can be shared. To make this component shareable, follow this procedure:

a. On the node with the upgraded-remote-node database, copy the file to the shared system disk. For example:

```
$COPY SYS$SPECIFIC:[SYSEXE]NETNODE_REMOTE.DAT -
_$SYS$COMMON:[SYSEXE]NETNODE_REMOTE.DAT
```

b. On each node in the cluster that will share this file, delete the copy of the database from SYS$SPECIFIC:[SYSEXE]. For example:

```
$ DELETE SYS$SPECIFIC:[SYSEXE]NETNODE_REMOTE.DAT;*
```

c. If you want to move the information from the shared permanent database into your volatile database, enter the following command:

```
$ RUN SYS$SYSTEM:NCP
NCP>  SET KNOWN NODES ALL
```

For additional information on the permanent database, refer to the *VMS Networking Manual*.

9 Updating Console Media for VAX 8530, 8550, 8700 or 8800 Computers

Complete this step only if you have a VAX 8530, 8550, 8700, or 8800 computer.

NOTE: Use the console terminal, DEC Pro CPU, for steps 9a through 9m.

You must copy the Version 5.0 VMB.EXE onto your system's console media.

a. Place a blank RX50 diskette in one of the diskette drive slots on the console (DEC Pro CPU). The console refers to the upper (or left-hand) diskette drive as DZ1 while the VMS operating system refers to it as CSA1. The console refers to the lower (or right-hand) drive as DZ2 while the VMS operating system refers to it as CSA2.

b. Ensure that the diskette drives have been connected to the VMS operating system, as follows:

```
$ SHOW DEVICE CSA
Device                  Device           Error     Volume
 Name                   Status           Count      Label
CSA0:          (SYSTUM) Online               0
CSA1:          (SYSTUM) Online               0
CSA2:          (SYSTUM) Online               0
```

If the devices are not on line, the following message is displayed:

```
%SYSTEM-W-NOSUCHDEV, no such device available
```

If this message is displayed, enter the following commands (CMKRNL privilege is required):

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>  CONNECT CONSOLE
SYSGEN>  CTRL/Z
```

c. Using the VMS operating system, initialize the diskette. Substitute CSA1 or CSA2 for CSAu and substitute a 1- to 12-character name for the volume identification (VOLID) in the following command:

```
$ INITIALIZE/STRUCTURE=1 CSAu: volid
```

d. To mount the diskette that you placed in the console's diskette drive, enter the following command. Substitute CSA1 or CSA2 for CSAu.

```
$ MOUNT/OVERRIDE=ID CSAu:
```

e. To create the required directories on the diskette, enter the following command. Substitute CSA1 or CSA2 for CSAu.

```
$ CREATE/DIRECTORY CSAu:[TRANSFER]
```

f. To copy the new VMB.EXE to the diskette, enter the following command. Substitute CSA1 or CSA2 for CSAu.

```
$ COPY SYS$SYSTEM:VMB.EXE CSAu:[TRANSFER]
```

g. Once you have copied all the files, dismount the diskette. Substitute CSA1 or CSA2 for CSAu in the following command:

```
$ DISMOUNT CSAu:
```

h. To enter console mode and then exit from the console program, enter the following commands:

```
$ CTRL/P
>>> EXIT
```

The dollar sign ($) prompt should be displayed.

i. Open the diskette access door, pause for a moment, and then close the diskette access door. The red indicator light should flash.

j. To copy VMB.EXE from the diskette to the fixed drive, enter the following commands:

```
$ COPY DZu:[TRANSFER]VMB.EXE   LBO:[CONSOLE]
```

Substitute DZ1 or DZ2 for DZu.

The fixed drive in the console is known by two names, DW2 and LB0. Most console files are stored using the name LB0. If the system displays a message that describes a protection violation on the output device, copy the files using the device name DW2.

k. When you have finished copying the files, remove the diskette from the drive.

l. Restart the console program by entering the following command:

```
$ RUN CONTROL
```

m. Return the console to the VMS operating system by entering the following command:

```
>>> SET TERM PROGRAM
```

10 Updating Console Media for a VAX-11/725, 11/730, 11/750, 11/780, 11/785, 8200, 8250, 8300, 8350, 8600, or 8650 Computer

Complete this step only if you have a VAX–11/725, 11/730, 11/750, 11/780, 11/785, 8200, 8250, 8300, 8350, 8600, or 8650 computer.

You must copy the Version 5.0 VMB.EXE onto the console media for all systems. The BOOT58 file for VAX 11/750, 8200, 8250, 8300 and 8350 computers is also updated.

Follow this procedure to check that you have enough space for the new VMB.EXE and to copy the new VMB.EXE onto the console media:

a. Log in to the system manager's account, SYSTEM.

b. Enter the following commands to check if there is enough space for the new VMB.EXE. You need 75 free blocks. If you are updating a VAX–11/725 or VAX–11/730, use CSA2 for CSAu; if you are updating one of the other VAX computers, use CSA1.

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>  CONNECT CONSOLE
SYSGEN>  EXIT
$ EXCHANGE DIRECTORY CSA1:VMB.EXE

Directory of RT-11 volume TIGNES$CSA1:      29-APR-1988 14:04

  VMB.EXE      64   29-Apr-1988

Total of 1 file, 64 blocks.  Free space 23 blocks, largest 16.
```

If the size of the current VMB plus the free space does not equal at least 75 blocks, then delete unneeded files on the console until you have 75 blocks available. In the above example, there are 87 blocks available, so no files have to be deleted.

c. Enter the following command:

```
$ @SYS$UPDATE:UPDATE_CONSOLE.COM
```

d. If you are updating a VAX 8600 or 8650 computer, the new VMB.EXE is copied onto the console.

If you are updating one of the other VAX computers, the procedure uses the EXCHANGE procedure to copy the contents of the existing console, merge the new files on to the copy of the console, and then create a new version of the console media. When it is ready to create the new version, a prompt appears asking you to insert a scratch

medium. Take out the original console medium and insert a new medium.

11 Special File Handling

This step is optional. You do not have to do it to complete the upgrade.

You might want to delete files that you do not need and change file sizes to increase free disk space. Make sure not to delete edited files and shareable images that may be essential to the operating system.

a. You can use the VMSTAILOR program to assist you with deleting files you do not need. VMSTAILOR groups files according to function. Then you can delete all files for an unwanted function without having to search for them. For more instructions about using this program, refer to the installation and operations guide for your VAX computer.

b. The size of the following files may have been changed to fit the system. Check these files to be sure that the sizes are appropriate.

```
[SYSEXE]SYSDUMP.DMP
[SYSEXE]PAGEFILE.SYS
[SYSEXE]SWAPFILE.SYS
```

To modify the size of these files, refer to step 6 in Chapter 2.

c. You might want to purge the following files from the previous version of the VMS operating system:

```
[SYSEXE]SHUTDOWN.COM
[SYSEXE]STARTUP.COM
[SYSLIB]ENCRYPSHR.EXE
[SYSLIB]*RTL*.EXE
[SYSLIB]*SHR.EXE
[SYSMGR]EDTINI.EDT
[SYSMGR]DBLSTRTUP.COM
[SYSMGR]LOGIN.COM
[SYSMGR]TFF$STARTUP.COM
[SYSMGR]WELCOME.TXT
[SYSMGR]*.TEMPLATE
[SYSHLP.EXAMPLES]XADRIVER.MAR
```

d. You might want to purge the following files from the new version of the operating system:

```
[SYSLIB]CDDSHR.EXE
[SYSLIB]UISSHR.EXE
[SYS$STARTUP]VMS$LAYERED.DAT
```

12 Decompressing the System Libraries

This step is optional. You do not have to do it to complete the upgrade.

Many system libraries are shipped in a compressed format. If you have enough disk space on your system, you can decompress the libraries to gain faster access. If you have a time share system that contains a large amount of disk space, it is recommended that you decompress the system libraries. If you do not decompress the libraries, you adversely affect the performance of the HELP and LINK commands.

The decompressed libraries require approximately 7000 additional blocks of disk space. Depending on the type of VAX computer you are using, the decompression process may take up to two hours. Generally, HELP libraries increase in size by 50 percent, while OBJECT libraries increase by 25 percent when decompressed.

To decompress libraries, enter the following command:

```
$ @SYS$UPDATE:LIBDECOMP
```

13 Running UETP

You should run the User Environment Test Package (UETP) to verify that the upgrade was done correctly. For instructions on running UETP, refer to the installation and operations guide for your VAX computer.

14 Building Standalone BACKUP

This step is optional. You do not have to do it to complete the upgrade.

Build the standalone BACKUP kit. For instructions, refer to the installation and operations guide for your VAX computer.

15 Backing Up System Disk

This step is optional. You do not have to do it to complete the upgrade.

Back up the system disk to ensure that you have a fully operational system disk. For instructions, refer to the installation and operations guide for your VAX computer.

16 Backing Up Console Media

This step is optional. You do not have to do it to complete the upgrade.

Back up the console media. From the system manager's account, SYSTEM, invoke the CONSCOPY command procedure, as follows:

```
$ @SYS$UPDATE:CONSCOPY
```

When the procedure asks you what type of operation you want to perform, enter SAVE. Answer all other prompts appropriately for your system.

For complete instructions on using this procedure, refer to the installation and operations guide for your VAX computer.

17 Reinstalling Software Products

Reinstall all optional software products. Refer to Section 1.3, "Layered Products Caution".

18 Installed Images

This step is optional. You do not have to do it to complete the upgrade.

The procedure creates the file SYS$MANAGER:VMSIMAGES.DAT as part of the upgrade. SYS$MANAGER:VMSIMAGES.DAT lists files that the upgrade installed for enhanced system performance. Some files in this list might already be installed in the SYS$MANAGER:SYSTARTUP_V5.COM file; delete the names of these files from the installation section of SYSTARTUP_V5.COM.

19 Restarting System Queues

Restart your system queues. For more information, refer to the START /QUEUE/MANAGER and START/QUEUE commands in the *VMS DCL Dictionary*.

You can also reboot the system to restart the system queues, if you prefer.

If you are doing a concurrent upgrade, return to step 7 in Section 4.1.

If you are doing a rolling upgrade, return to step 7 in Section 4.3.1.

6 License Management

The following sections include information to supplement the *VMS License Management Utility Manual* now provided with VMS Version 5.0. 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.

6.1 Registering Your Licenses

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. With VMS Version 5.0, the license registration procedure replaces those key installations and expands the licensing and enabling concept to many software products.

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

Although this section of the *VMS Version 5.0 Release Notes* explains registering licenses for VMS and the SIPs, many layered products that run on VMS Version 5.0 also require license registration. See the *VMS License Management Utility Manual* for a full explanation of license registration.

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 Figure 6–1. If you do not have a PAK, the following information may not apply to your license registration procedure. For example, if you have a service contract with Digital, a VMS license is registered for you during the VMS installation or upgrade procedure. For details on this kind of VMS license registration, see Section 6.6.

Figure 6–1 Example of a Product Authorization Key (PAK)

```
 - - - - - - -                                        ____________________
| | | | | | | |                                    | DOCUMENT ISSUE DATE |
|d|i|g|i|t|a|l|  PRODUCT AUTHORIZATION KEY (PAK)     |    10-MAY-1988      |
| | | | | | | |                                    |____________________|
 - - - - - - -
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: 5.0
          Product Release Date:

 KEY TERMINATION DATE:
          Key Termination Date: 23-OCT-1988

 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
*************************************************************************
```

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, a process illustrated in Section 6.3.
2. Register the licenses for the System Integrated Products that you purchased. Section 6.5 provides an example of a SIP registration.

Section 6.2 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. Following this are three examples of license registration using VMSLICENSE.COM and fictional samples of PAKs. Do not attempt to register the PAKs in the examples, as they do not work.

You can also register licenses with the LICENSE REGISTER command. See the *VMS License Management Utility Manual* for details about the LICENSE commands, the error messages, and recovery procedures for licensing tasks.

6.2 Registering a License with VMSLICENSE.COM

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:

```
$ @SYS$UPDATE:VMSLICENSE
```

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.  Exit this procedure

Select option:
```

2 Type 1 and press RETURN (DIGITAL is not currently using option 2). The procedure displays the following message:

```
* Do you have your product authorization key?
```

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

3 The procedure asks you to enter a value for each item on the PAK. It starts by displaying some information and asking for the first PAK information:

```
When prompted for input, enter data from corresponding fields on
your Product Authorization Key (PAK) or Product Authorization
Amendment (PAAM).

Some prompts display a default reply (shown in brackets).  To use
the default, press the RETURN key.  To replace default data, enter
the new data to be used.  To cancel the use of default data without
entering new data, enter the backslash (\) character.

You will have the opportunity to review and correct your responses
before actually registering this license. If you wish to exit from
giving responses, do so by typing CTRL/Z.

     PAK ID:

* Issuer [DEC]:
* Authorization Number:
```

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

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]:
```

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

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 (vv.uu):
```

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

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 (dd-mmm-yyyy):
```

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 (dd-mmm-yyyy):
```

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:
```

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.

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. If the PAK does *not* list information for one of the tables, do not enter a value for the item. Press RETURN after the prompt and then go to the next step.

9 The procedure asks for the following information:

```
   MISCELLANEOUS:

* Key Options:
* Product Token:
* Hardware ID:
```

If the PAK does *not* give values for these items, press RETURN after each prompt and go to the next step.

If the PAK gives values for these items, enter the values and press RETURN after each one. 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:

```
* Key Options: MOD_UNITS,NO_SHARE
```

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 the cluster.

* Node this authorization key 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:
* Checksum:
```

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

Note: Currently, the checksum is a 17-character value that always begins with the number 1, which is the only number in the checksum. The other sixteen characters are always letters from A through P.

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

```
              LMF Database:  SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB
                    Issuer:  DEC
      Authorization Number:  USA1956
              Product Name:  VAX-VMS
                  Producer:  DEC
          Number of Units:  400
                   Version:  5.0
     Product Release Date:
     Key Termination Date:  23-OCT-1988
Availability Table Code:  A
    Activity Table Code:
             Key Options:  MOD_UNITS,NO_SHARE
           Product Token:
             Hardware ID:
                Checksum:  1-COOD-AHGO-NEFI-CHIN

This authorization key is restricted to: JUPITR

* Is this information correct?
```

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 the procedure displays the following question, type Y (for YES) and press RETURN:

```
* Do you wish to make corrections?
```

If you choose to start over, the procedure presents the previously entered data as defaults. 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.

12 If you entered all the information correctly, the procedure displays the following:

```
VAX-VMS has been registered.
```

13 After the license is 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 are registering this license for the current node, press RETURN (for YES). If the license loads successfully, the procedure displays an informational message. If you are in a VAXcluster environment and are registering a license for another node, type N (for NO) and press RETURN. You might need to activate the license for the other node after you exit this command procedure. License activation is a process that makes a registered license known to a system. The example in Section 6.3 briefly describes how you register and activate a VMS license in a VAXcluster environment. For details of license activation

in a VAXcluster environment, see the *VMS License Management Utility Manual*.

The procedure displays the following question:

```
* Do you want to register another PAK?
```

If you have more licenses to register, type Y (for YES) and press RETURN. The procedure begins again and asks you for the information on the next PAK.

If you do not have more licenses to register, type N (for NO) and press RETURN. The procedure displays the following:

```
VMS License Management Utility Options:

    1.  Register a Product Authorization Key
    2.  Amend an existing Product Authorization Key
    3.  Exit this procedure

Select option:
```

If you have registered all your licenses, type 3 and press RETURN. The procedure finishes and returns you to the dollar sign prompt ($).

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.

6.3 Registering a VMS License in a VAXcluster Environment

This example is offered primarily to help those registering VMS PAKs in a VAXcluster environment. It also illustrates the procedure for registering a VMS PAK that provides *Availability Table Code* information.

The next example, in Section 6.4, is offered primarily to help those registering a VMS license for a standalone VAX computer. That example also illustrates the procedure for registering a PAK that provides *Activity Table Code* information. If your circumstance does not match these examples exactly, read through the example that matches the PAK table code, but follow the example that matches your system configuration. That is, standalone registration differs from cluster registration more than availability registration differs from activity registration. The license types (designated by availability or activity codes) do not necessarily relate to system configuration.

When you register your VMS PAK, enter the information from your own PAK, not the information provided in the following example. (The example PAK does not work.)

Figure 6–2 Example of a VMS Availability Product Authorization Key (PAK)

```
 - - - - - - -                                        ---------------------
| | | | | | | |                                     | DOCUMENT ISSUE DATE |
|d|i|g|i|t|a|l|  PRODUCT AUTHORIZATION KEY (PAK)     |    10-MAY-1988      |
| | | | | | | |                                     |_____________________|
 - - - - - - -
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: 5.0
          Product Release Date:

 KEY TERMINATION DATE:
          Key Termination Date: 23-OCT-1988

 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
*************************************************************************
```

To register a VMS PAK, 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:

```
$ @SYS$UPDATE:VMSLICENSE
```

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.  Exit this procedure

Select option:
```

2 Type 1 and press RETURN (DIGITAL is not currently using option 2). The procedure displays the following message:

```
* Do you have your Product Authorization Key?
```

For the example you are registering the fictional PAK in Figure 6–2. Type Y and press RETURN.

3 The procedure asks you to enter a value for each item on the PAK. It starts by displaying some information and asking for the first PAK information:

```
When prompted for input, enter data from corresponding fields on
your Product Authorization Key (PAK) or Product Authorization
Amendment (PAAM).

Some prompts display a default reply (shown in brackets).  To use
the default, press the RETURN key.  To replace default data, enter
the new data to be used.  To cancel the use of default data without
entering new data, enter the backslash (\) character.

You will have the opportunity to review and correct your responses
before actually registering this license. If you wish to exit from
giving responses, do so by typing CTRL/Z.

     PAK ID:

* Issuer [DEC]:
* Authorization Number:
```

Press RETURN to specify DEC as the issuer. If the PAK provides another name, enter the name given on the PAK.

Enter the authorization number that appears on the PAK, *USA1956*, and press RETURN.

4 The procedure asks for the following information:

```
   PRODUCT ID:

* Product Name:
* Producer [DEC]:
```

Enter the product name string that appears on the PAK, *VAX-VMS*, and press RETURN.

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, in this case *400*, 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 (vv.uu):
```

If the PAK lists a version number, in this case *5.0*, enter the number. Press RETURN and go to the next step.

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

```
* Product Release Date (dd-mmm-yyyy):
```

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 (dd-mmm-yyyy):
```

If the PAK lists a key termination date, enter the date, in this case *23-OCT-1988*, 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:
```

If the PAK lists an availability table code value or the word CONSTANT followed by an integer, enter the information. For this example enter *A* and press RETURN. If the PAK lists activity table code information, enter that information and press RETURN.

Usually, a PAK lists information for either the availability table code *or* the activity table code. If the PAK does *not* list information for one of the tables, do not enter a value for the item. Press RETURN after the prompt and then go to the next step.

9 The procedure asks for the following information:

```
    MISCELLANEOUS:

* Key Options:
* Product Token:
* Hardware ID:
```

If the PAK gives values for these items, enter the values and press RETURN after each one. For this example, enter the options *MOD_UNITS, NO_SHARE* after the key options prompt and press RETURN. In this example, the PAK uses the NO_SHARE option that signals the command procedure to prompt you for an SCS node name, as follows:

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

* Node this authorization key is restricted to (SCS node name) [MUSIC]: ?
```

If you now type a question mark (?), notice that the procedure provides the following help text:

```
    This Product Authorization Key has been provided with the NO_SHARE
    option.  This requires that this key be restricted to a specific
    node within the cluster.  Enter a node name or just press RETURN
    to use the default name provided.
* Node this authorization key is restricted to (SCS node name) [MUSIC]:
```

In the example, the default SCS node name is MUSIC. An SCS node name is the node name defined by the SYSGEN parameter SCSNODE, If an SCS node name is not defined on your system, the prompt does not display a default SCS node name. Because this example involves registering a license in a VAXcluster environment, you must assign an SCS node name.

To assign a license to the current node, press RETURN. For example, to assign a license to the default SCS node, MUSIC, press RETURN. This would be a common approach if you are registering the first VMS operating system PAK in a VAXcluster environment.

To assign a license to another SCS node name, enter the SCS node name and press RETURN. For example, to assign the license to the SCS node name DRAMA enter *DRAMA* and press RETURN.

Note: You must activate licenses after registration. If you enter a DECnet node name that differs from the SCS node name, you cannot activate the license in a VAXcluster environment. If you are registering a license on a standalone system, you do not need an SCS node name. If the node will become part of a cluster, however, you must assign an SCS node name.

10 Next, the procedure asks you for product token and hardware identification. In this example, the PAK provides no information; press RETURN after each prompt.

11 The procedure asks for the checksum:

```
* Checksum:
```

Enter the checksum, in this case *1-COOD-AHGO-NEFI-CHIN*, and press RETURN.

Note: Currently, the checksum is a 17-character value that always begins with the number 1, which is the only number in the checksum. The other sixteen characters are always letters from A through P.

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

```
                LMF Database:  SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB
                      Issuer:  DEC
       Authorization Number:  USA1956
                Product Name:  VAX-VMS
                    Producer:  DEC
           Number of Units:  400
                     Version:  5.0
      Product Release Date:
      Key Termination Date:  23-OCT-1988
   Availability Table Code:  A
       Activity Table Code:
                Key Options:  MOD_UNITS,NO_SHARE
              Product Token:
                Hardware ID:
                   Checksum:  1-COOD-AHGO-NEFI-CHIN

   This authorization key is restricted to: DRAMA

* Is this information correct?
```

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 the procedure displays the following question, type Y (for YES) and press RETURN:

```
* Do you wish to make corrections?
```

If you choose to start over, the procedure presents the previously entered data as defaults. 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.

13 If you entered all the information correctly, after you type Y and press RETURN the procedure displays the following message:

```
VAX-VMS has been registered.
```

14 After the license is 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 type ? at this point, the procedure displays the following help text:

```
Loading a license makes it known on the current system. Do not load if
the license is restricted to another node. All valid licenses are loaded
at system boot time.

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

In our example, the license is already assigned to node DRAMA rather than the current node MUSIC. Because an attempt to activate it on the current node would fail,[1] type N (for NO) and press RETURN to finish with this license registration. The procedure displays the following prompt:

```
* Do you want to register another PAK?
```

15 If you answer Y and press RETURN, the procedure begins again. You can register another PAK, not only a VMS PAK but any PAK. This example assumes you are finished. Enter N (for NO) and press RETURN. The procedure displays the following:

```
    VMS License Management Utility Options:

        1.  Register a Product Authorization Key
        2.  Amend an existing Product Authorization Key
        3.  Exit this procedure

    Select option:
```

[1] Note that an attempt to activate a license assigned to another node does not produce an error message on the current node.

16 Type 3 and press RETURN to exit the procedure. You have registered the Product Authorization Key for the VMS operating system Version 5.0.

17 After you register a PAK for a node, you must ensure that the license is activated. Unless the VMS license on node DRAMA is activated, you can only log in to the operator's console (OPA0) on that node.

To ensure that the license is activated, use the following procedure: If DRAMA is not running, LMF will activate the license when DRAMA starts up. This occurs only if the node boots from a common system disk that is running VMS Version 5.0 and looks at the common LICENSE database in which the license is registered.

a. If DRAMA is running, log in to the system manager's account, SYSTEM, on that node.

b. Enter a command in the following format and press RETURN:

```
$ LICENSE LOAD /AUTHORIZATION=number VAX-VMS
```

For example:

```
$ LICENSE LOAD /AUTHORIZATION=USA1956  VAX-VMS
```

The VMS operating system is authorized for full use on node DRAMA.

Note that when the LICENSE database contains more than one license for a product, you must identify the correct license by using the /AUTHORIZATION qualifier in your command.

You must register one VMS operating system PAK for each VAX computer in a VAXcluster environment. Each license must be assigned to a node and activated on the assigned node. Although Hierarchical Storage Controllers (HSCs) are considered nodes in a VAXcluster environment, you do not register licenses for them. In a cluster you do not usually need to reinstall the VMS operating system software for each node.

6.4 Registering a VMS License on a Standalone VAX

This example is offered primarily to help those registering a VMS license for a standalone VAX. It also illustrates the procedure for registering a VMS PAK that provides *Activity Table Code* information.

The previous example, in Section 6.3, is offered primarily to help those registering VMS PAKs in a VAXcluster environment. That example also illustrates the procedure for registering a VMS PAK that provides *Availability Table Code* information. Again, if your circumstance does not match these examples exactly, read through the example that matches the PAK table code, but follow the example that matches your system configuration. That is, standalone registration differs from cluster registration more than availability registration differs from activity registration. The license types (availability or activity) do not necessarily relate to system configuration.

Although this example registers an activity PAK for a VAX workstation, you can use it as a guide when you register a PAK on any standalone VAX computer.

6.4 Registering a VMS License on a Standalone VAX

When you register your VMS PAK, enter the information from your own PAK, not the information provided in the following example. (The example PAK does not work.)

Figure 6–3 Example of a VMS Activity Product Authorization Key (PAK)

```
 - - - - - - -                                        ---------------------
| | | | | | | |                                     | DOCUMENT ISSUE DATE |
|d|i|g|i|t|a|l|  PRODUCT AUTHORIZATION KEY (PAK)     |    23-OCT-1988      |
| | | | | | | |                                     |_____________________|
 - - - - - - -
Digital Equipment Corporation
Maynard, Massachusetts

 ----------------------------------
| LICENSE ADMINISTRATION LOCATION:   |  ORDERED BY: Volume Matrix Systems INC.
|                                    |              Mr. S. A. Traybeen
| Digital Equipment Corporation      |              32 Times Blvd.
| Maynard, Massachusetts             |              N. Vera City, Massachusetts
|                                    |              12321
|                                    |
|                                    |
 ----------------------------------
**************************************************************************
 PAK ID:
                      Issuer: DEC
         Authorization Number: USA1776

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

 NUMBER OF UNITS:
              Number of Units: 100

 KEY LEVEL:
                      Version: 5.0
         Product Release Date:

 KEY TERMINATION DATE:
         Key Termination Date: 4-JUL-1989

 RATING:
      Availability Table Code:
          Activity Table Code: D

 MISCELLANEOUS:
                  Key Options: MOD_UNITS,NO_SHARE
                Product Token:
                  Hardware-Id:
                     Checksum: 1-INNA-GADA-DABI-ABAB
**************************************************************************
```

To register a VMS PAK, 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:

```
$ @SYS$UPDATE:VMSLICENSE
```

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.  Exit this procedure

Select option:
```

2 Type 1 and press RETURN (DIGITAL is not currently using option 2). The procedure displays the following message:

```
* Do you have your Product Authorization Key?
```

In this example, you are registering the fictional PAK in Figure 6–3. Type Y and press RETURN.

3 The procedure asks you to enter a value for each item on the PAK. It starts by displaying some information and asking for the first PAK information:

```
When prompted for input, enter data from corresponding fields on
your Product Authorization Key (PAK) or Product Authorization
Amendment (PAAM).

Some prompts display a default reply (shown in brackets).  To use
the default, press the RETURN key.  To replace default data, enter
the new data to be used.  To cancel the use of default data without
entering new data, enter the backslash (\) character.

You will have the opportunity to review and correct your responses
before actually registering this license. If you wish to exit from
giving responses, do so by typing CTRL/Z.

     PAK ID:

* Issuer [DEC]:
* Authorization Number:
```

To specify DEC as the issuer, press RETURN. If the PAK provides another name, enter the name given on the PAK.

Enter the authorization number that appears on the PAK, in this case *USA1776*, and press RETURN.

4 The procedure asks for the following information:

```
   PRODUCT ID:

* Product Name:
* Producer [DEC]:
```

Enter the product name string that appears on the PAK, in this case *VAX-VMS*, and press RETURN.

To specify DEC as the producer, press RETURN.

5 The procedure prompts for the number of units:

```
     NUMBER OF UNITS:

* Number of Units:
```

If the PAK lists the number of units, enter the number of units, in this case *100*, 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 (vv.uu):
```

If the PAK lists a version number, in this case *5.0*, enter the number. Press RETURN and go to the next step.

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

```
* Product Release Date (dd-mmm-yyyy):
```

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 (dd-mmm-yyyy):
```

If the PAK lists a key termination date, enter the date, in this case *4-JUL-1989*, and press RETURN.

If the PAK does *not* list a license 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:
```

If the PAK lists an availability table code value or the word CONSTANT followed by an integer, enter the information.

Usually, a PAK lists information for either the availability table code *or* the activity table code. If the PAK does *not* list information for one of the tables, do not enter a value for the item. Press RETURN after the prompt and then go to the next step.

For this example, press RETURN after the availability table code prompt. The PAK for this example provides activity table code information. In this case, enter *D* at the activity table code prompt and press RETURN.

9 The procedure asks for the following information:

```
        MISCELLANEOUS:

* Key Options:
* Product Token:
* Hardware ID:
```

If the PAK gives values for these items, enter the values and press RETURN after each one. For this example, enter the options *MOD_UNITS, NO_SHARE* after the key options prompt and press RETURN. In this example, the PAK uses the NO_SHARE option that signals the command procedure to prompt you for an SCS node name, as follows:

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

* Node this authorization key is restricted to (SCS node name) [GRAPH]: ?
```

If you now type a question mark (?), notice that the procedure provides the following help text:

```
    This Product Authorization Key has been provided with the NO_SHARE
    option.  This requires that this key be restricted to a specific
    node within the cluster.  Enter a node name or just press RETURN
    to use the default name provided.
* Node this authorization key is restricted to (SCS node name) [GRAPH]:
```

In the example, the default SCS node name is GRAPH. An SCS node name is the node name defined by the SYSGEN parameter SCSNODE, If an SCS node name is not defined on your system, the prompt does not display a default SCS node name. Because this example involves registering a license on a standalone system, you do not need to assign an SCS node name. If the node will become part of a cluster, however, you must assign an SCS node name.

To assign the license to the default SCS node name, press RETURN.

Note: If you enter a DECnet node name instead of an SCS node name and the node later joins a VAXcluster environment, you will not be able to activate the license.

10 Next, the procedure asks you for product token and hardware identification. In this example, the PAK provides no information; press RETURN after each prompt.

11 The procedure asks for the checksum:

```
* Checksum:
```

Enter the checksum, in this case *1-INNA-GADA-DABI-ABAB*, and press RETURN.

Note: Currently, the checksum is a 17-character value that always begins with the number 1, which is the only number in the checksum. The other sixteen characters are always letters from A through P.

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

```
            LMF Database:  SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB
                  Issuer:  DEC
    Authorization Number:  USA1776
            Product Name:  VAX-VMS
                Producer:  DEC
         Number of Units:  100
                 Version:  5.0
    Product Release Date:
    Key Termination Date:  4-JUL-1989
Availability Table Code:  A
    Activity Table Code:
             Key Options:  MOD_UNITS,NO_SHARE
           Product Token:
             Hardware ID:
                Checksum:  1-INNA-GADA-DABI-ABAB

   This authorization key is restricted to: GRAPH

* Is this information correct?
```

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 the procedure displays the following question, type Y (for YES) and press RETURN:

```
* Do you wish to make corrections?
```

If you choose to start over, the procedure presents the previously entered data as defaults. 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.

13 If you entered all the information correctly, after you type Y and press RETURN the procedure displays the following informational message:

```
VAX-VMS has been registered.
```

14 After the license is 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]?
```

In this example, which uses a standalone VAX system, type Y and press RETURN.

If you type Y and the license is activated, the procedure displays the following informational message and prompt:

```
%LICENSE-I-LOADED, DEC VAX-VMS was successfully loaded with 100 units

  * Do you want to register another PAK?
```

15 If you answer Y and press RETURN, the procedure begins again. You can register another PAK, not only a VMS PAK but any PAK. This example assumes you are finished. Enter N (for NO) and press RETURN. The procedure displays the following:

```
VMS License Management Utility Options:

    1.  Register a Product Authorization Key
    2.  Amend an existing Product Authorization Key
    3.  Exit this procedure

Select option:
```

16 To exit the procedure, type 3 and press RETURN. You have registered the Product Authorization Key for the VMS operating system Version 5.0. If an error message was displayed when the procedure attempted to load the license, the license activation, not the license registration, was affected. Exit the procedure and read the sections of the *VMS License Management Utility Manual* that describe activating a license. You must activate a registered license to make it known to a system.

6.5 Registering a System Integrated Product License

The following example provides a step-by-step procedure to register a PAK for VAX Volume Shadowing, a System Integrated Product. This example uses the command procedure VMSLICENSE.COM. and the following sample of a PAK.

When you register your System Integrated Product PAK, enter the information from your own PAK, not the information provided in the following example. (The example PAK does not work.)

Figure 6–4 Example of a VMS Volume Shadowing Product Authorization Key (PAK)

```
 - - - - - - -                                          ____________________
| | | | | | | |                                       | DOCUMENT ISSUE DATE |
|d|i|g|i|t|a|l|  PRODUCT AUTHORIZATION KEY (PAK)       |    10-MAY-1988      |
| | | | | | | |                                       |_____________________|
 - - - - - - -
Digital Equipment Corporation
Maynard, Massachusetts

 ------------------------------------
| LICENSE ADMINISTRATION LOCATION:   |  ORDERED BY: DATATWICE INC.
|                                    |              Mr. C. T. Friendly
| Digital Equipment Corporation      |              2 Times Blvd.
| Maynard, Massachusetts             |              Twin City, Connecticut
|                                    |              12321
|                                    |
|                                    |
 ------------------------------------
*************************************************************************
 PAK ID:
                       Issuer: DEC
         Authorization Number: USA126087

 PRODUCT ID:
                 Product Name: VOLSHAD
                     Producer: DEC

 NUMBER OF UNITS:
              Number of Units: 460

 KEY LEVEL:
                      Version: 5.0
         Product Release Date:

 KEY TERMINATION DATE:
         Key Termination Date: 31-DEC-1988

 RATING:
      Availability Table Code: E
          Activity Table Code:

 MISCELLANEOUS:
                  Key Options: MOD_UNITS
                Product Token:
                  Hardware-Id:
                     Checksum: 1-ADEB-DOCJ-NENC-KDBM
*************************************************************************
```

6.5 Registering a System Integrated Product License

To register a VAX Volume Shadowing PAK, 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:

```
$ @SYS$UPDATE:VMSLICENSE
```

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.  Exit this procedure

Select option:
```

2 Type 1 and press RETURN (DIGITAL is not currently using option 2). The procedure displays the following message:

```
* Do you have your Product Authorization Key?
```

For the example, you are registering the fictional PAK in Figure 6–4. Type Y and press RETURN.

3 The procedure asks you to enter a value for each item on the PAK. It starts by displaying some information and asking for the first PAK information:

```
When prompted for input, enter data from corresponding fields on
your Product Authorization Key (PAK) or Product Authorization
Amendment (PAAM).

Some prompts display a default reply (shown in brackets).  To use
the default, press the RETURN key.  To replace default data, enter
the new data to be used.  To cancel the use of default data without
entering new data, enter the backslash (\) character.

You will have the opportunity to review and correct your responses
before actually registering this license. If you wish to exit from
giving responses, do so by typing CTRL/Z.

     PAK ID:

* Issuer [DEC]:
* Authorization Number:
```

To specify DEC as the issuer, press RETURN.

Enter the authorization number that appears on the PAK, in this case *USA126087*, and press RETURN.

4 The procedure asks for the following information:

```
   PRODUCT ID:

* Product Name:
* Producer [DEC]:
```

Enter the product name string that appears on the PAK, in this case *VOLSHAD*, and press RETURN.

To specify DEC as the producer, press RETURN.

5 The procedure prompts for the number of units:

```
     NUMBER OF UNITS:

* Number of Units:
```

If the PAK lists the number of units, in this case *460*, 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 (vv.uu):
```

If the PAK lists a version number, in this case *5.0*, enter the number. Press RETURN and go to the next step.

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

```
* Product Release Date (dd-mmm-yyyy):
```

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

7 The procedure prompts for the key termination date:

```
    KEY TERMINATION DATE:

* Key Termination Date (dd-mmm-yyyy):
```

If the PAK lists a key termination date, enter the date, in this case *31-DEC-1988*, and press RETURN.

If the PAK does *not* list a license 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:
```

If the PAK lists an availability table code value or the word CONSTANT followed by an integer, enter the information. For this example enter *E* and press RETURN. If the PAK lists activity table code information, enter that information and press RETURN.

If the PAK does *not* list information for one of the tables, do not enter a value for the item. Press RETURN after the prompt and then go to the next step. For this example, press RETURN after the activity table code prompt.

9 The procedure asks for the following information:

```
    MISCELLANEOUS:

* Key Options:
* Product Token:
* Hardware ID:
```

If the PAK gives values for these items, enter the values and press RETURN after each one. For this example, enter the option *MOD_UNITS* after the key options prompt and press RETURN. Press RETURN after each other prompt.

10 The procedure asks for the checksum:

```
* Checksum:
```

Enter the checksum, in this case *1-ADEB-DOCJ-NENC-KDBM*, and press RETURN.

Note: Currently, the checksum is a 17-character value that always begins with the number 1, which is the only number in the checksum. The other sixteen characters are always letters from A through P.

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

```
                LMF Database:  SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB
                      Issuer:  DEC
       Authorization Number:  USA126087
                Product Name:  VOLSHAD
                    Producer:  DEC
            Number of Units:  460
                     Version:  5.0
       Product Release Date:
       Key Termination Date:  31-DEC-1988
   Availability Table Code:  E
       Activity Table Code:
                Key Options:  MOD_UNITS
              Product Token:
                Hardware ID:
                   Checksum:  1-ADEB-DOCJ-NENC-KDBM

* Is this information correct?
```

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

If the information 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 the procedure displays the following question, type Y (for YES) and press RETURN:

```
* Do you wish to make corrections?
```

If you choose to start over, the procedure presents the previously entered data as defaults. 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.

12 If you entered all the information correctly, after you type Y and press RETURN the procedure displays the following informational message:

```
VOLSHAD has been registered.
```

13 After the license is 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 type a question mark (?) at this point, the procedure displays the following help text:

```
Loading a license makes it known on the current system. Do not load if
the license is restricted to another node. All valid licenses are loaded
at system boot time.

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

If you register the PAK on a standalone system and want to make the software available (active) immediately, type Y and press RETURN. If you register the license in a VAXcluster environment but do not want to make the software available immediately on the current node, type N and press RETURN. This example assumes that you are registering the license in a VAXcluster environment; however you do not want it activated on the current node. Type N (for NO) and press RETURN. The procedure displays the following prompt:

```
* Do you want to register another PAK?
```

14 If you answer Y and press RETURN, the procedure begins again. You can register another PAK. This example assumes you are finished. Enter N (for NO) and press RETURN. The procedure displays the following:

```
VMS License Management Utility Options:

    1.  Register a Product Authorization Key
    2.  Amend an existing Product Authorization Key
    3.  Exit this procedure

Select option:
```

15 To exit the procedure, type 3 and press RETURN. You have registered the Product Authorization Key for Volume Shadowing.

16 You must activate the VAX Volume Shadowing license before you can use the product.

To ensure that the license is activated, use the following procedure: If the node that is to use volume shadowing is not running, LMF can activate the license when that node starts up. This occurs only if the node boots from a common system disk that is running VMS Version 5.0 and looks at the common LICENSE database in which the license is registered.

a. If the node intended to use volume shadowing is running, log in to the system manager's account, SYSTEM, on that node.

b. Enter a command in the following format and press RETURN:

```
$ LICENSE LOAD /AUTHORIZATION=number VOLSHAD
```

For example:

```
$ LICENSE LOAD /AUTHORIZATION=USA126087 VOLSHAD
```

VAX Volume Shadowing is authorized for full use on the current node.

17 If you then want to control which nodes in the VAXcluster environment have access to the volume shadowing license and software, enter a command in the following format and press RETURN:

```
$ LICENSE MODIFY /INCLUDE=(node-name[,node-name]) VOLSHAD
```

For a description of this command, see the *VMS License Management Utility Manual*.

6.6 Managing VMS W-KIT Licenses for Service Customers

If you have registered your VMS PAKs using VMSLICENSE or the LICENSE REGISTER command, the information in this section does not apply to you. If, however, you are a VMS Service Customer, this section explains the VMS license registration process and provides information for license management.

If you install VMS with a W-KIT, your VMS Service Update PAK (SUP) is generated when you apply the mandatory update (MUP) software after performing the VMS installation or upgrade procedure. The command procedure SYS$UPDATE:LMF$CONFIG.COM creates a system specific LICENSE database called SYS$SPECIFIC:[SYSEXE]LMF$SYSTEM.LDB. LMF$CONFIG then determines the appropriate VMS SUP for your system and registers it in this database.

If you are running 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.

Because all VMS SUPs must be associated with a particular VAX computer, LMF$CONFIG attempts to determine the computer's SCS node name. An SCS node name is the node name 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. License activation is a process that makes a license known to the current computer by loading information into the computer's memory.

When LMF$CONFIG successfully registers the VMS SUP for you, a confirmation message is displayed. This license is then activated automatically each time the system starts up. Although LMF usually displays a confirmation message when a registered license is activated, no confirmation is displayed when these VMS licenses are activated.

If you are upgrading a cluster, LMF$CONFIG.COM will be run on the other members of the cluster as they re-boot following the upgrade. If you add a new member to the cluster, LMF$CONFIG.COM will be run as part of the CLUSTER_CONFIG ADD option.

You should not register any other PAKs in the system specific database created by LMF$CONFIG.COM. Each VAX computer in a cluster can have one of these databases, but each should contain only one VMS SUP. Do not delete or rename the databases.

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

```
$  LICENSE MODIFY VAX-VMS /INCLUDE=new_node_name -
_$ /DATABASE = SYS$SPECIFIC:[SYSEXE]LMF$SYSTEM.LDB
```

A VAXcluster can operate with VMS licenses in both the system specific license databases and the default common LICENSE database. The LMF$CONFIG procedure is designed only to help your transition to VMS Version 5.0. If you receive VMS PAKs for new VAX computers, you must register these in the default common database (SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB), using the

License Management Utility (LICENSE) or the command procedure VMSLICENSE.COM. Be sure to assign any new PAKs to the proper SCS node name using VMSLICENSE or the LICENSE MODIFY command. LMF activates all new VMS licenses from the common database as well as those licenses generated by LMF$CONFIG.

6.7 Modifying License Units with the License Management Utility

The following information is provided as a supplement to other License Management Facility documentation. Before you use the following 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 5.0, 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 zero 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:

```
$ LICENSE LIST /FULL product-name
```

For example:

```
$ LICENSE LIST /FULL FORTRAN

Press CTRL/Z to exit, use arrow keys to scroll.

 License Management Facility

 License Database File:       ART::SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB
 Created on:                  17-AUG-1988
 Created by user:             MONET
 LMF Version:                 V1.0

 ----------------------------------------------------
 Issuer:                      DEC
 Authorization:               USA-2468
 Product Name:                FORTRAN
 Producer:                    DEC
 Units:                       900
 Modified Units:              9999
 Version:                     5.0
 Date:                        (none)
 Termination Date:            21-DEC-1990
 Availability:                F (Layered Products)
 Activity:                    0
 Options:                     MOD_UNITS
 Hardware ID:
```

```
 Revision Level:             1
 Status:                     History
 Command:                    REGISTER
 Modified by user:           DEGAS
 Modified on:                29-AUG-1988 12:12:27.33
[End of List]
```

The 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 and examine the display. Look for a code A, B, C, D, E, or F next to either the *Availability Table Code* label or the *Activity Table Code* label in the LICENSE LIST display. Check the codes, which designate license type, to determine an appropriate license size for your VAX computer.
2. Find the name of your VAX computer in the first column of the VMS Version 5.0 License Unit Requirement Tables (Table 6–1).
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, you might find VAX 8800, column F. 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 determined in the previous step 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:

   ```
   $ LICENSE MODIFY/UNITS=1200  FORTRAN
   ```

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:

```
$ LICENSE UNLOAD FORTRAN
$ LICENSE LOAD FORTRAN
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 these License Unit Requirement Tables (LURT) are subject to change.

Table 6–1 License Unit Requirement Tables (LURT)

	License Types by Code					
	VMS				SIP	LP
System Marketing Model	A	B	C	D	E	F
VAX 11/725	10	NA	NA	NA	230	100
VAX 11/730	10	NA	NA	NA	230	100
VAX 11/750	12	NA	NA	NA	230	100
VAX 11/780	13	NA	NA	NA	230	100
VAX 11/785	13	NA	NA	NA	230	100
VAX 6210	58	NA	NA	NA	230	300
VAX 6220	69	NA	NA	NA	230	600
VAX 6230	81	NA	NA	NA	400	900
VAX 6240	93	NA	NA	NA	400	1200
VAX 8200	20	NA	NA	NA	230	100
VAX 8250	20	NA	NA	NA	230	100
VAX 8300	25	NA	NA	NA	230	200
VAX 8350	25	NA	NA	NA	230	200
VAX 8530	65	NA	NA	NA	230	400
VAX 8550	72	NA	NA	NA	400	600
VAX 8600	28	NA	NA	NA	230	400
VAX 8650	28	NA	NA	NA	230	400
VAX 8700	72	NA	NA	NA	400	600
VAX 8800	93	NA	NA	NA	400	**1200**
VAX 8810	72	NA	NA	NA	400	600
VAX 8820	93	NA	NA	NA	400	1200
VAX 8830	119	NA	NA	NA	600	1800
VAX 8840	143	NA	NA	NA	600	2400
MicroVAX I	1	NA	NA	NA	50	10
VAXstation I	1	NA	NA	NA	50	10
MicroVAX II	18	NA	100	NA	230	50
VAXstation II	NA	NA	NA	100	50	10
VAXstation II/GPX	NA	NA	NA	100	50	10
MicroVAX 2000	18	NA	100	NA	230	20

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

Table 6–1 (Cont.) License Unit Requirement Tables (LURT)

	License Types by Code					
	VMS				SIP	LP
System Marketing Model	A	B	C	D	E	F
VAXstation 2000	NA	NA	NA	100	50	10
MicroVAX 3200	60	NA	100	NA	230	300
MicroVAX 3500	60	NA	100	NA	230	300
VAXstation 3500	NA	NA	NA	100	50	10
VAXserver 3500	NA	100	NA	100	50	10
MicroVAX 3600	60	NA	100	NA	230	300
VAXstation 3600	NA	NA	NA	100	50	10
VAXstation 3600/GPX	NA	NA	NA	100	50	10
VAXserver 3600	NA	100	NA	100	50	10
VAXstation 8000	NA	NA	NA	100	50	10

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

6.8 License Management Facility (LMF) Notes

The following list is offered to help new users with some common concerns and questions regarding the new 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 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 SYLOGICALS.COM:

  ```
  $ DEFINE/SYSTEM LMF$LICENSE device:[directory]LMF$LICENSE.LDB
  ```

  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, you may have VMS licenses registered in a separate LICENSE database located in SYS$SPECIFIC:[SYSEXE]LMF$SYSTEM.LDB. Each node of a VAXcluster environment might 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, however, unless you change the SCS node name of the VAX computer. For details see Section 6.6.

- 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 disks that boot from a separate system disk, you must redirect LMF to the LICENSE database. Define 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, make them identical. Whenever one database is modified, you must copy it to update the other database.

- 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 that an SCS node name can be different from a DECnet node name. SCSNODE is a System Generation Utility (SYSGEN) parameter.

6.9 VMS License Types

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 to be licensed.

- 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.

6.10 License Activity Use Definition for VMS Licenses

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.

6.11 VAXcluster License Notes

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
```

In VMS Version 5.0, all normal VAXcluster environment processing continues on the nodes, however. You should register your VAXcluster license as soon as possible.

6.12 DECnet–VAX License Notes

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 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*.

6.13 VAX RMS Journaling License Notes

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

6.14 VAX Volume Shadowing License Notes

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.

6.15 New DCL LICENSE Command

Invokes the License Management Utility (LICENSE), used to manage software licenses on the VMS operating system. For a complete description of this utility, see the *VMS License Management Utility Manual*, part of the VMS Base Documentation Set.

LICENSE *subcommand parameter*

6.16 New DCL SHOW LICENSE Command

Displays all the software product licenses active on the current node. An active license is one that has been registered in the LICENSE database *and* loaded into system memory. To register and activate software product licenses, use the License Management Utility (LICENSE), or VMSLICENSE.COM. Some licenses are registered automatically during product installation.

For a complete description of the License Management Utility, see the *VMS License Management Utility Manual.*

To display licenses registered in the LICENSE database, use the LICENSE LIST command, described with the utility.

SHOW LICENSE

Qualifier

/OUTPUT[=filespec]
/NOOUTPUT

By default, the output of the SHOW LICENSE command is sent to the current SYS$OUTPUT device (usually your terminal). To send the output to a file, use the /OUTPUT qualifier followed by a file specification.

You cannot use any wildcard characters for the file specification. If you enter a partial file specification (for example, specifying only a directory), SHOW is the default file name and LIS is the default file type.

If you enter the /NOOUTPUT qualifier, output is suppressed.

```
$ SHOW LICENSE

    Active licenses on node WTPOOH:

        DVNETEND
                Producer: DEC
                Units: 0
                Version:  5.0
                Date:  (none)
                Termination Date: (none)
                Availability: E (System Integrated Products)
                Activity:  0
                MOD_UNITS

        VAX-VMS
                Producer: DEC
                Units: 0
                Version:  5.0
                Date:  (none)
                Termination Date: (none)
                Availability: A (VMS Capacity)
                Activity:  0
                MOD_UNITS
                NO_SHARE
```

The SHOW LICENSE command in this example displays all the active licenses on the current node named WTPOOH.

```
$ SHOW LICENSE/OUTPUT=SYS$LOGIN:ACTIVE_LICENSES_OCT30.DAT
$
```

The SHOW LICENSE command in this example writes all the active licenses to the file named SYS$LOGIN:ACTIVE_LICENSES_OCT30.DAT.

6.17 LICENSE LIST Command Problem

Usually the LICENSE LIST command displays the last command entered for a particular license record. Currently, however, when you enter LICENSE MODIFY commands with the /INCLUDE or /EXCLUDE qualifiers, LMF does not correctly update the *Command* field of the license record. Thus, the display of the LICENSE LIST command does not reflect the most recent command entered. The previous command is displayed. This does not affect the performance of LICENSE MODIFY commands.

7 General User Release Notes

This chapter discusses information about the VMS Version 5.0 operating system that is of interest to the general user.

For information about the new features included in VMS Version 5.0, see the *VMS Version 5.0 New Features Manual.*

7.1 Digital Command Language (DCL) — Notes

The following sections describe elements of the Digital Command Language (DCL) that have been changed for VMS Version 5.0. It includes restrictions and problems that are present in DCL for Version 5.0.

7.1.1 DCL Command DEFINE/FORM Qualifier /SHEET_FEED — Restriction

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:

```
$ STOP/QUEUE/RESET queue-name
$ START/QUEUE queue-name
```

7.1.2 DCL Command Statement IF Level Clarification

The *VMS Version 5.0 New Features Manual* states that the DCL command procedure statement IF can be nested up to 16 levels. While this is true, you should note that the first level includes statements prior to the first IF statement. Subsequently, only 15 IF command statements may be nested.

7.1.3 DCL Command OPEN — Problem

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:

```
$ OPEN/SHARE FOO FOO.TMP

$ OPEN/NOSHARE FOO FOO.TMP
```

Consequently, 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.

7.1.4 DCL Command RENAME Requires Delete Access

The DCL command RENAME now requires that you have delete access to any file you rename. See Section 8.71.1 for more information about file protection changes in VMS Version 5.0.

7.1.5 DCL Command SET PROCESS/CPU=[NO]ATTACHED Removed

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). See Section 8.2 for more information.

7.1.6 Using the DCL Command SET TIME

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.

7.1.7 Using the DCL Command SET TIME/CLUSTER

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.

Note: Beginning with VMS Version 5.0, SET TIME/CLUSTER has been superseded by the System Management Utility (SYSMAN) command CONFIGURATION. See Section 8.37 for more information.

7.1.8 DCL Command SHOW QUEUE/SUMMARY Displays Incorrect Job Counts

The pending, holding, and retained job counts for a queue displayed with the DCL command SHOW QUEUE/SUMMARY or obtained from the $GETQUI system service, the F$GETQUI lexical function, and the LIB$GETQUI run-time library routine may be greater than the actual number of jobs in each of these states. Operations such as requeuing a job and deleting a job can result in the job controller failing to decrement the counts in the queue file properly. DIGITAL will correct this problem in a future release of the VMS operating system.

7.1.9 Obsolete Commands

Table 7–1 lists DCL commands that are obsolete for VMS Version 5.0. Table 7–1 also contains equivalent commands that you can use in place of these obsolete commands. Refer to the *VMS DCL Dictionary* and *VMS Obsolete Features Manual* for further information.

Table 7–1 Obsolete Commands

Obsolete Command	Equivalent Command
SET DEVICE/ACL	SET ACL/OBJECT_TYPE=DEVICE
SET DIRECTORY/ACL	SET ACL/OBJECT_TYPE=FILE
SET FILE/ACL	SET ACL/OBJECT_TYPE=FILE
SET QUEUE/ENTRY	SET ENTRY entry number

Note: The SET ACL command allows you to modify the access control list (ACL) of the object specified by the /OBJECT_TYPE qualifier. The default object type is FILE, so the /OBJECT_TYPE does not need to be specified when modifying the ACL of a file or directory.

7.2 EDT — Problem

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).

7.3 EVE — Notes

The following sections provide information about the Extensible VAX Editor (EVE).

7.3.1 Incompatibilities with Previous Versions

This version of EVE contains the following incompatibilities with previous versions of EVE:

1 The procedure EVE$SEARCH requires a new fifth parameter. This new parameter fixes a previous problem in the way EVE matched characters while executing the FIND and REPLACE commands.

The new parameter is of type integer and controls whether EVE$SEARCH matches the character at the current cursor position. A false (even) integer causes EVE$SEARCH to disregard the character at the current cursor position. A true (odd) integer causes EVE$SEARCH to match the character at the current cursor position.

2 In previous versions of VAXTPU, during a series of MOVE_VERTICAL operations, the cursor always processed as if COLUMN_MOVE_VERTICAL was not operative. That is, a tab was always treated as a single character, resulting in instances where the cursor switched columns as it moved vertically. (For more information on cursor behavior with COLUMN_MOVE_VERTICAL turned on and turned off, see the description of SET (COLUMN_MOVE_VERTICAL) in Chapter 4 of the *VAX Text Processing Utility Manual*.

This version of EVE sets the COLUMN_MOVE_VERTICAL attribute to ON. Therefore, if you have an application that was layered on a previous version of EVE and it depended on the previous style of vertical cursor movement, DIGITAL recommends that you modify your application implementing the following steps:

- Fetch the status of COLUMN_MOVE_VERTICAL using GET_INFO (SYSTEM, "COLUMN_MOVE_VERTICAL")
- Turn the COLUMN_MOVE_VERTICAL attribute off
- Restore the previous COLUMN_MOVE_VERTICAL setting after your application has finished running

If you are writing new programs based on the new version of VAXTPU, see the description of SET (COLUMN_MOVE_VERTICAL) in Chapter 4 of the *VAX Text Processing Utility Manual* for information about using the MOVE_VERTICAL built-in procedure.

Note that previous versions of VAXTPU handled margins differently than this version. The incompatibilities are described in the next release note.

The fact that both vertical cursor movement and margin handling have changed complicates operations involving both vertical cursor movement and reference to margins. For example, a rectangular cut and paste application layered on a previous version of EVE requires careful modification for this release of the EVE editor. If you have an application layered on a previous version of EVE, which involves both vertical cursor movement and reference to margin settings, DIGITAL recommends that you read the release note on handling of margins before deciding how to modify your code.

3 This version of VAXTPU handles margin settings and text filling differently than previous versions of VAXTPU. As a result, some applications that were layered on previous versions of VAXTPU or EVE no longer correctly handle some formatting operations (such as rectangular cut and paste). The following information provides

background on the incompatibility and presents code enabling applications based on previous versions to run on the new versions of EVE and VAXTPU.

Background on Incompatibility

In previous versions of VAXTPU, each record started in Column 1. If an application supported setting of margins by the user, the application created the left margin by inserting padding blanks between Column 1 and the first column where text was to appear. When text was filled between margin settings, the left margin was handled in the same way.

In this version of VAXTPU, the SET (MARGINS) and SET (LEFT_MARGIN) built-ins create "hard" left margins. That is, these built-ins cause records to start in the column specified as the left margin. The FILL built-in also sets "hard" left margins after filling the specified range or string.

The new "hard" left margins interfere with applications layered on previous versions that either step from the beginning of one line to the beginning of the next, or that depend on records starting in Column 1.

Strategies to Deal with Version Incompatibilities

This note presents two possible strategies for addressing the incompatibility. The first strategy allows code layered on previous versions to take advantage of new left-margin-related features of the new version of VAXTPU. This strategy involves adding the procedure "user_strip_margin" to all operations (such as rectangular cut and paste) that depend on records starting in Column 1. (The code for this procedure is presented below.) The USER_STRIP_MARGIN procedure removes the "hard" left margins in a specified range and sets left margins in the range to Column 1. This procedure replaces the "hard" left margins with left margins created by inserting padding blanks between Column 1 and the column where text is to begin. DIGITAL recommends that you use this approach to modify your code.

The second strategy does not enable code layered on previous versions to take advantage of new left-margin-related features of the new version of VAXTPU, but does enable code layered on previous versions to work properly when layered on the newest version. By using this approach, you substitute a call to the procedure USER_SET_MARGINS (shown in the example that follows) for the SET (MARGINS) built-in.

The USER_SET_MARGINS procedure creates the arrays USER_LEFT_MARGIN and USER_RIGHT_MARGIN in which margin values specified by the user are stored. This prevents VAXTPU from setting "hard" margins. In this approach, the VAXTPU left margin is always set to Column 1. Record offsets correspond to screen columns, as they did in previous versions of VAXTPU.

If you use this method, you must substitute the value USER_LEFT_MARGIN (buffer) for GET_INFO (buffer, "LEFT_MARGIN") and USER_RIGHT_MARGIN (buffer) for GET_INFO (buffer, "RIGHT_MARGIN").

To implement this approach, make the following substitutions in your code:

Instead of	Use
SET (MARGINS, buffer, n, m)	USER_SET_MARGINS (buffer, n, m)
GET_INFO (buffer, "LEFT_MARGIN")	USER_LEFT_MARGIN (buffer)
GET_INFO (buffer, "RIGHT_MARGIN")	USER_RIGHT_MARGIN (buffer)

Note that this approach still requires that you call the USER_STRIP_MARGIN procedure after every operation that uses the FILL built-in. This is because in the newest version of VAXTPU, FILL creates "hard" margins that must be replaced. You must pass to the USER_STRIP_MARGIN procedure an argument specifying the range that was filled. Thus, this approach also requires the following substitution:

Instead of	Use
FILL (range, string)	FILL (range, string, USER_LEFT_MARGIN (CURRENT_BUFFER), USER_RIGHT_MARGIN (CURRENT_BUFFER)); USER_STRIP_MARGIN (range);

Note that previous versions of EVE handled vertical cursor movement differently. The fact that both vertical cursor movement and margin handling have changed complicates operations involving both vertical cursor movement and reference to margins. For example, a rectangular cut and paste application layered on a previous version of EVE requires careful modification for this version of EVE. If you have an application layered on a previous version of EVE that involves both vertical cursor movement and reference to margin settings, DIGITAL recommends that you read the release note on vertical cursor movement before deciding how to modify your code.

Code to Implement the First Strategy

The procedure USER_STRIP_MARGIN is as follows:

```
! The following procedure sets at Column 1 the left margin
! of all the lines in a specified range. Padding
! spaces are inserted to preserve the screen appearance
! of the text.  This procedure may be required
! to enable such operations as rectangular cut
! and paste to operate correctly when layered on
! VAXTPU 2.0.
!
! The procedure takes one parameter, of type range.
! The parameter specifies the text that is to be modified.
! The range must start at the beginning of a line.
!
! Note that the procedure uses POSITION (LINE_BEGIN)
! after each MOVE_VERTICAL (1) to ensure that the editing
! point is at the beginning of the line.
!
procedure user_strip_margin (the_range)

local   the_start,
        the_end,
        here,
        the_offset,
        saved_left_margin,
        saved_mark,
        saved_mode;

 on_error
    [OTHERWISE]:
         if saved_mode = OVERSTRIKE
         then
             set (OVERSTRIKE, current_buffer);
         endif;
         if saved_mark <> tpu$k_unspecified
         then
             position (saved_mark);
         endif;
endon_error;

saved_mark := mark (FREE_CURSOR);
start_mark := beginning_of (the_range);
the_end := end_of (the_range);

position (start_mark);
saved_left_margin := get_info (current_buffer, "left_margin");
if saved_left_margin > 1
then
    set (LEFT_MARGIN, current_buffer, 1);
endif;

saved_mode := get_info (current_buffer, "mode");
set (INSERT, current_buffer);
```

```
loop
    position (LINE_BEGIN);
    here := mark (NONE);
    exitif not get_info (here, "within_range", the_range);
    the_offset := get_info (here, "left_margin");
    if the_offset > 1
    then
        if here <> beginning_of (current_buffer)
        then
            append_line;
            split_line;
        else
            split_line;
            append_line;
        endif;
        copy_text (fao ("!#* ", the_offset - 1));
    endif;
    move_vertical (1);
endloop;

set (saved_mode, current_buffer);
set (LEFT_MARGIN, current_buffer, saved_left_margin);
position (saved_mark);

endprocedure;
```

Code to Implement the Second Strategy

The procedure USER_SET_MARGINS is as follows:

```
!  The following procedure associates user-specified margin
!  values with the current buffer by storing the values
!  in arrays.  This prevents the user-specified values
!  from creating "hard" margin settings.  Instead, the
!  left margin is set in Column 1.
!
!  This procedure takes three parameters.  The first
!  parameter is the buffer with which you want to
!  associate the specified margin values.  The second
!  third parameters are integers obtained from the
!  user specifying, respectively, the left and right
!  margin settings.
!
procedure user_set_margins (the_buffer, the_left, the_right)

on_error
    [OTHERWISE]:
endon_error;

if get_info (the_buffer, "type") <> BUFFER
then
    message (TPU$_INVPARAM, 0,
              1,
              "", str (get_info (the_buffer, "type")),
              "", "BUFFER");
    learn_abort;
    return (FALSE);
endif;
```

```
if get_info (the_left, "type") <> INTEGER
then
    message (TPU$_INVPARAM, 0,
             2,
             "", str (get_info (the_left, "type")),
             "", "INTEGER");
    learn_abort;
    return (FALSE);
endif;

if get_info (the_right, "type") <> INTEGER
then
    message (TPU$_INVPARAM, 0,
             3,
             "", str (get_info (the_right, "type")),
             "", "INTEGER");
    learn_abort;
    return (FALSE);
endif;

if get_info (user_left_margin, "type") <> ARRAY
then
    user_left_margin := create_array;
endif;

if get_info (user_right_margin, "type") <> ARRAY
then
    user_right_margin := create_array;
endif;

user_left_margin {the_buffer} := the_left;
user_right_margin {the_buffer} := the_right;

!    The code above uses array variables to store the margins.  Since
!    array elements can be accessed using either braces or parentheses
!    array variables can be used as if they were procedure calls.
!    The following statements explicitly declare these variables.

variable   user_left_margin;
variable   user_right_margin;


endprocedure;
```

7.3.2 Incompatibility with Future Versions

In this version of EVE, the word GOLD has assumed the meaning formerly assigned to SHIFT. For this version only, the word SHIFT is preserved as a synonym for GOLD. For example, in Version 2.0 SET NOSHIFT KEY is a synonym for SET NOGOLD KEY and SET SHIFT KEY is a synonym for SET GOLD KEY. In a future version of EVE, SHIFT will no longer be synonymous with GOLD.

To avoid the need to rewrite code later, you may want to use GOLD rather than SHIFT wherever possible.

7.3.3 Problems in EVE

This version of EVE contains the following problems. Many of these problems will be resolved in the next version of EVE:

1 The EVE commands UPPERCASE WORD and LOWERCASE WORD do not move the cursor correctly when operating on a range selected as the result of the FIND command. These commands, after changing the letters in the range to uppercase or lowercase, are supposed to move the cursor from the beginning of the found range to the beginning of the first word after the end of the range. However, the commands do not move the cursor at all.

   Presently there is no solution for this problem.

2 If you use the F13 key to execute the ERASE WORD command while the cursor is in the command window to erase the first word of a command in the command buffer, EVE erases the word but also moves the command buffer's editing point to the preceding line. As a result, repeated presses of the F13 key can eventually erase all the commands in the command buffer.

   There is no way to avoid this problem. DIGITAL recommends that you use care when pressing the F13 key while the cursor is in the command window.

3 EVE's EDT-style FIND NEXT key (PF3) does not work correctly after you enter the following sequence of EVE commands:

   - SET KEYPAD EDT
   - SET FIND WHITESPACE
   - WILDCARD FIND pattern-search-string or FIND string-containing-spaces

   If you press the FIND NEXT key under these circumstances, EVE does not find the next occurrence of the string you specified. Instead, EVE displays a VAXTPU error message that is not supposed to be user-visible. EVE might also search for a string other than the one you specified.

   To avoid this problem on a VT300-series or VT200-series terminal, press the FIND key twice to implement FIND NEXT. To avoid this problem on a VT100-series terminal, bind the FIND command to a key and press that key twice to implement FIND NEXT.

4 EVE's EDT-style DELETE TO START OF WORD keys (F13, LF, and CTRL/J) do not work correctly when the cursor is positioned beyond the end of the line and beyond the right edge of the screen. In such a case, pressing the key correctly deletes the last word in the line. However, subsequent presses of the key delete the wrong words.

5 Several EVE commands and several defined keys do not work correctly if you use the command or key while the cursor is positioned to the left of the left margin or in the middle of a tab.

   The commands affected by this problem are ERASE WORD and ERASE CHARACTER. The EDT-style keys affected by the problem are the PF4 (DELETE LINE), MINUS (DELETE WORD), COMMA (DELETE CHARACTER), and GOLD/KP2 (DELETE TO END OF LINE). The WPS-style key sequences affected by the problem are GOLD/F13 (DELETE

TO BEGINNING OF SENTENCE) and GOLD/LINEFEED (DELETE TO BEGINNING OF SENTENCE).

Under these circumstances, the cursor incorrectly moves to the nearest text and deletes the character, word, line, or sentence that the cursor has moved to. EVE is supposed to insert padding spaces between the cursor location and the nearest text and then perform the specified delete or erase operation. Because EVE's correct behavior may delete or erase the padding spaces, the insertion of the spaces before performing the operation produces a different result when the operation is finished.

6 EVE does not display HELP correctly if you use the DEFINE KEY command to do any of the following:

- Binding the HELP KEYPAD command to any key or key sequence, including the HELP key.
- Binding the HELP KEYS command to any key or key sequence, including the sequence GOLD/HELP.
- Binding any command declared as a synonym for HELP KEYS or HELP KEYPAD to any key or key sequence.

If you have used the DEFINE KEY command in any of these ways, the key or key sequence to which you have bound the HELP KEYS or HELP KEYPAD command does not produce the correct results when you press the key more than twice.

The correct behavior is as follows for multiple presses of the key executing HELP KEYS or HELP KEYPAD:

- When the user first presses the key to which HELP KEYPAD (or HELP KEYS) is bound, EVE displays the keypad diagram (or list of defined keys).
- If the user presses the key again (without any other intervening key presses), EVE displays information on the HELP command.
- If the user continues to press the key, the screen display toggles back and forth between the keypad diagram (or list of defined keys) and information on the HELP command.

If you define a key to execute HELP KEYS, HELP KEYPAD, or a synonym for either of these commands, the first press of the key correctly displays the keypad diagram or list of defined keys. The second press of the key correctly displays information on HELP or on the word you have defined as a synonym. However, subsequent key presses do not cause the screen displays to toggle. Instead, EVE displays the message "You are already viewing the help on topic: <name-of-topic> ."

You can avoid this problem if you are able to substitute a VAXTPU statement for the DEFINE KEY command that causes the problem. For example, suppose you had bound the HELP KEYPAD command to the sequence GOLD/H using the following command:

```
DEFINE KEY=GOLD/H HELP KEYPAD
```

If you were working in a context that allowed you to substitute a VAXTPU statement for this command, you could use the following statement:

```
define_key ("on_error" +
            " [TPU$_CONTROLC]:" +
            "  eve$learn_abort;" +
            "  abort;" +
            " [OTHERWISE]:" +
            " ENDON_ERROR;" +
            " return eve_help ('keypad')",
            key_name ("H", SHIFT_KEY),
            "EVE help (help keypad)",
            eve$x_user_keys);
```

7 Eight EVE variables were designated as public (using EVE$X_) when they should have been designated as reserved for DIGITAL (EVE$$X_). The incorrectly named variables are as follows:

- EVE$X_M1DOWN_LEARN
- EVE$X_M1UP_LEARN
- EVE$X_M2DOWN_LEARN
- EVE$X_M2UP_LEARN
- EVE$X_M3DOWN_LEARN
- EVE$X_M3UP_LEARN
- EVE$X_M4DOWN_LEARN
- EVE$X_M4UP_LEARN

Although these are designated as public, you should not attempt to modify them.

These variables will either be correctly designated or removed in the next version of EVE.

8 If you define a GOLD key in EVE using a VAXTPU statement instead of an EVE command, EVE does not undefine the GOLD key correctly. This problem interferes with both the SET NOGOLD and SET NOSHIFT commands, and the problem is triggered regardless of whether the VAXTPU statement is used in a command file, compiled from a command file into a section file, or used interactively during an editing session.

When you use the SET NOGOLD command or SET NOSHIFT command after defining a GOLD key using a VAXTPU statement, EVE returns the error message "There is no user GOLD key currently set." Although this message appears to say that the GOLD key has successfully been undefined, what it really means is that EVE does not recognize that a GOLD key was ever defined.

When you see the message "There is no user GOLD key currently set", you cannot undefine the GOLD key using SET NOGOLD KEY or SET NOSHIFT KEY. Also, you cannot set a new GOLD key by using SET GOLD KEY or SET SHIFT KEY and letting EVE prompt you to press the key you want to define. If you attempt to do so, EVE does not respond when you press the key you want to define. You can break out of this inaction by pressing any key, but when you do so, EVE returns the message "You may press only a single key when setting the GOLD key."

To redefine a gold key in these circumstances, you can use either of the following approaches:

- Use SET GOLD KEY or SET SHIFT KEY and, without prompting, specify the name of the key you want to define as the GOLD key.
- Undefine the GOLD key using the VAXTPU statement SET (SHIFT_KEY, KEY_NAME (PF1, SHIFT_KEY)). Then set the GOLD key using the SET GOLD KEY or SET SHIFT key command.

9 The SPELL command suspends screen activity if your subprocess quota is too low to permit creation of the two subprocesses needed to run DECspell from EVE. If your subprocess quota is too low, when you enter the SPELL command EVE stops updating the screen. There is no further visual feedback of what you type.

To activate your screen after encountering this problem, you must use a VAXTPU statement to turn screen updating back on. Unfortunately, you must type in the statement without being able to see what you are typing, so you must type with care. Do the following:

1 Press the DO key.

2 Type the EVE command TPU, but do not press the RETURN key.

3 Type the VAXTPU statement SET (SCREEN_UPDATE, ON).

4 Press the RETURN key.

If you have done this correctly, the screen reactivates.

If you are an experienced EVE programmer and want to modify your local copy of EVE to fix the problem, find the procedure EVE$$SPELL in the file EVE$EXTRAS.TPU. In the error handler in EVE$$SPELL, find the clause for TPU$_CREATEFAIL. Change the code to the following:

```
[TPU$_CREATEFAIL]:
    SET (SCREEN_UPDATE, ON);
    eve$message (EVE$_CANTCREASPELL);
    eve$learn_abort;
    RETURN (FALSE);
```

10 The LEARN command instructs you to press CTRL/R to remember the keystrokes in a learn sequence, even if you have bound the REMEMBER command to a different key or sequence. If REMEMBER is bound to another key or sequence and not to CTRL/R, use the correct key or sequence to terminate the learn sequence.

11 The commands SAVE EXTENDED TPU and SAVE EXTENDED EVE display a message "Compilation completed without errors" if EVE has been extended successfully. You can ignore this message.

12 If you specify an ambiguous string while entering an EVE HELP topic, the EVE choices window may not display all the topics matching the string you have typed. If the string you entered matches both commands and informational topics, EVE displays all possible commands but none of the informational topics. If the string you entered matches only informational topics, EVE displays all possible informational topics.

13 If you press the Next Screen key or Prev Screen key several times in rapid succession, EVE scrolls through each screen of text instead of optimizing text movement and simply displaying the final result.

7.3.4 Restrictions in EVE

The restrictions in EVE not described in the documentation are as follows:

1. You cannot define a self-inserting key as the GOLD key.
2. The REPEAT command does not work well in initialization files. Execution of the initialization file stops when a REPEAT command is encountered, and EVE prompts for the command to repeat and the number of times to repeat it. Current plans call for this restriction to be removed in a future release of EVE.
3. You can get help on a user-written command (assuming you have supplied a user HELP library) by pressing the DO key, entering the command HELP, entering the name of the command, and pressing the RETURN key. However, if you define a key using the EVE DEFINE KEY command, you cannot get HELP on a user-written command, even if you have supplied a user HELP library, by pressing the HELP key and then pressing the key bound to that user command.
4. Version 2.0 of EVE defines a supported programming interface. Routines, variables, and constants with the prefix EVE$ are supported for public use. Those with the prefix EVE$$ are private to EVE development and may change without notice. Routines with the prefix EVE_ are also supported for public use. If you decide to modify or replace a routine starting with EVE$ or EVE_, first locate all calls to the routine. Your modification could have unintended effects on code that calls the modified routine.
5. None of the following language elements is allowed as the subject of an !%IF-!%THEN-!%ELSE statement during a conditional compilation of EVE$BUILD:

   - An expression
   - A procedure invocation
   - A literal

   Under Version 2.0 of EVE$BUILD, the subject of such an !%IF-!%THEN-!%ELSE statement must be either a TPU variable or a named constant. For example, none of the following statements would be acceptable:

```
!% IF 0
!% IF A<>B
!% IF REVERSE
!% IF NOT EVE$K_FOO_OPTION
```

6. EVE$BUILD does not produce a log file if /NODISPLAY is used on the DCL command line. In addition, EVE$BUILD will not produce a log file if /DISPLAY is used on the DCL command line and the build produces errors.
7. When EVE's WPS-like keypad is enabled, the sequences GOLD/PF3 and GOLD/PF4 do not work the same way in EVE that they do in WPS or WPS-Plus.

   In WPS and WPS-Plus, both these keys undelete the last character, word, line, or sentence that was deleted. In EVE with the WPS-like keypad enabled, GOLD/PF3 and GOLD/PF4 are bound to the RESTORE commmand. This command restores the last word, line, or sentence that was deleted, but does not restore the last character that was deleted.

To make EVE's WPS-like keypad more like WPS, you can bind the EVE command RESTORE CHARACTER to the sequence GOLD/PF4. Because EVE's WPS-like keypad defines PF4 as DELETE CHARACTER, you may find it helpful to have GOLD/PF4 undelete a deleted character.

7.3.5 Problems in EVE Documentation

Appendix F of the *VAX Text Processing Utility Manual* does not clearly explain how to define a key for use in EVE in a command file or other VAXTPU program. DIGITAL recommends you not define a key merely by using the DEFINE_KEY built-in, because DEFINE_KEY does not automatically handle errors. There are different recommended methods for defining a key in EVE, depending on whether the code you are binding to the key returns a result. If you are defining a key using code that returns a result, use the procedure EVE$DEFINE_KEY. For example, if you wanted to bind the EVE command TOP to the sequence CTRL/P, you would use the following statement:

```
EVE$DEFINE_KEY ("eve_top;", CTRL_P_KEY, "TOP");
```

If you are defining a key using code that does not return a result, use the following constant declaration before defining any keys:

```
CONSTANT USER_KT_HANDLER :=
                  "ON_ERROR" +
                  "  [TPU$_CONTROLC]:" +
                  "  EVE$LEARN_ABORT;" +
                  "  ABORT;" +
                  "  [OTHERWISE]:" +
                  "  ENDON_ERROR;";
```

After defining the constant USER$KT_HANDLER, use this constant as part of the first parameter to DEFINE_KEY. For example, if you wanted to bind the VAXTPU statement MOVE_VERTICAL (-1) to the sequence CTRL/K, you would use the DEFINE_KEY built-in as follows:

```
DEFINE_KEY (USER_KT_HANDLER + "MOVE_VERTICAL (-1)", CTRL_K_KEY, "Up");
```

7.4 Command Procedures — Restriction

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.

7.5 Extended File Names or File Types — Caution

Beginning with VMS Version 4.0, file names and file types of up to 39 characters were permitted. Note that you should use caution when you name files because there are some files that may require you to use the VMS Version 3.n maximum lengths (9 characters for the file name and 3 characters for the file type) or other maximum lengths as appropriate.

You must use caution when naming files that will be accessed by the following:

- Operating systems that do not support longer file names and file types, such as VMS Version 3.n systems and systems for PDP–11 processors.
- Applications software that does not accept longer file names and file types.

In addition, use caution when you are naming files that will be copied or accessed by remote systems. The file naming abilities of the VMS operating system after Version 4.0 exceed those of most other computer systems, including VAX systems running VMS Version 3.n.

For example, a system running VMS Version 3.n returns a syntax error when a file specification contains a file name (including a directory name) longer than 9 characters, a file type longer than three characters, a dollar sign ($) or an underscore (_). Valid file specifications of the VMS operating system after Version 4.0 that are invalid on a VMS Version 3.n system include the following:

```
NODE::DBA2:[YOUR_DIR]FILE.DAT
NODE::DBA2:[DIR]FILETOOLONG.DAT
NODE::DBA2:[DIR]FILE_TEST.DAT
NODE::DBA2:[DIR]FILE.DATA
```

You would have to rename these files on a VMS Version 4.0 system (or later) before the remote system could access them. Alternatively, you could copy these files to the remote system by using valid VMS Version 3.0 output file specifications.

File name restrictions are generally determined by the file name capabilities of the remote systems that require access to the file. Such restrictions should be considered as part of the overall application design when network access is required.

Applications that parse file specifications using the pre-Version 4.0 file specification conventions should be modified to use the services or routines that can parse or scan file specifications using the new extended file specification conventions. These services and routines include the RMS $PARSE service and the $FILESCAN system service (see the *VMS Record Management Services Manual* and the *VMS System Services Reference Manual*) and the LIB$FIND_FILE and LIB$FILE_SCAN routines (see the *VMS RTL Library (LIB$) Manual*).

7.6 Primary as Opposed to Alias Directory Entries

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 Section 8.71.1.
- 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:

```
$ CREATE [A]X.X
$ SET FILE [A]X.X/ENTER=[A]Y.Y
$ DELETE [A]Y.Y;
```

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

7.7 File Definition Language Facility — Restriction

The File Definition Language Facility (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"
```

7.8 Symbol Names — Caution

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.

7.9 Symbol Substitution for F$VERIFY Lexical Function

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 in a future version of the VMS operating system.

8 System Manager Release Notes

This chapter includes information about VMS Version 5.0 that is of interest to the system manager.

8.1 Installation and Upgrade Information

The following sections contain additional information relating to your VMS Version 5.0 installation and upgrade.

8.1.1 Installing VMS on a VAX 8200 from an HSC Disk Drive

This note contains information on installing the VMS operating system on a VAX 8200 from an HSC disk drive. Read this note before installing the VMS operating system on a VAX 8200 from an HSC disk drive.

8.1.1.1 How to Proceed

If you have edited the CIBOO.CMD file on the console RX50 to reflect your system configuration, you can use the BOOT58 program and CIBOO.CMD to boot the VMS operating system automatically from an HSC disk drive. Install the VMS operating system as described in Chapter 7 of *VMS Installation and Operations: VAX 8200, 8250, 8300, 8350.*

If you have not edited the CIBOO.CMD file to reflect your system configuration, the BOOT58 program displays syntax error messages when you try to boot the VMS operating system automatically. You must deposit values at the *BOOT58>* prompt to boot the VMS operating system manually from an HSC disk drive. If you have not edited CIBOO.CMD, install the VMS operating system as described in the following sections.

Once you have installed and booted the VMS operating system and are logged in to the SYSTEM account, you can edit CIBOO.CMD. Then you can use CIBOO.CMD to perform subsequent boots.

8.1.1.2 Installing the VMS Operating System

If you have not edited CIBOO.CMD, follow this procedure to install the VMS operating system from an HSC drive.

Prepare for the installation as described in the *VMS Installation and Operations: VAX 8200, 8250, 8300, 8350.* Follow the instructions in Chapter 7, Sections 7.1 through 7.3. Obtain the following information from either the network or VAXcluster manager:

- VAXBI node number
- HSC node number
- HSC controller number or numbers
- Unit number of the disk drive

Then boot standalone BACKUP as follows:

1. Identify the disk drive that contains the distribution disk and the HSC that controls it. You need to know the node number of the HSC and the unit number of the disk drive.
2. Make sure the console RX50 is in the console diskette drive, CSA1.
3. To start the BOOT58 program, enter the following command and press RETURN:

```
>>> B/R5:800 CSA1
```

4. At the *BOOT58>* prompt, deposit the following values. Note that all numeric entries are made using hexadecimal notation.

   a. Deposit the value 20 for the type code for the boot device for the HSC on a CIBCI or CIBCA device into register 0 using the following command:

```
BOOT58> D/G 0 20
```

   b. Deposit the VAXBI node number into register 1 using the following format:

```
BOOT58> D/G 1 node-number
```

   For example, if the VAXBI is node 6, enter the following and press RETURN:

```
BOOT58> D/G 1 6
```

   c. Deposit the HSC node number into register 2 using the following format:

```
BOOT58> D/G 2 node-number
```

   For example, if the HSC is node 12 on a CIBCI device, enter the following command and press RETURN:

```
BOOT58> D/G 2  C
```

Note: If the drive holding the system disk is accessible to two HSCs, deposit both node numbers. Put the greater number in hexadecimal digits 3 and 2, and the lesser in digits 1 and 0. For example, suppose one HSC is numbered 18 (hexadecimal 12) and the other is numbered 10 (hexadecimal A), enter the following and press RETURN:

```
BOOT58> D/G 2 120A
```

   d. Deposit the unit number of the drive holding the system disk into register 3 using the following format:

```
BOOT58> D/G 3 unit-number
```

   For example, if the drive holding the system disk is unit number 21, deposit hexadecimal 15 into register 3:

```
BOOT58> D/G 3 15
```

   e. Deposit the number of the correct root directory from which to boot using the following format. By default, the VMS operating system is in SYS0.

```
BOOT58> D/G 5 r0000000
```

For example, if the correct root directory is SYS0, enter the following and press RETURN:

```
BOOT58> D/G 5 0
```

f. Enter the following commands and press RETURN after each one:

```
BOOT58> D/G 4 0

BOOT58> D/G E 200

BOOT58> LOAD VMB.EXE/START:200

BOOT58> START 200
```

5 The procedure may ask for the date and time. Enter the date and time using the 24-hour clock format and press RETURN. For example:

```
VAX/VMS Version V5.0 Major version id = 01 Minor version id = 00

PLEASE ENTER DATE AND TIME (DD-MMM-YYY HH:MM:) 31-DEC-1988 13:00
```

6 The procedure displays a list of the local devices on your system and, if you have them, HSC- and MSCP-served devices. For example:

```
Available device DJA2    device type RA60
Available device DJA3    device type RA60
```

7 Check the list of devices. If the list is incomplete, make sure that all the drives are connected properly to the system. See your hardware manuals for details.

8 When standalone BACKUP finishes booting, it displays an identification message followed by the dollar sign prompt ($):

```
%BACKUP-I-IDENT, Stand-alone BACKUP V5.0; the date is 31-DEC-1988 13:00:00.00)
$
```

Continue installing the VMS operating system as described in the *VMS Installation and Operations: VAX 8200, 8250, 8300, 8350*. Follow the instructions in Chapter 7, Sections 7.5 through 7.8.

8.1.1.3 Editing CIBOO.CMD

Once you have installed and booted the VMS operating system and are logged in to the SYSTEM account, edit the CIBOO.CMD file using the following procedure:

1 Make sure the console RX50 is in the console diskette drive, CSA1.

2 To connect the console drive to the system, enter the following commands and press RETURN after each one:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CONNECT CONSOLE
SYSGEN> EXIT
```

3 Use the Exchange Utility to copy CIBOO.CMD to the system disk. Enter the following command and press RETURN:

```
$ EXCHANGE COPY CSA1:CIBOO.CMD *.*
```

4 To edit CIBOO.CMD, enter the following command and press RETURN:

```
$ EDIT CIBOO.CMD
```

5 CIBOO.CMD contains the following text:

```
!CIBOO.CMD   :Boot command file to boot a VAX 8200/8300 from an HSC disk.
!
!
!Note "n", "p" (and "q"), "u", and "r" are single hexadecimal characters
!
D/G 0 20                ! CI Port Device Type Code
!D/G 1 n                ! n = CI adaptor's VAXBI node number
!D/G 2 p                ! Use the HSC controller at CI node p
!D/G 2 0p0q             ! Use either the HSC controller at CI nodes p and q
!D/G 3 u                ! u = Disk drive unit number
D/G 4 0                 ! Boot Block LBN (not used)
!D/G 5 r0000000         ! r = system root [SYSR...], Software boot flags
D/G E 200               ! Address of Working Memory+^X200
LOAD VMB.EXE/START:200  ! Load Primary Bootstrap
START 200               ! Start Primary Bootstrap
```

a. Delete the comment character (!) that appears before the D/G 1 command and replace *n* with the VAXBI node number of the CI adapter.

b. If your computer is only connected to one HSC controller, delete the comment character (!) that appears before the first D/G 2 command. Replace *p* with the HSC controller number in hexadecimal notation. Delete the second D/G 2 command.

If the drive holding the system disk is connected to two HSC controllers, do the following:

- Delete the comment character (!) that appears before the second D/G 2 command
- Replace *p* with the controller number of the first HSC in hexadecimal notation
- Replace *q* with the controller number of the second HSC in hexadecimal notation
- Delete the first D/G 2 command

c. Delete the comment character (!) that appears before the D/G 3 command and replace *u* with the unit number of the HSC drive from which you will boot the VMS operating system. Use hexadecimal notation.

d. Delete the comment character (!) that appears before the D/G 5 command and replace *r* with the number of the correct root directory from which to boot. By default, the VMS operating system is in SYS0.

e. Exit from the editor.

6 Use the Exchange Utility to copy CIBOO.CMD to the console RX50. Enter the following command and press RETURN:

```
$ EXCHANGE COPY CIBOO.CMD CSA1:CIBOO.CMD
```

7 When you are finished, enter the following command and press RETURN:

```
$ DISMOUNT CSA1
```

8 To secure the console from unauthorized access, you must enter the following command and press RETURN:

```
$ MOUNT/FOREIGN/SYSTEM/NOWRITE/NOASSIST CSA1
```

If you change the system configuration, use the previous procedure to modify CIBOO.CMD.

8.1.2 Tape Device Names for MicroVAX, VAXstation, and VAXserver 3600 Series Computers Installations

This section contains information on the correct format to use when you refer to a tape cartridge or magnetic tape drive by its device name. Read this note before installing the VMS operating system from magnetic tape or tape cartridge on a MicroVAX I or II, VAXstation I, II, or II/GPX, or VAXserver 3600 series.

8.1.2.1 Device Names for Tape Cartridge and Magnetic Tape Drives

When you refer to a magnetic tape drive or a tape cartridge drive with its device name, use the following format:

MUcu

where:

- *MU* is the *device code*. The device code tells what type of device you are using.
- *c* is the *controller designation*. A controller designation can be one of the alphabetic letters A through Z. The controller designation, along with the unit number, identifies the device.
- *u* is the *unit number*. A unit number can be a decimal number in the range of 0 to *n*. The unit number, along with the controller designation, identifies the device.

For example, if you are referring to a tape cartridge drive with a controller designation of B and a unit number of 1, use the device name MUB1.

8.1.3 RK07 Installation Distribution Kit

Both *VMS Installation and Operations: VAX-11/750* and *VMS Installation and Operations: VAX-11/780, 785* list only one RK07 disk for the RK07 distribution kit. If you ordered an RK07 kit for Version 5.0 of the VMS operating system, you received two RK07 disks in the kit. They are labeled as follows:

VMS V5.0 BIN RK07 1/2
VMS V5.0 BIN RK07 2/2

To install the VMS operating system, place the disk labeled VMS V5.0 BIN RK07 1/2 in the RK07 drive. Spin up the drive. Follow the appropriate installation procedure in the installation and operations guide for your VAX computer.

After the procedure restores most of the optional save set, it asks you to remove the first RK07 disk and replace it with the second one. For example:

```
%BACKUP-I-RESUME, resuming operations on volume 2
%BACKUP-I-READYREAD, mount volume 2 on DMcu: for reading
Enter "YES" when ready:
```

Spin down the drive and remove the RK07 disk labeled VMS V5.0 BIN RK07 1/2. Place the disk labeled VMS V5.0 BIN RK07 2/2 in the RK07 drive. Spin up the drive. When you are ready to continue, type Y and press RETURN. Continue with the installation procedure.

8.1.4 Error Message Displayed During AUTOGEN

Sometimes error messages are displayed during the AUTOGEN procedure that occurs at the end of a VMS Version 5.0 installation. Usually the errors are report "invalid stream identifier" or "improperly handled condition" messages. However, other error messages may be displayed.

If this occurs, you should halt your system and reboot it. Depending during which phase of AUTOGEN the errors occurred, the system either displays that you have successfully installed VMS or it reapplies the mandatory update and invokes AUTOGEN again.

8.1.5 Upgrade Procedure Driver Version Mismatch Errors

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.

8.1.6 ANALYZE/ERROR_LOG Restriction for Rolling Upgrades

During the rolling upgrade procedure, if it becomes necessary to analyze an error log file for a VMS operating system version other than the version that wrote it, ERF will *not* correctly translate all bugcheck codes.

8.2 Asymmetric Multiprocessing (ASMP) and VAX–11/782 — Support Discontinued

Beginning with VMS Version 5.0, support for the VAX–11/782 and for Asymmetric Multiprocessing (ASMP) has been discontinued. Customers with VAX–11/782 systems should contact their DIGITAL sales representative to discuss hardware upgrade alternatives. ASMP has been superseded by Symmetric Multiprocessing (SMP), which provides more predictable performance over a wider range of workloads. See Section 7.1.5 for additional information.

8.3 System STARTUP Procedure — Notes

VMS Version 5.0 contains a new system startup procedure. This procedure includes a new support subfunction to the SYSMAN Utility (the creation of several system- and cluster-wide data files), a new startup command procedure, called SYSTARTUP_V5.COM.

The system startup procedure has been modified as follows:

1 STARTUP.COM command procedure

The file SYS$SYSTEM:STARTUP.COM has been revised to function as a master procedure that invokes other startup command procedures.

2 STARTUP directory

A new clusterwide directory, SYS$STARTUP, has been added to contain command procedures run by STARTUP.COM.

3 STARTUP files

Several command procedures that reside in the SYS$STARTUP directory contain the information previously executed by STARTUP.COM. Three data files also reside in this directory containing information controlling the execution of the STARTUP.COM command procedure.

4 Renaming of site-specific startup command procedure

The file SYS$MANAGER:SYSTARTUP.COM has been renamed to SYS$MANAGER:SYSTARTUP_V5.COM for rolling upgrade purposes.

5 SYSMAN Utility startup subfunction

A set of commands has been added to the SYSMAN Utility to modify and display information stored in the SYS$STARTUP data files.

The following sections provide a description of these enhancements to the startup command procedure.

8.3.1 The STARTUP.COM Command Procedure

The VMS Version 4.n SYS$SYSTEM:STARTUP.COM has been significantly enhanced for VMS Version 5.0 as a master procedure that invokes other startup command procedures.

The startup command procedure runs in a sequence of requirements or "phases." Each phase describes the minimum environment available during that phase so that it can be determined whether a component should run. During each phase, the startup procedure brings the system up to the requirements of the next phase. Each phase has one or more startup components that are described in the STARTUP database. These components build upon the existing environment and have no interdependencies. All components of a phase have to be completed before the next phase can begin. Thus, the startup procedure enters a phase, starts all the components to be completed in the phase in an unspecified order, and waits for all components to complete before starting the process for the next phase.

There are four base VMS operating system phases to get the system from SYSINIT to completion of the basic VMS environment, and four phases for layered product and site-specific use. The VMS operating system phases are as follows:

- INITIAL
- CONFIGURE
- SYSFILES
- BASEENVIRON

You should not modify the execution of these phases because any change to a phase may produce uncertain results in the system startup procedure.

8.3.2 The SYS$STARTUP Directory

Startup component files are individual command procedures or executable files in the VMS Version 4.n SYS$SYSTEM:STARTUP.COM. For VMS Version 5.0, a new directory, SYS$STARTUP, has been added to SYS$COMMON to store these files. This directory is referenced by the logical name SYS$STARTUP, which, for VMS Version 5.0, is a search list pointing to the new SYS$SYSROOT:[SYS$STARTUP] directory and the SYS$MANAGER directory. SYS$STARTUP should be the directory used to store any layered product or site-specific startup files that use the new system STARTUP procedure.

8.3.3 The Startup Data Files

Three data files have been added in the SYS$STARTUP directory to pass information to the startup command procedure. The first file is the phase file, referred to by the logical name STARTUP$PHASES. This file is a sequential list of phases that the command procedure runs. The other two files are component data files, called by their logical names STARTUP$STARTUP_VMS and STARTUP$STARTUP_LAYERED. Both are indexed sequential files and identical in format. However, the contents of each file are different. Combined, these two files make up the startup database, which contains the following fields:

- The names of either EXE or COM component files that are executed by the startup command procedure.
- The phase in which each component file is to be executed.
- The way the file is executed. The startup procedure may execute a component file directly, run it in batch mode, or spawn it as a subprocess.
- The restrictions to the file. The restriction field is available so that you can list nodes where this file may or may not be executed. The restriction field is site specific.
- A byte field specifies whether the file should be executed on the nodes or not. This field is also site-specific.
- The parameters that may be used in the component file.

STARTUP$STARTUP_VMS describes files that start the base VMS environment. It is provided as a means for displaying information. DIGITAL does not recommend or support changes made to STARTUP$STARTUP_VMS.

STARTUP$STARTUP_LAYERED includes information on site-specific layered-product and third-party startup files. It may be modified using the SYSMAN Utility.

Both files are used in conjunction with system startup and system management. The purpose of creating two component data files is to allow the STARTUP$STARTUP_LAYERED file to be left intact when a VMS operating system upgrade is being performed and, if necessary, allow the STARTUP$STARTUP_VMS file to be modified. Note that initially STARTUP$STARTUP_LAYERED only contains the site-specific startup file (see Section 8.3.4) until layered products are installed or the site-specific command procedures have been included.

8.3.4 New Name for the Site-Specific Startup Procedure

SYS$MANAGER:SYSTARTUP.COM is used by system managers to perform site-specific startup operations. To comply with rolling upgrade constraints and VMS Version 5.0 changes, two versions of SYSTARTUP.COM have been created: one for VMS Version 5.0 systems and one for VMS Version 4.7 systems. Note that a separate copy of SYSTARTUP.COM must exist for both Version 4.7 and Version 5.0 systems.

For Version 4.7 systems, STARTUP.COM still invokes SYS$MANAGER:SYSTARTUP.COM. However, VMS Version 5.0 STARTUP.COM invokes a new file called SYS$MANAGER:SYSTARTUP_V5.COM. In general, both files are similar—the main difference is that Version 5.0 layered products may be started from the startup component data file. Thus, a VMS Version 5.0 site-specific startup command procedure must be named SYS$MANAGER:SYSTARTUP_V5.COM to be invoked by STARTUP.COM. No internal changes to this file are required by the system STARTUP procedure.

8.3.5 SYSMAN Utility STARTUP Subfunction

A command set called STARTUP has been included in the SYSMAN Utility. This provides system managers with the software tool for managing the new STARTUP process. The SYSMAN command set STARTUP contains the following commands:

Function	Description
ADD	Adds files to the startup data files
DISABLE	Disables a file from running on one or more nodes in a cluster
ENABLE	Enables a file to run on one or more nodes in a cluster
MODIFY	Modifies information in an entry to a startup component file such as the phase in which it runs
REMOVE	Removes files from the startup data files
SET DATABASE	Establishes the default database
SHOW	Displays information about one or more files stored in a startup component file

Wildcard processing is supported for these functions. All functions require the same privileges required to run SYSMAN. See the *VMS SYSMAN Utility Manual* for more information about using SYSMAN and the STARTUP commands.

8.4 SYENVIRON.COM Obsolete

The command procedure SYS$MANAGER:SYENVIRON.COM is obsolete, and any commands previously residing in this file should be placed in the command procedure SYS$MANAGER:SYLOGICALS.COM.

SYS$MANAGER:SYLOGICALS.COM is a new command procedure for VMS Version 5.0.

8.5 Authorize Utility — Notes

The following sections describe changes in the Authorize Utility.

8.5.1 Network Proxy Authorization File Changes

There have been a number of changes in the operation of proxy logins on the VMS operating system for Version 5.0.

- The format of the proxy login authorization file has changed.
- System managers can grant remote users proxy access to multiple local accounts. System managers can also grant remote users from non-VMS systems proxy access to the local system.

The name of the network proxy authorization file has changed from NETUAF.DAT to NETPROXY.DAT. If you use the logical name NETUAF for the network proxy authorization file, you must change the logical name from NETUAF to NETPROXY. (The logical name NETPROXY must be a systemwide logical name.)

NETPROXY.DAT is treated as a permanent database. Additions and changes to NETPROXY are made from the Authorize Utility using the commands ADD/PROXY, MODIFY/PROXY, and REMOVE/PROXY. NETACP then copies these changes from the permanent database to the volatile copy of NETPROXY on the running system. Note that the volatile database

is automatically updated when any changes are made to the permanent database using the Authorize Utility.

NETPROXY's internal file format has also changed in VMS Version 5.0. The image CVTNAFV5.EXE, located in the SYS$SYSTEM directory, converts existing records in NETUAF to the new format used by NETPROXY as part of the upgrade procedure.

The new network authorization mechanisms used in VMS Version 5.0 allow for a single remote user to have proxy access to one default local account and up to 14 other local accounts. Access to the default account is obtained in the same manner as in previous versions of the VMS operating system; access to any of the nondefault accounts is obtained by specifying the user name associated with the desired proxy account in the access control string of the appropriate network command. The default proxy account must be defined by the system manager. You can do this by using the Authorize Utility commands ADD and MODIFY including the /DEFAULT qualifier. See the *VMS Version 5.0 New Features Manual* for further details.

Note: As part of the update procedure, the CVTNAFV5.EXE image supplied with VMS Version 5.0 marks all previous entries in NETUAF.DAT as default proxy accounts in NETPROXY.DAT. This is done to maintain the consistency of proxy logins with previous versions of the VMS operating system.

The VMS operating system proxy login supports proxy access from users on non-VMS systems. The operating system remote users are identified by UIC, as shown in the following example:

```
UAF> ADD/PROXY GLINCH::[360,54]  COLLINS/DEFAULT
```

In this example, a remote user on node GLINCH (not a VMS system) identified by UIC [360,54] is granted proxy access to the COLLINS user account on the local system.

The Authorize Utility has been enhanced for VMS Version 5.0 to support the new proxy login features. Refer to the *VMS Version 5.0 New Features Manual* for a description of new and changed AUTHORIZE commands related to proxy logins.

Note: The system manager must set one of the proxy accounts as the default proxy account, otherwise there will be no default account for the user.

8.5.2 Changed UAF Limits for SYSTEM and DEFAULT Records

A number of initial values for the SYSTEM and DEFAULT records in AUTHORIZE have been changed for VMS Version 5.0. The following table shows the AUTHORIZE qualifiers associated with the updated UAF limits, along with the old and new initial values of the qualifiers.

Table 8–1 New Initial Values for SYSTEM and DEFAULT Records in AUTHORIZE

SYSTEM Record		
Qualifier	**Old Value**	**New Value**
/BYTLM	20480	32768
/ENQLM	30	200
/FILLM	20	40
/WSDEFAULT	150	256
/WSEXTENT	1024	2048
/WSQUOTA	350	512
DEFAULT Record		
Qualifier	**Old Value**	**New Value**
/ASTLM	24	10
/BIOLM	18	10
/BYTLM	4096	8192
/DIOLM	18	10
/ENQLM	10	100
/PGFLQUOTA	10000	10240
/WSEXTENT	500	512
/WSQUOTA	200	256

8.6 AUTOGEN Command Procedure — Notes

This section describes changes made to the AUTOGEN command procedure for VMS Version 5.0.

8.6.1 AUTOGEN Aborts During GENFILES Phase

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 may occur in the following two ways:

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

```
$ SHOW LOGICAL SYS$SYSROOT
   "SYS$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.

2 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.

8.6.2 Feedback Mechanism Added

AUTOGEN employs a new mechanism whereby information from the running system may be used in its parameter and system file calculations. This allows AUTOGEN to size a system based on how the user's workload is actually using resources.

The VMS executive maintains pertinent information primarily as peak usage of key resources since boot time; for example, maximum number of concurrent processes. A new image, SYS$SYSTEM:AGEN$FEEDBACK.EXE, reads this information and writes it to the file SYS$SYSTEM:AGEN$FEEDBACK.DAT during AUTOGEN's SAVPARAMS phase. This information is read in during the GETDATA phase and is used in the subsequent calculations. A report is generated that shows the parameters and system files affected by feedback, their current and new values, the relevant feedback information, and any user or DIGITAL-supplied modifications or overrides. This report is placed in the file SYS$SYSTEM:AGEN$FEEDBACK.REPORT.

There are situations when it may not be advisable to use this new mechanism; for example, if you have just booted the system and it has not been up long enough to have executed your normal workload.

Refer to the *VMS Version 5.0 New Features Manual* for further details.

8.6.3 Additional Data Files Used with Feedback Mechanism

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.

8.6.4 Oldsite Mechanism Is Obsolete

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.

8.6.5 Mechanism to Control MSCP Server Buffer Size Is Obsolete

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 be 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*.

8.6.6 System Files Not Marked for NOBACKUP

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.

8.6.7 Swap File Size Changes

For Version 5.0, 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.

8.6.8 Dump File Size Changes

For Version 5.0, 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.

8.6.9 Selective Crash Dump Files — Caution

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:

```
@SYS$UPDATE:AUTOGEN SAVPARAMS REBOOT
```

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

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.

8.6.11 Hexadecimal Values Processed Correctly in MODPARAMS.DAT

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.

8.7 Additional Privilege Required to Execute STABACKIT.COM

Section 6 of 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. VMS Version 5.0 also requires that you have either the user privilege PHY_IO or SETPRV to execute STABACKIT.COM.

8.8 Batch Jobs Submitted Using the DCL Command SUBMIT/USER

Batch jobs submitted to run in the context of another user (using the DCL command SUBMIT/USER) now have access to queues based on the identity of the specified user rather than the identity of the submitter. Changes made to the queue protection checking algorithm for VMS Version 5.0, allow a job to execute on queue only when the owner of the job has a UIC, identifier, or privilege, which allows him to enter a batch job directly to the queue.

The VMS Version 4.n behavior of using the UIC and privileges of the submitting process to grant or deny access to queues for a job entered in behalf of another user was both contrary to the documentation and not in line with the enhanced queue protection features for VMS Version 5.0.

8.9 CIBCA — New Device Support

Since Version 4.6, the VMS operating system has supported the CIBCA. The CIBCA is a 2-board computer interconnect (CI) interface for the backplane interconnect (BI) bus and is functionally equivalent to the CIBCI. The CIBCA uses the same driver (PADRIVER) and device mnemonic (PA) as the CIBCI, CI750, and CI780. The CIBCA also uses the same hierarchical storage controller (HSC) booting and installation procedures as the CIBCI.

8.10 Supported Computer Interconnect Port Microcode

System managers should ensure that systems using computer interconnect (CI) interfaces have Version 8.0 of the CI780 microcode for the CI780, CI750, and CIBCI interfaces. Systems using the CIBCA-AA interface should have the Version 5.0 CI microcode.

System managers can determine the current microcode version by executing the DCL command SHOW CLUSTER/CONTINUOUS and then entering the ADD RP_REVIS subcommand at the *COMMAND>* prompt.

For the CI780, CI750, and CIBCI, the low-order word is the random-access memory (RAM) version and the high-order word is the programmable read-only memory (PROM) version. For CI780 Version 8.0 microcode, this field contains 80007_8.

For the CIBCA-AA interface, the high-order word contains the microcode version. The low-order word contains the self-test microcode version, which may be ignored. For CIBCA-AA Version 5.0 microcode, the RP_REVIS field contains 50003_8.

8.11 Debugger — Making Images Shareable for Better Performance

If there are many users of the debugger on your system, you can improve the use of system resources by installing the debugger and some related images as known shareable images.

For VMS Version 5.0, the debugger is composed of the two shareable images SYS$LIBRARY:DEBUG.EXE and SYS$LIBRARY:DEBUGSHR.EXE (prior to Version 5.0, only DEBUG.EXE existed). DEBUGSHR.EXE contains most of the debugger code and is the most important to share among users.

In addition to DEBUG.EXE and DEBUGSHR.EXE, you may also want to install DBGTBKMSG.EXE and SMGSHR.EXE as known shareable images. DBGTBKMSG.EXE is the shareable message file used by the debugger and the traceback utility. SMGSHR.EXE is the screen management facility, which is heavily used by the debugger.

You can install the DEBUG images using the Install Utility. However, you might find it helpful to have the system install the images for you. To do this, place the following commands in the file SYS$MANAGER:SYSTARTUP_V5.COM. Your system will install these images as known shareable images every time the system reboots.

```
$ INSTALL:=$SYS$SYSTEM:INSTALL/COMMAND
$ INSTALL
REPLACE/OPEN/HEADER/SHARED  SYS$LIBRARY:DEBUG.EXE
REPLACE/OPEN/HEADER/SHARED  SYS$LIBRARY:DEBUGSHR.EXE
REPLACE/OPEN/HEADER/SHARED  SYS$MESSAGE:DBGTBKMSG.EXE
REPLACE/OPEN/HEADER/SHARED  SYS$LIBRARY:SMGSHR.EXE
EXIT
```

8.12 DEBNA VAXBI Ethernet Controller — Notes

This section covers the following two topics concerning DEBNA controllers:

- Tuning the VMS operating system for DEBNA
- Configuration and startup

8.12.1 Tuning the VMS Operating System for DEBNA Controllers

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

SYSGEN Parameters

The VMS operating system must be tuned for DEBNA for the following reasons:

- The DEBNA shares some resources with other BI controllers; therefore, these resources need to be tuned to allow them to operate properly.
- The DEBNA is an intelligent controller that requires more resources than other Ethernet controllers. These resources should be tuned properly to allow the DEBNA to operate properly. The DEBNA controller replaces the DEBNT controller and cannot be run on a system that uses the DEBNT. The DEBNT is no longer supported by the VMS operating system.
- Use of AUTOGEN, including FEEDBACK, should ensure that the SYSGEN parameters are set correctly for system configurations with single DEBNA controllers. If your system configuration includes multiple DEBNA controllers, include the following line in SYS$SYSTEM:MODPARAMS.DAT where *n* is the number of controllers in your system:

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

Network Parameters

Check and adjust the network parameters in the following list:

1 LINE BUFFER Size Must Be 1498

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

2 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).

3 **Hello Timer**

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

4 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.

8.12.2 Configuration and Startup

Before starting LAT, assign the following logical name to its controller:

```
$ ASSIGN/SYSTEM  ETxO LAT$DEVICE
```

In this example, *x* represents A for the first DEBNA, B for the second, and so forth.

The first DEBNA is the controller on the lowest-address VAXBI with the lowest VAXBI node ID.

A Local Area VAXcluster is assigned to the first Ethernet controller configured by the VMS operating system.

8.13 DECnet–VAX Notes

The following sections provide information related to the DECnet–VAX software.

8.13.1 Support for X.25 Virtual Circuits Requirement

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.

8.13.2 Constraints on Passive Maintenance Functions Relaxed

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.

8.13.3 Proxy Access Parameters — Changes

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> DEFINE EXEC INCOMING PROXY ENABLED  ! Enable incoming proxy access
NCP> DEFINE EXEC INCOMING PROXY DISABLED ! Disable incoming proxy access
NCP> DEFINE EXEC OUTGOING PROXY ENABLED  ! Enable outgoing proxy access
NCP> DEFINE EXEC OUTGOING PROXY DISABLED ! Disable outgoing proxy access
```

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

```
NCP> SET KNOWN PROXIES ALL
```

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.

8.13.4 MAXIMUM PATH SPLITS Default Value

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.

8.14 DEQNA Ethernet Adapter May Receive Corrupt Data

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 which will reduce 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.

8.15 DMB32 Product Software Required for DMB32 Communications Controller

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.

8.16 DIGITAL Storage Architecture Requirement for Disk Path Failover

DIGITAL Storage Architecture (DSA) disks that are dual-pathed between VAX computers (using devices such as UDAs, KDAs, and BDAs) must be mounted on both systems for path failover to occur.

8.17 Highwater Marking on System Volumes May Cause System Failures

Your VMS system will fail as a result of memory data page corruption when you use the SYSGEN command CREATE to extend the size of the primary paging file SYS$SYSTEM:SWAPFILE.SYS or the primary swapping file SYS$SYSTEM:SWAPFILE.SYS while all of the following conditions exist:

- Highwater marking is enabled on the system volume containing the primary paging or swap file
- The system volume has been restored with BACKUP and the highwater mark on the paging or swap file is incorrect

When this occurs, memory data pages are filled with zeros and the system fails.

You can avoid this problem by temporarily disabling highwater marking on the system volume while using the SYSGEN command CREATE. To do this, enter the following DCL commands:

```
$ SET VOLUME/NOHIGHWATER_MARKING volume-name
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CREATE  . . .
SYSGEN> EXIT
$ SET VOLUME/HIGHWATER_MARKING volume-name
```

Another method that may be easier to perform includes using the command procedure SYS$MANAGER:SWAPFILES.COM. SWAPFILES checks for the preceding conditions and, if necessary, disables highwater marking.

Note: Note that this problem only applies to SYS$SYSTEM:PAGEFILE.SYS or SYS$SYSTEM:SWAPFILE.SYS. All other paging and swapping files (that is, all files installed with the SYSGEN command INSTALL) can be extended while highwater marking is in effect.

A similar problem occurs when you use the ANALYZE /DISK_STRUCTURE /REPAIR command on the system volume. In general, DIGITAL does not recommend that you use this command on a running system. Instead, DIGITAL recommends that the system volume be repaired immediately after it has been backed up and restored with the Backup Utility (BACKUP).

However, if you find it necessary to use the ANALYZE /DISK_STRUCTURE /REPAIR command on the system volume for a running system, you should disable highwater marking on the system volume before invoking ANALYZE. Note the following example:

```
$ SET VOLUME/NOHIGHWATER_MARKING volume-name
$ ANALYZE/DISK_STRUCTURE/REPAIR volume-name
$ SET VOLUME/HIGHWATER_MARKING volume-name
```

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

8.18 Forced Error Handling

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.*

8.19 INITIALIZE Command — Defining Volume Serial Numbers

Since the introduction of DSA disks, the VMS operating system has not properly initialized the Files–11 On-Disk Structure Level for 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.

8.20 LAT Software — Notes

This section discusses known problems, changes, and restrictions affecting the Local Area Terminal (LAT) software. It includes suggested solutions for any known problems.

8.20.1 Delay in Process Disconnect

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.

8.20.2 Solicit Connection QIO

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.

8.20.3 LAT PASSALL Session

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.

8.20.4 LAT Control Program (LATCP) Changes and Restrictions

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.

8.21 LIBRARIAN Routines — Caution when Using Locate Mode

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.

8.22 Modem Signal Requirements now Enforced

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
```

8.23 Monitor Utility — Notes

The following sections provide information about the Monitor Utility for VMS Version 5.0.

8.23.1 MONITOR Display Error

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.

8.23.2 MONITOR RMS Bucket and Multibucket Split Rates Invalid

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.

8.24 Mount Utility — Notes

The following sections discuss changes to the Mount Utility.

8.24.1 MOUNT Qualifier /MULTI_VOLUME — Requirement

VMS Version 5.0 supports a new MOUNT qualifier that pertains to multiple volume tape sets. The /MULTI_VOLUME qualifier is now required when a utility reads from or writes data to a tape mounted with the /FOREIGN qualifier when the data is expected to span two or more reels of tape.

Requiring the /MULTI_VOLUME qualifier for such operations and requiring the user to have the VOLPRO privilege in order to specify the /MULTI_VOLUME qualifier prevents a possible breach of security. The potential for a security breach existed because some utilities mount only the first tape in a set of tapes mounted with the /FOREIGN qualifier; therefore, only the first tape was subject to MOUNT's access checks. VOLPRO privilege is required because it ensures that the user has enough privilege to override any access checks.

DIGITAL recommends that this qualifier be used in cases where it is not possible to alter the utilities in question to explicitly perform MOUNT and DISMOUNT operations on each reel in the set.

The VMS Backup Utility (BACKUP) has been modified for VMS Version 5.0 to use the system services $MOUNT and $DISMOUNT for each volume required, including the first volume. It is, therefore, not necessary to use the /MULTI_VOLUME qualifier to mount a tape for use by BACKUP. For additional information, refer to the *VMS Backup Utility Manual*.

8.24.2 Changes to Mount Verification

VMS Version 5.0 provides support for mount verification of tape volumes. Both American National Standards Institute (ANSI) magnetic tapes and tapes mounted with the /FOREIGN qualifier are subject to mount verification. In addition, the static failover of dual-pathed HSC tape drives added in VMS Version 4.6 has been enhanced to provide dynamic failover of dual-pathed drives. This further increases the usefulness of dual-pathed HSC tape drives in situations where high availability is important. The following list provides additional information about changes in mount verification.

- A dual-pathed HSC tape drive is a drive that connects to two HSCs, both of which have the same nonzero tape allocation class. (The tape allocation class is set using the HSC console command SET ALLOCATE TAPE.) The VMS operating system recognizes the dual-pathed nature of such a tape drive, provided that it has access to both HSCs and that both port select buttons are depressed on the tape drive.

A device enters mount verification when an I/O request fails because the device has become inoperative. This may occur as a result of the device being accidentally placed off line, a hardware error occurring, or the device accidentally being write-protected during a write operation. If the device is off line, mount verification validates the volume once it is restored to the online condition, restores the tape to the position where the I/O failure occurred, and retries the failed I/O request. If the device enters mount verification because it is write-protected, then mount verification simply waits for the device to become write-enabled and retries the failed I/O request.

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 mount verification proceeds as previously described. The operation then continues where it left off, using the alternate path. It is no longer necessary to remount the tape as described in the *VAX/VMS Version 4.6 Release Notes*. (See the *VMS VAXcluster Manual* for a discussion of automatic failover.)

The VMS operating system also continues to select a functional HSC automatically when processing a MOUNT or INITIALIZE command.

- The SYSGEN parameter TAPE_MVTIMEOUT replaces the function of the SYSGEN parameter VMSD3 described in the *VAX/VMS Version 4.6 Release Notes*. If a device is unable to complete mount verification within the timeout specified by the SYSGEN parameter TAPE_MVTIMEOUT, then mount verification is aborted and the I/O operation fails. By default, TAPE_MVTIMEOUT is set to 600 seconds or 10 minutes.

8.25 Modular Executive — Notes

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

8.25.1 Introduction to the Modular Executive

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.

8.25.2 Effects on Privileged Code

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.

8.25.3 Version Numbers and Version Checking

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.

8.26 Modular Executive — Effects upon System Management

The following sections describe the effect of the modular executive upon system images.

8.26.1 SYS$LOADABLE_IMAGES Directory on the System Disk

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.

8.26.2 Version Checking

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.

8.26.3 System Failure

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.

8.27 Modular Executive — Effects upon SYSGEN

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.

8.28 MSCP Server and Diskette Devices

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 allow diskette devices such as the RX01, RX02, and RX33 to be served.

8.29 Network Control Program — Changes

The following are changes to the Network Control Program (NCP):

- NCP now supports multiple command line recall. It allows all the features currently supported with DCL command line recall.
- If you use command line mode and the command returns an error, NCP returns the command's error status in the symbol $STATUS.
- NCP now supports wildcard characters in the command line.
- NCP accepts both the BNT and BNA mnemonics for the DEBNA. However, NCP always displays BNA as the mnemonic.
- The circuit BLOCKING parameter of the NCP command SET/DEFINE CIRCUIT has been removed. This command applied to X.25 data link mapping (DLM) circuits.

8.30 Restriction for RQDX3 Controllers

If you are using RQDX3 controllers on a system that serves disks in a Local Area VAXcluster or a mixed-interconnect VAXcluster, and the RQDX3 controller does not contain a microcode revision level of 3.10 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 using the command ANALYZE/ERROR_LOG.

8.31 Device Unit Number Changed with RQDX3 Controllers

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 erase command. Thus, the file will be allocated but not zeroed.

8.32 VAX RMS Journaling — Notes

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 6.1. You should also read Section 6.13 for additional information.

8.32.1 STREAM Formats not Supported for Shared Sequential Files

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.

8.32.2 CONTROL Access Required to Modify a File Within a Recovery Unit

You must have CONTROL access to all files modified within a recovery unit. If you do not have CONTROL access, the following set of error conditions results:

```
-RMS-F-ACC_RUJ, recovery unit journal can not be accessed
-RMS-F-NOPRIV, no privilege for attempted operation
```

A user has CONTROL access to all files for which the owner user identification code (UIC) matches the user's UIC. A user with sufficient privilege, such as SYSPRV, has CONTROL access to all files on the system.

The system manager or file owner must grant CONTROL access to a user who does not have CONTROL access to a file in order for that user to be able to modify the files successfully. For information on granting CONTROL access to a file, see the *Guide to VMS System Security*.

DIGITAL expects to remove this restriction in a future release of the VMS operating system.

8.32.3 New File Access Block Field for VAX RMS Journaling (FAB$B_JOURNAL)

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 8–2 lists the bits that RMS may set in this field and their meaning.

Table 8–2 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.

8.32.4 Running the VAX RMS Journaling Installation Verification Procedure

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:

```
$ @SYS$EXAMPLES:RUFEXAMPLE
```

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
        SEGO_LENGTH             9
        SEGO_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 searchlist 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 searchlist of directories points.

8.32.5 CONTROL Access Required for After-Image or Before-Image Recovery

In addition to WRITE access, RMS after-image and before-image recovery (the RECOVER/RMS_FILE/FORWARD and RECOVER/RMS_FILE/BACKWARD commands) requires CONTROL access to the file being recovered. If you do not have CONTROL access to the file being recovered, RMS recovery may be unable to close the file after finishing the recovery operation. RMS recovery displays the following error messages if it is unable to close the file:

```
%RMSREC-F-CLOSERR, error closing file
-RMSREC-F-FILE, file file-specification
-RMS-F-NOPRIV, no privilege for attempted operation
```

A user has CONTROL access to all files for which the owner user identification code (UIC) matches the user's UIC. A user with sufficient privilege, such as SYSPRV, has CONTROL access to all files on the system.

The system manager or file owner must grant CONTROL access to a user who does not have CONTROL access to a file in order for that user to be able to recover the file successfully. For more information on granting CONTROL access to a file, see the *Guide to VMS System Security*.

DIGITAL expects to remove this restriction in a future release of the VMS operating system.

8.32.6 Additional VAX RMS Journaling Error Messages

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.

8.32.7 Backup Utility Errors

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 OPCOM, the operator communication process. See Section 4.6.3 of the *VAX RMS Journaling Manual* if detached recovery fails.

8.32.8 The Backup Utility and the SET FILE Command

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.

8.32.9 Errors During the Execution of Recovery Unit Service

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 serve 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 sends error messages to OPCOM, 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
```

Section 4.6 of 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.

8.32.10 WRTJNL_BIJ Error Message

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). Appendix B of the *VAX RMS Journaling Manual* lists the information you need to include with RMS Journaling-specific SPRs.

8.32.11 Files DIGITAL Recommends Not Marking for Journaling

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.

8.32.12 Handling RMS I/O Errors when Journaling

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.

8.32.13 Accessing Indexed Files on Systems Without VAX RMS Journaling Installed

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.

8.32.14 Restriction on Exclusive Access to Recovery Unit Journaled Files

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.

8.32.15 SET FILE/AI_JOURNAL or SET FILE/BI_JOURNAL Command

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:

```
$ CREATE X.X
CTRL/Z
$ SET FILE X.X /BI_JOURNAL=(CREATE,FILE=X_JOURNAL)
$ SET FILE X.X /BI_JOURNAL=(CREATE,FILE=X_JOURNAL)
```

8.32.16 VFC Format Sequential Files Partially Supported for Before-Image or Recovery Unit Journaling

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.

8.32.17 Mount Utility Creates the Logical Name DISK$volume_label

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:

```
$ MOUNT/SYSTEM DBA0: FINANCE_DISK
$ MOUNT/SYSTEM DBA0: FINANCE_DISK DISK1
$ MOUNT/CLUSTER DBA0: FINANCE_DISK
$ MOUNT/CLUSTER DBA0: FINANCE_DISK DISK1
```

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:

```
$ DEFINE/EXECUTIVE_MODE/TRANSLATION_ATTRIBUTES=CONCEALED
_Log name: DISK$FINANCE_DISK
_Equ name: DBA0:
```

8.33 System Bootstrap — Handling of the Special Files

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.DMP. You can do this by entering the following commands:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CREATE SYS$COMMON:[SYSEXE]SYSDUMP.DMP/SIZE=n
SYSGEN> EXIT
$
```

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

```
$ SET FILE/ENTER=SYS$SYSDEVICE:[SYSn.SYSEXE]
```

Note: Note that DIGITAL does not recommend that you use the SET FILE /ENTER command. In addition, DIGITAL does not support shared dump files because they cannot be relied on to capture memory dumps from multiple failing systems.

8.34 Show Cluster Utility — Notes

The following sections describe problems and restrictions with the Show Cluster Utility.

8.34.1 HW_TYPE Field Changes

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 > REMOVE SYSTEMS/TYPE=V780
```

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 > REMOVE SYSTEMS/TYPE=(V780,"VAX-11/780")
```

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.

8.34.2 WRITE/ALL Method for Determining Page Size

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*.

8.35 New Field in SHOW MEMORY Display—Reservable Pages

VMS Version 5.0 contains a new field of information displayed with the DCL command SHOW MEMORY: *Reservable* (reservable pages). This new field is displayed for each paging and swap file on the system, as shown in the following sample output to the SHOW MEMORY/FILES command:

```
            System Memory Resources on 31-DEC-1988 09:45:44.04
Paging File Usage (pages):                  Free  Reservable    Total
  DISK$LATEST:[SYS2.SYSEXE]SWAPFILE.SYS      256         256     4096
  DISK$LATEST:[SYS2.SYSEXE]PAGEFILE.SYS     7613        6912     8192
```

8.35 New Field in SHOW MEMORY Display—Reservable Pages

The *Free* field in the SHOW MEMORY display indicates the number of free blocks in each paging and swap file currently installed. Free blocks are those which may be physically allocated in the file.

The *Reservable* field indicates the number of blocks in each paging and swap file currently installed which may be logically claimed by processes for future physical allocation. A negative value may indicate that the file is overcommitted and needs to be enlarged; however, the significance of a negative value depends upon the configuration of the system. For example, on a system where the following conditions exist:

- Most page files have a non-zero free page count.
- The average number of pages on the free and modified lists is non-zero.
- The modified paging writing activity is modest, a negative reservable page count is not indicative of a problem.

The *In Use* field is no longer displayed for paging and swap files currently installed on the system. The actual number of pages in use can be calculated by subtracting the total number of pages displayed from the free pages.

The *VMS DCL Dictionary* will be updated to reflect these changes in the SHOW MEMORY command in future versions of the VMS documentation.

8.36 SYSGEN — Parameter Changes

A number of new SYSGEN parameters have been added for VMS Version 5.0. In addition, the names of three SYSGEN parameters have been changed, as have the definitions of the MSCP_LOAD and MSCP_SERVE_ALL parameters.

These changes are described in the following sections.

8.36.1 SYSGEN Parameter RECNXINTERVAL

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.)

8.36.2 SYSGEN Parameter MULTIPROCESSING Default Value

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.

8.37 SET TIME/CLUSTER Command Superseded

The DCL command SET TIME/CLUSTER has been superseded by the SYSMAN command CONFIGURATION. For more information, see the *VMS SYSMAN Utility Manual*. However, for compatibility reasons, the SET TIME/CLUSTER command may still be used. In addition, please refer to Section 7.1.7 before using the SET TIME/CLUSTER command.

8.38 System Management Utility (SYSMAN) — Notes

The System Management Utility (SYSMAN) is a new utility created for Version 5.0 of the VMS operating system. See the *VMS Version 5.0 New Features Manual* for detailed information about this utility.

The following problem applies to the SYSMAN Utility for Version 5.0.

- 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.

8.39 SYS$LOADABLE_IMAGES [SYS$LDR] Directory

The SYS$LOADABLE_IMAGES directory contains various files and images that are loaded during the bootstrap of the system. Images in this directory are not executable in the conventional sense; that is, they cannot be executed with RUN or other DCL commands.

Modular Executive Files in [SYS$LDR] Directory

File Name	Image Type
ERRORLOG.EXE	Executive loaded image
EVENT_FLAGS_AND_ASTS.EXE	Executive loaded image
EXCEPTION.EXE	Executive loaded image
EXEC_INIT.EXE	Executive loaded image
IMAGE_MANAGEMENT.EXE	Executive loaded image
IO_ROUTINES.EXE	Executive loaded image
LOCKING.EXE	Executive loaded image
LOGICAL_NAMES.EXE	Executive loaded image
MESSAGE_ROUTINES.EXE	Executive loaded image
PAGE_MANAGEMENT.EXE	Executive loaded image
PRIMITIVE_IO.EXE	Executive loaded image
PROCESS_MANAGEMENT.EXE	Executive loaded image
RECOVERY_UNIT_SERVICES.EXE	Executive loaded image
RMS.EXE	Executive loaded image
SECURITY.EXE	Executive loaded image
SYS.EXE	Executive base image
SYSDEVICE.EXE	Executive loaded image
SYSGETSYI.EXE	Executive loaded image
SYSLICENSE.EXE	Executive loaded image
SYSTEM_DEBUG.EXE	Executive loaded image
SYSTEM_PRIMITIVES.EXE	Executive loaded image
SYSTEM_SYNCHRONIZATION.EXE	Executive loaded image
WORKING_SET_MANAGEMENT.EXE	Executive loaded image
VMS$SYSTEM_IMAGES.DATA	Executive loaded image data file

8.39 SYS$LOADABLE_IMAGES [SYS$LDR] Directory

Other Files in [SYS$LDR] Directory

File Name	Image Type
CLUSTRLOA.EXE	Loadable VAXcluster support code
CONINTERR.EXE	Connect-to-interrupt driver
SCSLOA.EXE	Loadable routines used by SCS
FPEMUL.EXE	Floating point emulation for F-, D- G- and H-floating point
VAXEMUL.EXE	VAX–11 instruction emulator

Processor-Specific System Image Files in [SYS$LDR] Directory

File Name	File Name
SYSLOA410.EXE	SYSLOA41D.EXE
SYSLOA41W.EXE	SYSLOA730.EXE
SYSLOA750.EXE	SYSLOA780.EXE
SYSLOA790.EXE	SYSLOA8NN.EXE
SYSLOA8SS.EXE	SYSLOAUV1.EXE
SYSLOAUV2.EXE	SYSLOAWS1.EXE
SYSLOAWS2.EXE	SYSLOAWSD.EXE

Device Driver Files in [SYS$LDR] Directory

File Name	File Name
CNDRIVER.EXE	CRDRIVER.EXE
CTDRIVER.EXE	CVDRIVER.EXE
CWDRIVER.EXE	DBDRIVER.EXE
DDDRIVER.EXE	DLDRIVER.EXE
DMDRIVER.EXE	DQDRIVER.EXE
DRDRIVER.EXE	DUDRIVER.EXE
DVDRIVER.EXE	DXDRIVER.EXE
DYDRIVER.EXE	DZDRIVER.EXE
DZVDRIVER.EXE	ESDRIVER.EXE
ETDRIVER.EXE	FBDRIVER.EXE
FYDRIVER.EXE	LCDRIVER.EXE
LIDRIVER.EXE	LPDRIVER.EXE
LTDRIVER.EXE	MBXDRIVER.EXE
NDDRIVER.EXE	NETDRIVER.EXE
NODRIVER.EXE	PADRIVER.EXE
PBDRIVER.EXE	PDDRIVER.EXE
PUDRIVER.EXE	RTTDRIVER.EXE
RXDRIVER.EXE	TFDRIVER.EXE

File Name	File Name
TMDRIVER.EXE	TSDRIVER.EXE
TTDRIVER.EXE	TUDRIVER.EXE
WPDRIVER.EXE	XDDRIVER.EXE
XEDRIVER.EXE	XGDRIVER.EXE
XMDRIVER.EXE	XQDRIVER.EXE
YCDRIVER.EXE	YEDRIVER.EXE
YFDRIVER.EXE	YIDRIVER.EXE

8.40 Tailoring Program Changed

You can tailor a VMS computer system to create an ideal system environment for your site-specific needs. Using the VMSTAILOR program, you can either add or remove system files to create this environment. The VMS tailoring program for Version 5.0 has been modified.

Dual system disks are no longer supported. As a result, all kits are distributed on packed volumes, including RL02 disks. Previously, the RL02 kit contained the REQUIRED and LIBRARY save sets on one disk, and the REQUIRED and OPTIONAL save sets on the other disk. The RL02 kit now contains the following save sets packed consecutively on the RL02 disk:

VMS050.A	For upgrade
VMS050.B	Old REQUIRED save set
VMS050.C	Old LIBRARY save set
VMS050.D	Old OPTIONAL save set

When VMS Version 5.0 is installed on your system, you can remove unwanted parts of the VMS operating system by using the tailoring program. If any parts need to be replaced after they have been removed, you can add them again by using the tailoring program along with the distribution kit. However, before you begin adding files to your system, you should make a note of how much available disk space you have.

The installation and operations guide for your processor describes how to invoke the tailoring program.

8.41 Quorum Disks — Notes

The following sections describe changes to the quorum disk environment.

8.41.1 New Configuration Support

In addition to the configurations previously supported, quorum disks are now supported in Local Area VAXcluster configurations and mixed interconnect clusters.

8.41.2 Disk Behavior

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 Guide to VAXclusters*.

8.42 UIS Workstations — Setting of Multiprocessing SYSGEN Parameter

If you have a workstation that is running UIS, do not set the SYSGEN parameter MULTIPROCESSING to the value 2. Multiprocessing is not supported for VAX workstations.

8.43 SET HOST/DTE Command Causes System Failures on Workstations

The SET HOST/DTE command does not pass flow control to the remote system. This allows it to communicate with other devices that do not understand XOFF/XON flow control. When data is received from the remote system, SET HOST/DTE buffers this data in process virtual memory and then displays on your terminal. If you request the display to stop by typing CTRL/S or by pressing the HOLD SCREEN button, the output routine stops. However, any additional input from the remote system still continues to be buffered.

Eventually SET HOST/DTE will run out of process virtual memory and abort. However, on workstations, the error message display requires some additional process virtual memory which is not available. Once this condition is detected, the UIS workstation software will perform a BUGCHECK instruction and the current image is RTPAD.EXE.

This problem cannot be avoided. The problem may be delayed or reduced in frequency by increasing process virtual memory. Additionally, not suspending output by using HOLD SCREEN and CTRL/S (XOFF) for long periods of time also reduce the frequency of this problem. The factors involved in the problem are baud of the remote port, a user's PGFLQUOTA and VIRTUALPAGECNT. Process virtual memory can be increased by increasing PGFLQUOTA in the authorization file, and if needed, increase the SYSGEN parameter VIRTUALPAGECNT by editing MODPARAMS.DAT and running AUTOGEN.

SET HOST/DTE will be changed in a future release of the VMS operating system to correct this problem.

8.44 UETP Changes

The following sections describe changes that have been made to UETP.

8.44.1 SYSUAF Quotas for SYSTEST and SYSTEST_CLIG Accounts

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

8.44.2 Support for Q-bus Devices

VMS Version 5.0 is the first release of UETP to support Q-bus devices (the MicroVAX chip set and its associated devices). UETP files are included in the OPTIONAL saveset, VMS050.D, of the VMS Version 5.0 kit.

8.44.3 Cluster Integration Test Phase

The Cluster Integration Test Phase of UETP now limits itself to a selection of three other nodes within the cluster for testing. This change was made to help decrease the overall testing time required to verify the successful addition of a new VMS node to an existing cluster.

8.44.4 Device Support Removed

Beginning with VMS Version 5.0, UETP no longer supports the following devices:

- DR11C
- DN11
- DV11

Entries for these devices have been removed from the file UETSUPDEV.DAT.

8.45 Installing a BUA Board on a VAX 8820, 8830, or 8840

If you install a BI unibus adaptor (BUA) board on a VAX 8820, 8830, or 8840, make sure you connect the cables immediately after you put the board in the slot. If you leave the board in the slot without connecting the cables, the VMS operating system will crash.

8.46 Stopping a CPU on a VAX 8530, 8550, 8700, 8800, 8820, 8830, and 8840

If you want to stop a specific CPU while the VMS operating system is running, use the DCL command STOP/CPU. The STOP/CPU command lets you stop a CPU without halting the VMS operating system. Only press CTRL/P and enter the HALT/CPU command when you want to stop the VMS operating system. For more information on the STOP/CPU command, see the *VMS DCL Dictionary*.

8.47 Unsynchronized Cluster Time Affects SUBMIT/AFTER Command

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.

8.48 VAXcluster Ethernet Adapter Restriction

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 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.

8.49 UNIBUS Devices on VAX 8800 Systems Running SMP — Known Problem

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 the VAX 8300 or VAX 8350 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.

8.50 UNIBUS Floating Interrupt Vector Change

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 which 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 the SYSBOOT breakpoint, 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:

```
$ RUN SYS$SYSTEM:SYSGEN SHOW/CONFIGURE/COMMAND_FILE
```

6 Next, use the CONFIGURE command to determine the correct vectors for all devices. Enter the following commands:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CONFIGURE
DEVICE> device-name
DEVICE> CTRL/Z
```

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.

8.51 Reduction in VAXcluster State Transition Time

The lock rebuild component of a VAXcluster state transition has been modified to either eliminate or greatly reduce the perceived effect of adding or removing nodes in many large VAXcluster configurations. In general, this modification applies to many Local Area VAXcluster and mixed-interconnect VAXcluster configurations. It may also apply to some CI-only VAXcluster configurations.

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.

8.52 Command Procedure for the VAX–11/750

Booting a VAX–11/750 from the console TU58 can be time consuming. One way to reduce the amount of time this takes is to physically place the files needed for booting after the directory on the console TU58. This minimizes the amount of time the tape has to rewind during the boot process. The following command procedure is an example of how you might do this. Before using this procedure, make a backup copy of the console TU58.

```
$ !
$ ! CONSOLE_T58_OPTIMIZER.COM
$ !
$ !*****************************************************************************
$ !*
$ !*  COPYRIGHT (c) 1988 BY
$ !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
$ !*  ALL RIGHTS RESERVED.
$ !*
$ !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
$ !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE
$ !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER
$ !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
$ !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY
$ !*  TRANSFERRED.
$ !*
$ !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE
$ !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT
$ !*  CORPORATION.
$ !*
$ !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS
$ !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
$ !*
$ !*
$ !*****************************************************************************
$ !*
$ !*  DIGITAL ASSUMES  NO  RESPONSIBILITY TO SUPPORT THE SOFTWARE DESCRIBED
$ !*  DESCRIBED IN THIS MODULE, NOR TO ANSWER INQUIRIES ABOUT IT.
$ !*
$ !*  THIS SOFTWARE MODULE IS PART OF A TEMPLATE WHICH MAY REQUIRE CUSTOMER
$ !*  MODIFICATIONS TO WORK IN ALL CIRCUMSTANCES.
$ !*
$ !*****************************************************************************
$ !
$ !  This is a sample DCL command procedure that you can modify to
$ !  create a console TU58 that is optimized to reduce tape motion.
$ !  Using an optimized console TU58 can significantly decrease the
$ !  amount of booting time for a VAX-11/750.
$ !
$ !  DIGITAL STRONGLY RECOMMENDS THAT YOU MAKE A BACKUP COPY OF YOUR
$ !  CONSOLE TU58 AND STORE THE ORIGINAL IN A SAFE PLACE.
$ !
$ !  Before running this command procedure, enter the following
$ !  command:
$ !
$ !    $ @SYS$UPDATE:CONSCOPY "" SAVE SYS$UPDATE:CONSOLE.T58 CSA1:
```

```
$ !
$ !  Now you are ready to execute this command procedure with the
$ !  following command:
$ !
$ !    $ @CONSOLE_T58_OPTIMIZER
$ !
$ !  Invoke EXCHANGE
$ !
$ If F$search("sys$update:console.t58").eqs."" then $EXIT 0
$ If F$getdvi("csa1","mnt") then $DISMOUNT CSA1:
$ EXCHANGE
!
! mount the copy of the console we just made in SYS$UPDATE
!
MOUNT/VIRTUAL DISK: SYS$UPDATE:CONSOLE.T58
INITIALIZE CSA1: /SEGMENTS=2 "750 CONSOLE"
MOUNT CSA1: /DATA_CHECK /WRITE
!
! Copy the needed boot files, optimizing their placement to reduce
! tape movement.
!
!       *** This list may not be complete for all sites. ***
!       *** Check with your system manager to see what   ***
!       *** files may be needed for your specific site.  ***
!
COPY /LOG  DISK:DEFBOO.CMD      CSA1:/START=10  ! default boot command file
COPY /LOG  DISK:BOOT58.EXE      CSA1:/START=14  ! bootstrap to read DEFBOO.CMD
COPY /LOG  DISK:VMB.EXE         CSA1:/START=138 ! VAX bootstrap
COPY /LOG  DISK:CI780.BIN       CSA1:/START=266 ! CI microcode file
COPY /LOG  DISK:PCS750.BIN      CSA1:/START=394 ! 750 microcode patch file
COPY /LOG  DISK:PCSLOD.EXE      CSA1:/START=420 ! 750 microcode loader
!
! Copy all the remaining files, their placement can not be guaranteed.
! Any frequently used files should be explicitly placed towards the
! "beginning" of the tape.
!
COPY /LOG /NODELETE DISK:*.* CSA1:
!
DIRECTORY /FULL /ALL CSA1:      ! include unused blocks and file placement data
DIRECTORY /SIZE      DISK:      ! just list file and their size
DISMOUNT CSA1:
DISMOUNT DISK:
EXIT
$ !
$ ! use the WRITEBOOT utility to set the boot block on the console TU58
$ ! to look for BOOT58.EXE
$ !
$ RUN SYS$SYSTEM:WRITEBOOT
CSA1:BOOT58.EXE
1
COOO
$ !
$ !
$ !
$ Type sys$input

Carefully review the directory of the console TU58. If EXCHANGE
detected any bad blocks, it might not have been able to place the files
where you wanted them. It might be necessary to use a different TU58
cassette or specify slightly different starting block numbers.
```

8.52 Command Procedure for the VAX–11/750

```
$ ! Since the console device is now visable to the system, prevent
$ ! unauthorized access to the console TU58.
$ !
$ If f$getdvi("CSA1","MNT") then $DISMOUNT CSA1:
$ MOUNT/FOREIGN/SYSTEM/NOWRITE CSA1:  /NOASSIST
$ !
$ EXIT
```

8.53 Undocumented New VAX Computers

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.

8.54 VAX 6210/6220 Systems — LP11 Printer not Supported when Standalone BACKUP Is Booted from a TK50

VAX 6210 and 6220 systems that are configured with both a UNIBUS and an LP11 line printer display the following message when Standalone BACKUP is booted from a TK50 tape cartridge:

```
SYSGEN%W-NOSUCHFILE, file not found
[SYSEXE]LPDRIVER.EXE
```

If you receive this message, you cannot direct the output of a BACKUP/LIST operation to an LP11 line printer. All other features of Standalone BACKUP will work normally, however.

This will be fixed in a future maintenance release of the VMS operating system.

8.55 VAX 8800 — STOP/CPU Command Fails

On VAX 8800 systems, the STOP/CPU command followed by the START /CPU command fails.

The SHOW/CPU/FULL command shows the target CPU in the INIT state initially, and then in the TIMOUT state.

The suggested solution for this problem is to avoid using the STOP/CPU command followed by the START/CPU command. This sequence is not operationally useful on a VAX 8800. Should this command sequence be issued and the secondary processor be removed from operation, the secondary processor can be restarted by performing the following operation at the console terminal:

Press CTRL/P.

Enter the following commands:

```
>>> SET CPU CURRENT_SECONDARY
>>> HALT
>>> SET CPU CURRENT_PRIMARY
>>> SET TERMINAL PROGRAM
```

8.56 Large System Bootstrap Failure

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> SHOW BALSETCNT
SYSBOOT> SET BALSETCNT new value
SYSBOOT> CONTINUE
```

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.

8.57 Color VAXstations in Clustered Environments

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 which contain VAXstation/GPX workstations with separate console terminals.

8.58 VAXstation 2000/MicroVAX 2000 Hard Disk Geometry Change

There is an inconsistency in the geometry reported for hard disks in the VAXstation 2000 and MicroVAX 2000 systems as viewed locally and as a served disk on a remote system. This inconsistency exists in the reported number of cylinders on the volume. This discrepancy has been corrected in VMS Version 5.0, but there are important consequences for disks initialized under a previous release. The following information applies only to local disks on a VAXstation 2000 or MicroVAX 2000 that were or are running versions of the VMS operating system prior to Version 5.0.

- The major effect of this change is in the restoring of any save sets created with the command BACKUP/PHYSICAL. Unless the geometry of the disk is the same, the restoration will not take place. It will be necessary to restore any save sets made using this method before initializing the disk on VMS Version 5.0 or before using the solution described in the next paragraph for disks initialized before VMS Version 5.0.
- Using the command ANALYZE/DISK/REPAIR properly resets the information that is recorded on the disk volume itself. This is the technique to use on disks initialized on VAXstation 2000 and MicroVAX 2000 systems running versions of the VMS operating system prior to VMS Version 5.0.

8.59 VAX 6200 Series Computer Systems

The VAX 6200 series computers are new multiuser systems that are fully compatible with other VAX processor software, including VMS software, optional software, and applications software. The VAX 6200 series can be used as multiprocessors that interconnect several KA62A processors. The VAX 6210 contains one KA62A processor, the VAX 6220 contains two, the VAX 6230 contains three, and the VAX 6240 contains four.

For specific information about VAX 6200 series computers, refer to the *VMS Installation and Operations: VAX 6200 Series* guide.

8.60 VAX 8820, VAX 8830, and VAX 8840 Computer Systems

The VAX 8820, VAX 8830, and VAX 8840 are air-cooled systems with battery backup that can contain up to four processors. They use symmetric multiprocessing to deliver up to 22 times the performance of a VAX–11/780.

These systems are packaged in two cabinets. The memory and I/O cabinet supports 64Mb to 512Mb of shareable memory, up to six VAXBI buses, and both an NI and a CI port. The CPU cabinet can be configured with up to four processors in a tightly coupled multiprocessor configuration. An optional BI expander cabinet provides two BI channels with space for up to six channels. In addition to the above cabinets, there is a MicroVAX II console subsystem.

These systems offer full compatibility with software written and used on other VAX computers, including VMS software, optional software products, and applications software.

For more information about these VAX computers, refer to the *VMS Installation and Operations: VAX 8820,8830,8840* guide.

8.61 Restriction on a VAXBI 5

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.

8.62 Halting a VAX 8530, 8550, 8700, 8800

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

```
>>> HALT
>>> CLEAR RESTART_FLAGS
```

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.

8.63 Fewer Cluster Messages Displayed on the Console Terminal and Sent to OPCOM

The number of messages displayed on the console terminal and sent to OPCOM has been reduced for VMS Version 5.0. Certain messages are now displayed only when a quorum is lost or a node has waited longer than two minutes while attempting to join a VAXcluster environment. Once quorum is regained or the node succeeds in joining a VAXcluster environment, the messages are inhibited.

The following messages are only displayed when quorum is lost or a node is unable to boot within the specified period:

```
%CNXMAN, Discovered system xxxxxx
%CNXMAN, Established connection to system xxxxxx
%CNXMAN, Deleting CSB for system xxxxxx
```

The following new messages are displayed upon failing to join a cluster within two minutes of booting in order to describe connections that have already been formed:

```
%CNXMAN, Have connection to system xxxxxx
%CNXMAN, Have "connection" to quorum disk
```

Messages that relate to losing and regaining connections to nodes and to joining a VAXcluster system are generally unaffected by this change. One exception is that the output of a connection broken message is inhibited when a node is removed from a VAXcluster system before the connection to it breaks. This happens during a reconfiguration when a node is being removed because it only has connections to a subset of the total members of a VAXcluster system.

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

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.

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

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:

  ```
  $ RENAME DUA0:[000000]SYS0.DIR DUA0:[000000]SYS1.DIR
  ```

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

  ```
  $ SET FILE/REMOVE DUA0:[SYSEXE]SYSBOOT.EXE;
  ```

  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:

  ```
  >>> B/10000000
  ```

- 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.)

8.66 VMS Bootstrapping Problem and Solution

There are occasions when a machine bootstrapping a VMS Version 5.0 system for the first time will halt in the booting sequence. The VAX 8500-series processors are particularly susceptible to this problem. The problem occurs because the default nonpaged pool SYSGEN parameters are too small for some configurations.

How to Recognize the Problem

Shortly after the VMS operating system version banner appears, a HALT instruction is executed with the program counter (PC) equal to 800025F1.

What to Do

Perform a conversational boot of the processor as described in the installation and operations guide for your processor. When the SYSBOOT prompt is displayed, increase the values for the SYSGEN parameters NPAGEDYN, SRPCOUNT, IRPCOUNT, and LRPCOUNT. Increase the SRPCOUNT, IRPCOUNT, and NPAGEDYN values by 25%, and increase the LRPCOUNT value from 4 to 10.

Iteratively continue this approach until your system no longer halts during booting. After you invoke AUTOGEN, the proper nonpaged pool values will be set.

8.67 Shutdown Notification on Clusters — Note

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:

```
$ DEFINE SHUTDOWN$INFORM_NODES MOE,LARRY
$ @SYS$SYSTEM:SHUTDOWN
```

In this example, only users on systems MOE and LARRY will be notified.

8.68 VMSINSTAL Option N — Compatibility Problem

Use of the VMSINSTAL Option N to display, print, or copy layered product release notes is not compatible with VMSINSTAL Option A, which uses autoanswer files to supply answers to questions output by the VMSINSTAL command procedure. With Version 5.0, Option N no longer records or reads responses in the autoanswer file. As a result, use of Option A with Option N produces the following error messages:

```
%VMSINSTAL-F-AUTOSYNC, Autoanswer file is not in synch with questions.
-VMSINSTAL-F-AUTOSYNC, question: * Do you want to purge files replaced by
                                  this installation [YES]?
-VMSINSTAL-F-AUTOSYNC, file:     * Select option [3]:
%VMSINSTAL-F-UNEXPECTED, Installation terminated due to unexpected event.
         VMSINSTAL procedure done at 14:00
```

The solutions are either not to use Option A with Option N or to re-create the autoanswer file before installing the layered product.

8.68 VMSINSTAL Option N — Compatibility Problem

Option N allows the installer to view, print, or copy the online release notes for those layered products that support online release notes.

Note: Currently, not all layered products support online release notes. Use of Option N in these cases produces no difference in the flow of the installation procedure.

8.69 VMS/MicroVMS — Notes Related to Merge

The following notes cover topics related to the merging of the VMS and MicroVMS operating systems into a single operating system. The MicroVMS operating system no longer exists as a separate entity.

8.69.1 Format and Installation Procedures

All kits now share the same format and installation procedures, regardless of what type of machine they are to be installed on (MicroVAX or VAX).

8.69.2 ERRFMT and OPCOM

The error logging process (ERRFMT) is now started on all MicroVAX systems. Previously, error logging was started only on VAX systems. OPCOM, which was also not started on MicroVAX systems, is now started on all systems except for workstations not in a cluster.

8.69.3 STARLET.OLB

The subset version of STARLET.OLB, which used to be in the MicroVMS operating system, is no longer available.

8.69.4 SYSALF.DAT

The SYSALF.DAT file, which gave the autologin function to the extra MicroVAX serial ports in the MicroVMS operating system, no longer provides any default autologin function for MicroVAX serial ports.

8.69.5 HELPLIB.HLB

The subset version of HELPLIB.HLB for the MicroVMS operating system no longer exists.

8.69.6 Tailoring

A new tailoring system has been introduced for all VAX computers. For information about the new tailoring system, refer to the installation and operations guide for your system.

8.69.7 Installing System Images

Systems that used to run the MicroVMS operating system no longer have to install system images from SYSTARTUP_V5.COM. All VMS Version 5.0 systems have the system images installed automatically, unless they are tailored off.

8.69.8 Logical Names

A new file called SYLOGICALS.COM has been put into the SYS$MANAGER directory to define logical names. If you have a MicroVAX that is not in a cluster, the logical names $DISKn, $FLOPPYn, $TERMINALn, are still defined as they were in the MicroVMS operating system.

8.69.9 Accounts

The USER and USERP accounts for the MicroVMS operating system have been eliminated. Also, all systems now have the SYSTEST account. The SYSTEM, SYSTEST, and FIELD accounts must have a password assigned to them. This is enforced on all systems by the installation procedure.

8.69.10 UVDEFAULT.PAR

There is no longer a UVDEFAULT.PAR file, which was only implemented on the MicroVMS operating system. This means that on a MicroVAX system, the SYSGEN command USE DEFAULT must be specified instead of USE UVDEFAULT.PAR. (This is used from SYSBOOT to reboot if the system has been rendered unbootable by the modification of some system parameters.)

8.69.11 VMSTAILOR.EXE and VMSKITBLD.COM

VMSTAILOR.EXE and VMSKITBLD.COM now use an indexed data file (VMSKITBLD.IDX) for reference, rather than a sequential file. VMSKITBLD.DAT still exists as a sequential version of VMSKITBLD.IDX.

8.69.12 Template File

A new type of file called the template file (extension = TEMPLATE) has been introduced. When a new system disk is built with VMSKITBLD.COM, it uses the TEMPLATE version of the file. For instance, SYSTARTUP_V5.COM is in both SYSTARTUP.TEMPLATE and SYSTARTUP_V5.COM. The TEMPLATE version should not be altered. (If you need a copy of the template, copy a TEMPLATE version to a COM version.) This ensures that all system disks built have the SYSTARTUP files and that there will always be an original version of the file on the system.

8.69.13 MicroVMS Files Included for VMS Version 5.0

SYLOGIN.COM, LOGIN.COM, EDTINI.EDT, and SYSTARTUP.COM, which used to be only on the MicroVMS operating system, are in the VMS operating system as example template files for all systems in VMS Version 5.0. Note that the command procedure SYSTARTUP.COM has been renamed to SYSTARTUP_V5.COM for Version 5.0.

The command procedures MGRMENU.COM, ADDUSER.COM, BACKUSER.COM and RESTUSER.COM, which used to be only on the MicroVMS operating system, are now included as examples and are contained in the directory SYS$EXAMPLES. Note that MGRMENU.COM is not automatically invoked when you log in to the SYSTEM account.

Note also that DECnet–VAX is no longer distributed as a separate kit for MicroVAX systems. It is distributed with the operating system.

8.70 VMS Executive — Changes

There are several changes in the VMS executive for Version 5.0. The following sections describe these changes.

8.70.1 Modified Page Writing now Multithreaded

Prior to VMS Version 5.0, the modified page writer never had more than one I/O operation outstanding at a time. For Version 5.0, the modified page writer can have up to 127 I/Os outstanding, with the default being 4. Multithreading allows significantly greater modified page writing throughput. On fast processors with large memories, multithreading can have a significant effect on overall system performance.

The maximum number of I/O threads is set by the SYSGEN parameter MPW_IOLIMIT.

8.70.2 SYSGEN Parameters Added

The following SYSGEN parameters have been added for Version 5.0:

- MPW_IOLIMIT

  This parameter sets the maximum number of outstanding, modified page writer I/O operations, or threads. Each I/O thread requires a permanent nonpaged pool allocation equal to approximately the following bytes:

  ```
  176 + (6 * MPW_WRTCLUSTER)
  ```

  Assuming the default value of 96 for MPW_WRTCLUSTER, the required allocation per I/O thread is 752 bytes.

- MPW_LOWAITLIMIT

  This parameter sets the lower modified page list threshold that controls when the modified page writer allows processes in the miscellaneous wait state (RWMPB) to proceed. Prior to Version 5.0, such processes were removed from the wait state only when the number of pages on the modified list was at or below MPW_LOLIMIT. For Version 5.0, using

this new typically higher threshold can significantly increase system performance for fast processors with large memories.

System initialization now forces the following relationships among modified page writer related SYSGEN parameters:

```
MPW_WAITLIMIT >= MPW_HILIMIT
MPW_LOWAITLIMIT <= MPW_HILIMIT
MPW_LOWAITLIMIT >= MPW_LOLIMIT
```

8.70.3 Changes to Process Paging File Assignment

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.

8.70.4 Paging File Recommendation

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.

8.70.5 PAGEFILE.SYS — Changes

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 being shipped with 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.

8.70.6 Extended Working Set Sizes

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.

8.71 System Security Information

This section includes information about VMS Version 5.0 system security.

8.71.1 Directory and File Protection — Changes

VMS Version 5.0 includes changes to enhance the protection of files and directories. Because a directory is a separately protected file, prior versions of VMS allowed situations in which it was possible to edit a file and create a new version even though the file itself was protected read-only. In VMS Version 5.0, each entry in a directory is protected by both the directory file protection, as before, and also the protection of the file to which the directory entry points.

- The creation of a new version of a file requires write access to the previously existing version of the file. This permits an environment in which multiple users are allowed write access to a public directory, allowing them to freely place new files in the directory. However, the individual files may still be protected such that only the creator of a file is allowed to update it.
- Because removing a directory entry logically deletes the file, the removal of a directory entry requires delete access to the file. You can remove a directory entry by using the DCL commands DELETE, RENAME, or SET FILE/REMOVE. Directory files are by default created with no delete permission; therefore, renaming a directory file will normally require that its protection be changed first.

  This protection applies only to primary directory entries. Alias directory entries are protected only by the directory file protection. (For more information on primary and alias directory entries, see Section 7.6.)

Because the protection changes make a directory file a specially protected object, the protection rules for directory files have also changed. In VMS Version 5.0, opening a directory file for write access requires that the accessor have READ and WRITE and CONTROL access to the file.

8.71.2 Enhanced LAT Security Auditing and Break-In Detection

VMS Version 5.0 includes enhancements to the operating system that make it easier to audit and detect potential break-ins from sources connected to the system through terminal servers, such as the LAT.

- A new field of information has been added to security auditing records: the LAT server/port name. The server/port name identifies the name of the LAT terminal server and the port to which the terminal is connected.
- Similarly, VMS break-in detection has been enhanced to identify the server/port name of LAT-connected terminals attempting break-ins to the system. The LAT server/port name is displayed with the DCL command SHOW INTRUSION, as shown in the following example:

```
Intrusion       Type       Count  Expiration   Source
   TERM_USER    SUSPECT        2  12:24:46.70  ZK44C2/LC-2-16:PIPER
```

  This sample record in the break-in database identifies the source of the break-in as user PIPER connected to port LC-2-16 on LAT server ZK44C2.

Note: This new feature is not available for use with the Accounting Utility.

This information will be documented in future revisions of the VMS documentation.

8.71.3 Protection of Security Auditing Information

Because the operator log file (SYS$MANAGER:OPERATOR.LOG) contains all of the security auditing information, it is possible to lose auditing information if the disk on which this file resides becomes full.

While a well-managed system will normally have adequate storage capacity, unexpected circumstances can cause excessive consumption of disk space. If all blocks on the disk are in use, a situation may arise where audit data could be lost. The National Computer Security Center (NCSC) has requested, as part of the evaluation of the VMS operating system, that a warning be issued whenever this condition occurs.

NCSC requires that a message be issued prior to any audit data being lost and in sufficient time to allow corrective action to be taken before all free blocks are exhausted.

To honor this requirement, DIGITAL supplies the following procedure for users who want to operate the VMS operating system as a Class C2 evaluated system. This procedure samples the available free blocks at a specified interval. The default sampling interval is every ten minutes. If the free space on the disk is less than a specified threshold, warning messages are issued to all terminals that have been enabled as operator terminals. The default threshold is 1% of the maximum available blocks.

The following command procedure, while fully functional, is provided as a guideline to be tailored to your specific requirements:

```
$ !    SYS$MANAGER:AUDIT_GUARD.COM
$ !
$ !     Procedure to protect the audit trail when the system disk is
$ !     approaching capacity.
$ !
$ ! User adjustable parameters.  If no parameters are specified on the
$ ! command line, supply the default values.
$ $ IF P1.EQS."" THEN
$ P1 = "00:10"
$ IF P2.EQS."" THEN
$ P2 = 1
$ !
$ INTERVAL = P1         ! Sample remaining disk space at 10-minute intervals
$ THRESHOLD = P2        ! Report shortage when 1% of disk blocks are free
$ !
$ ! Determine the parameters for the device on which the operator log file
$ ! is located.
$ ! $ SET PROCESS/PRIVILEGE=OPER
$ LOG_FILE = F$SEARCH("SYS$MANAGER:OPERATOR.LOG") ! For search lists
$ IF LOG_FILE.EQS."" THEN
$ GOTO NO_LOG_FILE
$ ! $ AUD_DEV = F$PARSE (LOG_FILE,,,"DEVICE","NO_CONCEAL")
$ MAX_BLOCKS = F$GETDVI(AUD_DEV,"MAXBLOCK")
$ FREE_BLOCK_LIMIT = (MAX_BLOCKS * THRESHOLD)/100
$ !
$ ! Sit in a loop, checking the amount of available free space.
$ ! $ C2_LOOP: $ REMAINING = F$GETDVI(AUD_DEV,"FREEBLOCKS")
$ IF (REMAINING .GT. FREE_BLOCK_LIMIT) THEN
$ GOTO PAUSE
$ !
$ ! If the amount of free space drops below the selected threshold, report
$ ! the condition.
$ !
$ REQUEST  "ONLY ''REMAINING' BLOCKS AVAILABLE ON AUDIT TRAIL DISK!"
$ REQUEST  "PLEASE TAKE CORRECTIVE ACTION!"
$ !
$ ! Wait before checking the free space.
$ !
$ PAUSE:
$ WAIT 'INTERVAL'       ! Wait the interval before looking again.
$ GOTO C2_LOOP          ! Time to look again
$ !
$ ! If there is no log file, report it, and then exit cleanly.
$ !
$ NO_LOG_FILE:
$ REQUEST  "NO AUDITING INFORMATION KEPT DUE TO MISSING OPERATOR LOG FILE"
$ !
$ EXIT
```

In this command procedure, INTERVAL and THRESHOLD have the following meanings:

INTERVAL— The delta time used to control the sampling rate for checking the remaining disk space and for issuing the warning message.

THRESHOLD— The value representing a percentage of free blocks remaining on the disk relative to the total available blocks. Warning messages are generated if the percentage of free blocks remaining on the disk falls below this value.

When you run the command procedure, messages are output at the sampling rate specified by the INTERVAL parameter until action has been taken to increase the number of available free blocks above the THRESHOLD value.

Once started, this procedure continues to execute until it is either stopped by a privileged user or until the system is rebooted.

To ensure that this procedure executes each time you bootstrap the system, add the following command to your site-specific startup command procedure (SYS$MANAGER:SYSTARTUP_V5.COM):

```
$ SUBMIT SYS$MANAGER:AUDIT_GUARD /NOLOG
```

8.71.4 Security Alarm ACEs—Restriction

The use of *security alarm ACEs* on system objects other than files is unsupported.

This restriction will be documented in future revisions of the VMS documentation.

8.71.5 SET ACL Command—Restriction

A *hidden ACE* is a special type of application-dependent Access Control List Entry (ACE) that cannot be created or modified with the DCL command SET ACL.

You can use the ACL editor to view hidden ACEs; however, only applications that create hidden ACEs can remove them from an Access Control List. See the *VMS Access Control List Editor Manual* for information about creating hidden ACEs with the ACL editor.

This restriction will be documented in future revisions of the VMS documentation.

8.71.6 $SETUAI System Service—Restriction

The $SETUAI (Set User Authorization Information) system service is used to modify records in the User Authorization File (UAF). $SETUAI does not support the modification of records in the UAF where the UIC of the user specified in the record matches the UIC of the calling process, unless the calling process holds the SYSPRV privilege.

Note: DIGITAL does not recommend the practice of sharing UICs on the system.

This restriction will be documented in future revisions of the VMS documentation.

8.71.7 DECnet Security

VMS supplies SYS$MANAGER:NETCONFIG.COM to perform the initial setup of your system as a DECnet node. The security configuration created by NETCONFIG.COM is relatively open, and is suitable for use in a small, friendly network. If your network is too large to trust to this extent or if you have other requirements for increased security, there are some basic steps you should take to increase the security of your network configuration.

In brief, NETCONFIG.COM sets up a single default account named DECNET which is made available for all network objects and applications. Securing your network installation involves restricting the use of the DECNET account.

(Note that the NCP commands given throughout this section operate on the permanent DECnet database. They will take effect on your running system when it is next rebooted.)

1 Change the DECNET password. NETCONFIG creates the DECNET account with a known password. Use the Authorize Utility to change the password, and note it for use in the following steps. At the same time, make sure the DECNET account is limited to NETWORK access only. (For details, see the *Authorize Utility Reference Manual.*)

2 Remove the DECNET account from the executor database. The executor specifies a default account to be used for all activity not overridden by explicit access control, proxy, or an object definition. By removing it from the executor, you ensure that network objects will by default be protected. For audit purposes it is best to set up a non-existent username.

```
NCP> DEFINE EXECUTOR NONPRIVILEGED USER NO_ACCOUNT
NCP> PURGE EXECUTOR NONPRIVILEGED PASSWORD
```

You must now selectively enable default access to various network objects, as described in the remainder of this section.

3 MAIL—Mail reception should always run under the DECNET account. You need the password you assigned to the DECNET account above.

```
NCP> DEFINE OBJECT MAIL USER DECNET
NCP> DEFINE OBJECT MAIL PASSWORD <password>
NCP> DEFINE OBJECT MAIL PROXY OUTGOING
```

4 PHONE—You may or may not choose to allow open access to PHONE. Remember that if you enable open access to PHONE anyone in the network can get a list of who is currently logged in on your system (and thus get a list of usernames to attempt login). If you want PHONE openly enabled:

```
NCP> DEFINE OBJECT PHONE USER DECNET
NCP> DEFINE OBJECT PHONE PASSWORD <password>
NCP> DEFINE OBJECT PHONE PROXY OUTGOING
```

If you want phone restricted only to users with valid proxies:

```
NCP> PURGE OBJECT PHONE USER
NCP> PURGE OBJECT PHONE PASSWORD
NCP> DEFINE OBJECT PHONE PROXY BOTH
```

If you don't want PHONE at all:

```
NCP> PURGE OBJECT PHONE USER
NCP> PURGE OBJECT PHONE PASSWORD
NCP> DEFINE OBJECT PHONE PROXY OUTGOING
```

5 NML—NML is the Network Management Listener. It allows remote access to your DECnet database. Access to NML via the DECNET account allows anyone in the network to inspect your network parameters. Open NML access:

```
NCP> DEFINE OBJECT NML USER DECNET
NCP> DEFINE OBJECT NML PASSWORD <password>
NCP> DEFINE OBJECT NML PROXY BOTH
```

If you want NML restricted only to users with valid proxies:

```
NCP> PURGE OBJECT NML USER
NCP> PURGE OBJECT NML PASSWORD
NCP> DEFINE OBJECT NML PROXY BOTH
```

6 FAL—FAL is the remote file access facility. Open FAL access allows general access to your files to anyone on the network. It also allows anyone to create files in your DECNET directory. While a carefully managed system can safely offer open access to selected files, closing off general FAL access helps to prevent outsiders from discovering your weaknesses. Open FAL access:

```
NCP> DEFINE OBJECT FAL USER DECNET
NCP> DEFINE OBJECT FAL PASSWORD <password>
NCP> DEFINE OBJECT FAL PROXY BOTH
```

Access to FAL only by way of valid proxies:

```
NCP> PURGE OBJECT FAL USER
NCP> PURGE OBJECT FAL PASSWORD
NCP> DEFINE OBJECT FAL PROXY BOTH
```

7 TASK—The TASK object allows arbitrary command files to be executed on your system. With careful use of separate accounts, safeguards in the login command files, and so forth, it is possible to set up an environment which allows access to a controlled set of tasks. However, there are many possible loopholes which make setting up a truly restricted TASK environment very difficult; therefore in general open TASK access should be disabled, as shown in the following example:

```
NCP> PURGE OBJECT TASK USER
NCP> PURGE OBJECT TASK PASSWORD
NCP> DEFINE OBJECT TASK PROXY BOTH
```

This note describes some basic measures for securing your network access. For more details on network operations, please refer to the *VMS Networking Manual*. For more comprehensive network security measures, see Chapter 7 of the *Guide to VMS System Security*.

9 Programmer Release Notes

This chapter includes information about VMS Version 5.0 that is of interest to both the application and system programmer.

9.1 VAX Ada Version 1.5 Run-Time Library — Notes

The following sections provide information related to VAX Ada Version 1.5.

9.1.1 Change in Use of Standard Input and Output Files

In previous versions, the files STANDARD_INPUT and STANDARD_OUTPUT would both be opened when one or the other was used, or when the default input or output files (CURRENT_INPUT and CURRENT_OUTPUT) were referenced. For applications that did not make use of either or both standard files, this may have been an inconvenience. For example, if the standard output file (usually SYS$OUTPUT) could not be opened, an error message was output, even if that file was never used in the program.

This behavior has been changed so that the standard input and output files are opened individually, and then only when they are actually referenced. For example, in the following program the standard output file is not opened:

```
with TEXT_IO;
procedure TEST is
F: TEXT_IO.FILE_TYPE;
begin
    TEXT_IO.CREATE (F, "F.DAT");
    TEXT_IO.SET_OUTPUT (F);
    TEXT_IO.PUT_LINE ("Hello");
end;
```

9.1.2 Closing and Re-opening Default Files

In previous versions, if an Ada program established a new default file with the SET_INPUT or SET_OUTPUT procedures, closed the file, then re-opened it, the default file would not automatically be re-opened. VAX Ada will now re-open the default file in this case. For example, in the following program:

```
with TEXT_IO;
procedure TEST is
F: TEXT_IO.FILE_TYPE;
begin
    TEXT_IO.CREATE (F, "F.DAT");
    TEXT_IO.SET_OUTPUT (F);
    TEXT_IO.CLOSE (F);
    TEXT_IO.CREATE (F, "F2.DAT");
    TEXT_IO.PUT_LINE ("Hello");
end;
```

The text "Hello" will be written to the file F2.DAT.

Furthermore, an attempt to re-open a default file with a different mode will raise MODE_ERROR. This change is required to conform with the Ada language standard interpretation AI-00048.

9.1.3 Temporary Files Have No Name

In previous versions, temporary files (created with a null name string argument to the CREATE procedure) were created as named files with names generated by the Ada Run-Time Library. These files remained available until the program exited. As of VMS Version 5.0, temporary files have no name, and are created as temporary-marked-for-delete files, and are thus deleted when the file is closed.

Although the temporary files have no filename, they are created using the file specification "SYS$SCRATCH:". The logical name SYS$SCRATCH is defined by default to be the same as SYS$LOGIN, and thus the temporary files will be created on your login device. If you wish to redirect the temporary files to another device (for example, if disk quotas are enabled on your login device), redefine the logical name SYS$SCRATCH to name a different device. Note that since temporary files are not entered in a directory, they cannot inherit the file ownership of any directory.

As an additional consequence of this change, the NAME function will now raise USE_ERROR for a temporary file. This change is permitted by the Ada language standard interpretation AI-00046.

9.1.4 Asynchronous File Operations Supported

VAX Ada now performs the file operations OPEN, CREATE and CLOSE asynchronously. This means that a task that performs a file operation will no longer block other eligible tasks from running while waiting for the operation to complete. Note however that file and record operations on process-permanent files, such as SYS$INPUT and SYS$OUTPUT, are forced to be synchronous by RMS. See the *VAX Ada Programmer's Run-Time Reference Manual* for more information about input-output and tasking.

9.1.5 VAX Ada Restrictions

The following restrictions apply to the VAX Ada layered product for VMS Version 5.0:

- 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 that disabling the "sequential only" mode incurs a performance penalty on all network file access.

- The implementation of the CALENDAR."-" function that subtracts two values of type TIME giving a result of type DURATION is too restrictive, raising the TIME_ERROR exception if the result falls outside the range of type DAY_DURATION rather than DURATION. This problem will be corrected in a future version of the VMS operating system.
- The INTEGER_IO.PUT procedure incorrectly formats the value zero when a BASE other than 10 is specified. Rather than using the proper syntax for the specified base, it always uses the syntax for base 10. This problem will be corrected in a future version of the VMS operating system.

9.1.6 VAX BASIC RTL — Changes

The following VAX BASIC Run-Time Library power routines (raising a number to a power) have been recoded to remove the EMODF, EMODD, EMODG, or EMODH MACRO instruction:

- BAS$POWDD
- BAS$POWDR
- BAS$POWGG
- BAS$POWHH
- BAS$POWRD
- BAS$POWRD

The accuracy of the routines is about the same, differing by no more than 1 LSB (least significant bit) in most cases, or significantly better in other cases. The performance of the new routines on a VAX–11/785, VAX 8800, and a MicroVAX II are better by 1 to 11 percent, while it is about 3 percent slower on a VAX 8650.

9.1.7 DIBOL Routines Added

There are additional DIBOL Run-Time Library Routines for VMS Version 5.0. The following DIBOL Version 4.0 modules have been included with this release:

- SYS$SYSTEM:DBLMSGMGR.EXE
- SYS$LIBRARY:DBLRTL.EXE
- SYS$MANAGER:DBLSTRTUP.COM
- SYS$MESSAGE:DBLRTLMSG.EXE

Please note that DBLSTRTUP.COM should be used to start the DIBOL Message Manager. Also, the installation of any earlier version of DIBOL as a layered product supersedes the installation of the files listed above.

DIBOL Version 4.0 will be available as a separate layered product.

9.2 ALL-IN-1 Version 2.2 — Restrictions

With VMS Version 5.0 it is necessary to use Message Router VMSmail Gateway Kit Version 3.0 because Message Router Version 2.1 is not supported. When you run ALL-IN-1 Version 2.2 with Message Router VMSmail Gateway Kit Version 3.0 and VMS Version 5.0, the following restrictions apply:

- 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:

  ```
  $  INSTALL REPLACE MAIL /PRIVILEGES=SYSPRV
  ```

- 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 create 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

- If you are running the Directory Service part of Message Router Version 3.0 on VMS Version 5.0, the following error messages may occur:

  ```
  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. These error messages display 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.

- As stated in the Message Router Version 3.0 Release Notes, the verification procedures need DECnet and the Queue Manager to be started on the system. However, the verification procedures also need at least one queue to be defined in the system startup procedure.

If you do not have a queue defined in the system startup procedure, use the DCL command, INITIALIZE /QUEUE /BATCH, to define a queue in the system startup procedure. Refer to the *VMS DCL Dictionary* for details of this command.

If you are running ALL-IN-1 and you want to use VMS Version 5.0 without incurring the effects of the Message Router, DIGITAL recommends that you wait until the next version of ALL-IN-1 is released. The next version of ALL-IN-1 will contain Message Router Version 3.0.

If you do not want to upgrade to ALL-IN-1 Version 2.3, but you do want to install VMS Version 5.0, then Message Router Version 3.0 is required and must be installed before upgrading to VMS Version 5.0. However you should note the following additional restriction which will be fixed in a future version of ALL-IN-1:

- If you try to transfer your unread VMSmail into ALL-IN-1 Electronic Messaging by using the VMSmail Import option, an access violation occurs.

  If you want to read your VMSmail from your ALL-IN-1 account, make sure Message Router Version 3.0 is installed and then set Autoforward on your VMSmail account. VMSmail will automatically be imported without causing problems.

9.3 VAX BASIC Version 3.2 Installation Note

To install VAX BASIC Version 3.2 successfully, you cannot build BASIC$STARLET.TLB. To do this, answer "NO" to the following VMSINSTAL question:

```
Do you want to install the VAX/VMS system definitions?:
```

9.4 Debugger — Notes

The following sections provide information about the Debugger for VMS Version 5.0.

9.4.1 Debugging SMG Programs — Restriction

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*.

9.4.2 Linking Shareable Images for Debugging — Problem

When linking shareable images for debugging, use the DCL command LINK /DEBUG to obtain full symbolic information for those images. Do not use LINK/TRACE to obtain only the traceback symbolic information. If you use LINK/TRACE, the proper fix up records in the debug symbol table (DST) are not generated. As a result, that symbolic information may not be available.

9.4.3 MACRO Support — Change from Previous Versions

For Version 5.0, debugger support for MACRO has been enhanced to match that for the other languages supported by the debugger.

Prior to Version 5.0, the debugger did not display the source lines of a MACRO program being debugged. In screen mode, the predefined instruction display INST was the default "source" display for MACRO (INST shows the decoded instruction stream rather than the actual source code). Now the debugger displays MACRO source lines, whether you are debugging in line mode or in screen mode. In screen mode, the predefined source display SRC shows MACRO source code for MACRO programs.

Prior to Version 5.0, when the debugger language was set to MACRO, the debugger interpreted an address expression used in a language expression as the *address* denoted by the address expression. Now the debugger interprets an address expression used in a language expression as the current *value* stored at that address.

Prior to Version 5.0, when the debugger language was set to MACRO, the STEP command stepped one instruction at a time, by default (STEP /INSTRUCTION). Now the STEP command steps one source line at a time, by default (STEP/LINE).

9.4.4 Making Images Shareable for Better Performance

If there are many users who invoke the debugger on your system, your system manager can improve the use of system resources by installing the debugger and some related images as known shareable images. This is explained in more detail in Section 8.11 of this manual.

9.4.5 Obsolete Commands

The following debugger commands and command qualifiers are obsolete starting with VMS Version 5.0 and are no longer documented. For compatibility with previous VMS operating system versions, these commands and qualifiers will be supported indefinitely except as indicated.

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 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.

9.4.6 Screen-Mode Note — Change from Previous Versions

This note describes how very long lines of debugger output are presented in the OUTPUT display or in a DO display. An example of this is the output you would get when you issue the EXAMINE command to examine long ASCII strings.

Before Version 5.0, the debugger wrapped text in the OUTPUT display or in a DO display if the text was longer than the width of the terminal, regardless of the width of the display window.

The debugger now wraps text in these displays only if it exceeds 255 characters. If there is any text beyond the right edge of the display window, a diamond-shaped character (or a question mark for terminals that do not support the diamond character) appears at the right edge. This indication is analogous to the behavior of a text editor.

To see the hidden text, issue the SCROLL/RIGHT command (or press keypad-key 6 repeatedly, as needed).

9.4.7 SET IMAGE Command — Problem Corrected

Before Version 5.0, when you entered the SET IMAGE command specifying a list of images to be set, only the last named image was set. For example, after the following command line was entered, only image C was set:

```
DBG> SET IMAGE A,B,C
```

Therefore, to set several images, you had to enter a separate SET IMAGE command for each image. This problem has been corrected. You can now set a list of images with a single SET IMAGE command. As before, the current image is the last image set or the last image named in a list of images to be set.

9.4.8 SET SCOPE Command — Problem Corrected

In other versions of the VMS operating system prior to Version 5.0, the following situation existed. Before issuing a SET SCOPE command you needed to ensure that the module containing the path-name elements named in the SET SCOPE command had first been set (either dynamically by the debugger, or by issuing a SET MODULE command). For example, before issuing the following command, you had to make sure that module MODK was set:

```
DBG> SET SCOPE MODK\ROUTD
```

This procedure is no longer a requirement. In VMS Version 5.0, when you issue a SET SCOPE command, the debugger sets the module automatically if it has not already been set.

9.4.9 Using the Debugger on a VAXstation — Problem

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 operating system.

9.4.10 VAXstation Support — Change from Previous Versions

Prior to Version 5.0, when you invoked the debugger on a MicroVAX workstation, the debugger appeared in an emulated-terminal window separate from the window where you invoked the debugger. The debugger window contained debugger input and output, while program input and output appeared in the original window. This was equivalent to assigning the logical names DBG$INPUT and DBG$OUTPUT to the debugger window.

Now a separate window is not created automatically when you invoke the debugger. The new debugger command SET MODE [NO]SEPARATE enables you to create or delete a separate window for debugger input and output. The default setting is SET MODE NOSEPARATE.

Also, prior to Version 5.0, the debugger window was automatically popped over other windows and attached to the keyboard whenever the debugger prompted for input. Now, the SET PROMPT command qualifiers /POP and /NOPOP enable you to control this behavior. By default (SET PROMPT /NOPOP), the debugger window is not popped and is not attached to the keyboard when the debugger prompts for input.

The SET MODE [NO]SEPARATE and SET PROMPT/[NO]POP commands have no effect on VT-series terminals.

9.4.11 VMS Version 4.6 and Version 4.7 Debugger Notes

The following sections provide additional imformation to changes in the Debugger during the release of VMS Version 4.6 and Version 4.7. See the *VMS Debugger Manual* for more information.

9.4.11.1 CALL Command — Restriction Removed

Prior to Version 4.6 you could not enter the CALL command directly after an exception breakpoint was triggered. This restriction was removed for Version 4.6.

If a routine is called with the CALL command just after an exception breakpoint is triggered, no breakpoints, tracepoints, or watchpoints set within that routine are triggered. However, they are triggered if the CALL command is given at another time.

SET WATCH Command — Restriction Removed

Prior to Version 4.6 the SET WATCH command set watchpoints only on static variables. (A *static* variable is associated with the same virtual memory location throughout program execution.) This restriction was removed for Version 4.6.

You can now use the SET WATCH command to set watchpoints on nonstatic as well as static variables. (A *nonstatic* variable is allocated dynamically, either on the stack or in a register, and exists only when its defining routine is active on the call stack.)

However, before you can set a watchpoint on a nonstatic variable, its defining routine must be active (on the call stack). Otherwise, the variable is not defined, and the debugger issues a diagnostic message.

STEP Command — Restriction Removed

Prior to Version 4.6, you could not issue the STEP command directly after an exception breakpoint was triggered. This restriction was removed for Version 4.6.

After an exception breakpoint is triggered, you can issue the STEP command to step to the start of the exception handler that is to handle the exception. If you have not declared a handler for that exception, the exception is resignalled, and the debugger prompt is displayed; that is, the STEP command has no effect.

9.5 Linking Workstation Applications on Nonworkstation Systems

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.

9.6 Message Router Version 3.0 Installation Notes

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:

```
$  INSTALL REPLACE MAIL /PRIVILEGES=SYSPRV
```

- 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.

9.7 VAX Pascal Version 3.7 Installation Note

To install VAX Pascal Version 3.7, you must first attempt to install PASSTR034 and let it fail. Once the PASSTR034 installation fails, enter the following commands:

```
$ DEFINE SDLPASCAL SYS$COMMON:[SYSUPD.PASSTR034]SDLPASCAL.EXE
$ DEASSIGN SDLPASCAL
$ SET PROCESS/PRIVILEGES=ALL
$ SET DEFAULT SYS$COMMON:[SYSLIB]
$ PASCAL/NOWARNING/ENVIRONMENT/NOOBJECT STARLET.PAS
```

9.8 Record Management Services — Notes

The following changes have been included for Record Management Services (RMS):

9.8.1 Extended Asynchronous Interface

RMS now supports asynchronous file level operations. The file option bit FAB$V_ASY is used to request this new feature. In addition, the $DISCONNECT service now supports asynchronous file level operations when requested by the RAB$V_ASY qualifier.

9.8.2 Block Mode Support for RMS Copy Operations

Network copy operations between RMS–11 based systems will now be done in block mode rather than record mode. This increases the speed of the operations and allows files with large record sizes to be copied.

9.8.3 RMS Transparent Task-to-Task Network Operations

The buffer size for RMS transparent task-to-task network operations is now controlled by the NETWORK_BLOCK_COUNT item code (SET RMS /NETWORK_BLOCK_COUNT or the RMS_DFNBC SYSGEN parameter). The minimum buffer size remains 4096 bytes (NETWORK_BLOCK_ COUNT = 8).

9.8.4 XAB$V_NUL Option — Clarification

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.

9.8.5 Appending to Shared Sequential Files

The use of the RMS Deferred Write option with sequential files opened for shared append access could result in file corruption. To avoid corruption, the Deferred Write option is automatically disabled for shared sequential files. Use of Deferred Write with shared append access to sequential files is expected to be reenabled in a future release of the VMS operating system.

9.8.6 $FREE Restriction

The $FREE service, which releases all RMS record locks held by the specified stream, does not return the RMS$_RNL (record not locked) warning status if there were no records locked by the specified stream but instead returns the RMS$_NORMAL success status.

This restriction will be removed in a future release of the VMS operating system.

9.9 Run-Time Library — Notes

The following sections pertain to the Run-Time Library.

9.9.1 Restriction on Using PPL$ENABLE_EVENT_SIGNAL

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.

9.9.2 Omission from the PPL$CREATE_SHARED_MEMORY Routine Description

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

9.9.3 Creating Multiple Copies of a Program Through PPL$SPAWN

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.

9.9.4 Corrections to the LIB$ADAWI Routine Description

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.

9.9.5 Omissions from the LIB$SPAWN Routine Description

The *VMS RTL LIB$ (Library) Manual* omits the last argument to LIB$SPAWN. 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 will use 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, these privileges are not available to the subprocess until a SET PROCESS /PRIVILEGE command or equivalent SYS$SETPRV call is performed in the subprocess to enable these 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.

The **cli** argument must be specified in all uppercase characters.

9.9.6 Correction to LIB$CREATE_VM_ZONE

In the LIB$CREATE_VM_ZONE routine description, the argument *number-of-areas* is listed; however, *number-of-areas* does not exist. LIB$CREATE_VM_ZONE has thirteen, not fourteen, arguments.

9.9.7 String Procedures Performance Improvements

All virtual memory for dynamic strings is now allocated from a new Run-Time Library private virtual memory zone called the String Zone, which has the attribute values and benefits listed in the following chart.

Attribute	Value
Algorithm	Quick Fit
Number of Lookaside Lists	17 (short strings from 8 to 136 bytes)
Area Initial Size	4 pages
Area Extension Size	32 pages
Block Size	8 bytes
Alignment	Quadword boundary
Smallest Block Size	16 bytes (includes boundary tags)
Boundary Tags	Boundary tags are used for long strings
Page Limit	No page limit
Fill On Allocate	No fill on allocate
Fill On Free	No fill on free

These improvements result in the following benefits:

- More efficient virtual memory use. The previous initial size and extension size was 128 pages.
- Allocation and deallocation for long strings (more than 136 bytes) is twice as fast as in previous versions.
- Elimination of paging contention with the Default Zone by isolation of the string virtual memory accesses to a separate zone. A direct side effect of this change is that corruptions caused by writing into previously freed strings no longer affects items allocated in the Default Zone, directly easing the debugging effort for such problems.

Users are reminded that the Run-Time Library string manipulation procedures are the only procedures that should be used to define or alter the length or address fields of dynamic string descriptors.

9.9.8 LIB$ Routine Changes

The following LIB$ Run-Time Library routines have been rewritten:

- LIB$POLYF, LIB$POLYD, LIB$POLYG, LIB$POLYH
- LIB$EMODF, LIB$EMODD, LIB$EMODG, LIB$EMODH

The new routines no longer call upon the MACRO instructions that they were named after. The new LIB$POLYx routines should not differ from the old LIB$POLYx routines (that called the MACRO POLYx instruction) by more than 1 LSB (least significant bit). The performance of the new routines is slightly faster (about 6%) on a MicroVAX II, while slightly slower on a VAX–11/785, VAX 8650, or an VAX 8800 (less than 8%).

The new LIB$EMODx routines are more accurate. However, performance may be slower because the new routines run two to three times slower.

Note that the new LIB$EMODx routines no longer return a fractional part equal to +/-1.

9.9.9 LIB$ Translation Tables Added

The following LIB$ translation tables have been added to VMS Version 5.0.

Routine Name	Routine Function
LIB$AB_LOWERCASE	ASCII Upper Case to Lower Case Translation Table
LIB$AB_ASC_EBC_REV	ASCII to EBCDIC Reversible Translation Table
LIB$AB_EBC_ASC_REV	EBCDIC to ASCII Reversible Translation Table
LIB$AB_CVT_O_U	Trailing Overpunch Numeric to Unsigned Numeric Table
LIB$AB_CVTPT_O	Packed Decimal to Trailing Overpunch Numeric Table
LIB$AB_CVTPT_U	Packed Decimal to Unsigned Trailing Numeric Table
LIB$AB_CVTPT_Z	Packed Decimal to Zoned Translation Table
LIB$AB_CVTTP_O	Trailing Overpunch Numeric to Packed Decimal Table
LIB$AB_CVTTP_U	Unsigned Numeric to Packed Decimal Translation Table
LIB$AB_CVTTP_Z	Zoned to Packed Decimal Translation Table
LIB$AB_CVT_U_O	Unsigned Numeric to Trailing Overpunch Table

9.9.10 Significant LIB$ Updates

The following Run-Time Library routines have been updated:

- LIB$AB_UPCASE—Now covers the full DEC Multinational Character set.
- LIB$FREE_VM—Changed to return a failure status when returning unaligned blocks.
- User-provided page allocation and deallocation routines can be specified to LIB$CREATE_VM_ZONE.
- LIB$GET_VM and LIB$FREE_VM—Enhanced for use in a multi CPU environment.
- LIB$RENAME_FILE—No longer uses up all available I/O channels on network renames.

9.9.11 RTL Routine LIB$SYS_TRNLOG — Notes

The Invoke $TRNLOG System Service to Translate Logical Name routine is still supported in VMS Version 5.0, but was removed from the RTL documentation.

This 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. See the (vms_obsolete) for a description of $TRNLOG.

9.9.12 MTH$ Math Routines — Changes

The following MTH$ math Run-Time Library (RTL) routines have been rewritten to remove the following VAX MACRO instructions:

- POLYF, POLYD, POLYG, POLYH
- EMODF, EMODD, EMODG, EMODH

The MTH$ routines that were rewritten to remove the POLYx and/or EMODx instructions are as follows:

- MTH$AINT, MTH$DINT, MTH$GINT, MTH$HINT
- MTH$ANINT, MTH$DNINT, MTH$GNINT, MTH$HNINT
- MTH$ALOG, MTH$DLOG, MTH$GLOG, MTH$HLOG
- MTH$ALOG2, MTH$DLOG2, MTH$GLOG2, MTH$HLOG2
- MTH$ALOG10, MTH$DLOG10, MTH$GLOG10, MTH$HLOG10
- MTH$ATAN, MTH$DATAN, MTH$GATAN, MTH$HATAN
- MTH$ATANH, MTH$DATANH, MTH$GATANH, MTH$HATANH
- MTH$COSH, MTH$DCOSH, MTH$GCOSH, MTH$HCOSH
- MTHEXP, MTHDEXP, MTH$GEXP, MTH$HEXP, UVXEXP
- MTH$FLOOR, MTH$DFLOOR, MTH$GFLOOR, MTH$HFLOOR
- MTHSIN, MTHDSIN, MTH$GSIN, MTH$HSIN, MTH$SINCOS, MTH$DSINCOS, MTH$GSINCOS, MTH$HSINCOS
- UVXSIN, UVXGSIN, UVXSINCOS, UVXGSINCO
- MTHCOS, MTHDCOS, MTH$GCOS, MTH$HCOS, MTH$SINCOS, MTH$DSINCOS, MTH$GSINCO, MTH$HSINCOS
- UVXCOS, UVXGCOS, UVXSINCOS, UVXGSINCO
- MTH$SINH, MTH$DSINH, MTH$GSINH, MTH$HSINH
- MTH$DTANH, MTH$GTANH — (MTH$TANH and MTH$HTANH did not need to change).

The new routines should not differ from the old routines by more than one LSB (least significant bit). The performance will vary from about 10% slower to 10% faster depending upon the hardware used. The test systems that were used for this were a VAX–11/785, VAX 8650, VAX 8800, and a MicroVAX II.

9.9.13 OTS$ Routines — Changes

The following OTS$ Run-Time Library power routines (raising a number to a power) have been recoded to remove the EMODF, EMODD, EMODG, or EMODH MACRO instruction:

- OTS$POWDD
- OTS$POWGG, UVXPOWGG
- OTS$POWHH
- OTS$POWRR, UVXPOWRR

The accuracy of the routines is about the same, differing by no more than 1 LSB (least significant bit). The performance of the new routines is better; performance tests were run on a VAX–11/785, VAX 8650, VAX 8800, and a MicroVAX II.

9.9.14 SMG$ — Enhancements

The following SMG$ features have been enhanced for Version 5.0:

Feature Name	Function
Viewports	Allows a portion of the virtual display contents to be visible on the pasteboard.
Menus	Allows easy use of simple menus.
Subprocesses	Allows images to be run in a subprocess with output in a virtual display.
User renditions	Allows users to define and use their own renditions.
Hardcopy support	Allows use of line drawing characters, renditions, and wide characters if terminal is a hardcopy device.
Hardware scrolling	Multiple line hardware scrolling is now used for screen updates.
Large virtual displays	Virtual displays larger than 65535 characters are now supported.
Display controls	All non-printing characters are now represented in a virtual display.
Truncation icon	SMGPUT_CHARS, SMGINSERT_LINE, SMG$INSERT_CHARS now support truncation icon.
Invisible rendition	All routines now support invisible rendition .

9.9.15 Using SMG$CREATE_PASTEBOARD and SMG$CREATE_VIRTUAL_KEYBOARD — Restriction

If a program calls both SMG$CREATE_PASTEBOARD and SMG$CREATE_VIRTUAL_KEYBOARD, make sure SMG$CREATE_PASTEBORD is called

first. The program will not function correctly if SMG$CREATE_VIRTUAL_KEYBOARD is called before SMG$CREATE_PASTEBOARD.

9.9.16 Additional SMG$ Features

The following SMG$ features have been added for Version 5.0.

- Terminal type parameter added to SMG$CREATE_PASTEBOARD
- Buffer size parameter added to SMG$CONTROL_MODE
- Row and column parameters added to SMG$FIND_CURSOR_DISPLAY
- Additional background colors added to SMG$CHANGE_PBD_CHARACTERISTICS
- Erase line option added to SMG$PUT_CHARS
- Word wrapping option added to SMG$PUT_LINE
- Rendition parameter added to SMG$READ_FROM_DISPLAY
- Row and column parameters to SMG$GET_DISPLAY_ATTR now optional
- Block border option added to SMG$CREATE_VIRTUAL_DISPLAY
- SMG$READ_FROM_DISPLAY now returns printable characters for line drawing characters
- Additional information is now returned by SMG$GET_DISPLAY_ATTR
- Key table parameter to SMG$READ_COMPOSED_LINE is now optional

9.9.17 SMG$ Internal Changes

The following internal changes have been made to SMG$ for Version 5.0:

- Improved performance for SMG$REPAINT_SCREEN.
- All dynamic memory used by SMG$ is now in its own VM zone.
- Maximum number of pasteboards has been raised from 16 to 28.
- One event flag is now used for all virtual keyboards.
- One event flag is now used for all pasteboards that do not use broadcast trapping/unsolicited input.
- Terminals that have column/row cursor addressing are now supported.

9.9.18 Obsolete SMG$ Routines

The following SMG$ routines are obsolete for Version 5.0:

- SMG$ALLOW_ESCAPE
- SMG$PUT_WITH_SCROLL
- SMG$PUT_VIRTUAL_DISPLAY_ENCODED

9.9.19 RTL Language Support Enhancements

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.
- Miscellaneous bug fixes in the BASIC, COBOL, FORTRAN, and Pascal language RTL's that are described in each language's release notes.

Note: Note that the preceding changes could require a new version of the language to take advantage of the new features.

9.10 NCS Diacriticals Collate in Opposite Order

The languages (that is, Multinational, Dutch, English, and so forth) in the NCS default library SYS$COMMON:[SYSLIB]NCS$LIBRARY.NLB have the diacriticals collating in the wrong order.

To correct the library, please do the following:

```
$  NCS/EXTRACT=each_language_collating_sequence /OUTPUT=file_name
$  EDIT file_name
```

When you edit the file, change the diacriticals to the correct order in the second pass only as follows. (Note that you should do this for each language. The old and new formats are displayed.)

```
!OLD VERSION

DANISH = CS(
    SEQUENCE = (%X00-"N", "", "O"-"Z", "", "", "", "["-"`", "{"-"", %XD0,
          %XDE, %XF0, %XFE-%XFF),
    MODIFICATIONS=("a"-"z" = "A"-"Z", ""-"" = "A", "" = "", "" = "C",
          ""-"" = "E", ""-"" = "I", ""-"" = "O", "" = "", ""-"" = "U",
          ""-"" = "Y", ""-"" = "A", ""-"" = ""-"", "" = "", "" = "C",
          ""-"" = "E", ""-"" = "I", "" = "", ""-"" = "O", "" = "",
          "" = "", ""-"" = "U", ""-"" = "Y", "" = "OE", "" = "SS",
          "" = "", "" < %X00))
**  + CS( SEQUENCE = (%X00-"A", ""-"", "B"-"C", "", "D"-"E", ""-"",
**        "F"-"I", ""-"", "J"-"N", "", "", "O", ""-"", "P"-"R", "",
**        "S"-"U", ""-"", "V"-"Y", "", "Z", "", "", "", "["-"`", "{"-"",
          %XD0, %XDE, %XF0, %XFE-%XFF),
    MODIFICATIONS=("a"-"z" = "A"-"Z", ""-"" = ""-"", ""-"" = ""-""))
    + REVERSE(_NATIVE);
```

9.10 NCS Diacriticals Collate in Opposite Order

```
! NEW VERSION

DANISH = CS(
    SEQUENCE = (%X00-"N", "", "O"-"Z", "", "", "", "["-"`", "{"-"", %XD0,
          %XDE, %XF0, %XFE-%XFF),
    MODIFICATIONS=("a"-"z" = "A"-"Z", ""-"" = "A", "" = "", "" = "C",
          ""-"" = "E", ""-"" = "I", ""-"" = "O", "" = "", ""-"" = "U",
          ""-"" = "Y", ""-"" = "A", ""-"" = ""-"", "" = "", "" = "C",
          ""-"" = "E", ""-"" = "I", "" = "", ""-"" = "O", "" = "",
          "" = "", ""-"" = "U", ""-"" = "Y", "" = "OE", "" = "SS",
          "" = "", "" < %X00))
    + CS( SEQUENCE = (%X00-"A", ""-"", "B"-"C", "", "D"-"E", ""-"",
          "F"-"I", ""-"", "J"-"N", "", "", "O", ""-"", "P"-"R", "",
          "S"-"U", ""-"", "V"-"Y", "", "Z", "", "", "", "["-"`", "{"-"",
          %XD0, %XDE, %XF0, %XFE-%XFF),
    MODIFICATIONS=("a"-"z" = "A"-"Z", ""-"" = ""-"", ""-"" = ""-""))
    + REVERSE(_NATIVE);
```

When you complete editing the file, enter the following command:

```
$ NCS/REPLACE file_name
```

9.11 VAX C Run-Time Library — Changes

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.

9.12 VAX MACRO — Notes

The following sections describe the restrictions, fixed problems, and known problems for VAX MACRO Version 5.0.

9.12.1 Restrictions

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)

9.12.2 Fixed Problems

The following problem is fixed in VAX MACRO Version 5.0:

- Prior to VAX MACRO Version 5.0, diagnostic messages did not point to the first character of a source record (using the "!" character) when appropriate.

9.12.3 Known Problems

The following is a list of all known problems in VAX MACRO Version 5.0:

1. 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.
2. 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.
3. 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.
4. 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.
5. Assembly errors may occur if the string ".REPT" is found in comments included within a .REPT loop.
6. The %EXTRACT macro string operator is parsed by the assembler even if it is found in an unsatisfied conditional block or in a comment.
7. 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.

8 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*.

9 For branch instructions located in the body of a macro, the assembler does not always generate a diagnostic for a branch out of range.

10 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".

11 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.

12 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.

13 The %LENGTH operator does not work within .REPEAT blocks.

14 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.

15 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.

16 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.

17 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.

18 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.

19 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).

20 Attempting to assemble a VAX MACRO program using a command line in excess of 512 characters results in a corrupt object file.

21 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.)

9.13 VAXTPU — Notes

The following sections describe changes, restrictions, and known problems for VAXTPU.

9.13.1 Incompatibilities with Previous Versions of VAXTPU

The new version of VAXTPU contains the following incompatibilities with previous versions of VAXTPU:

1 The format of section files has been changed. This change means that all section files created under this version will remain compatible with subsequent releases of VAXTPU. However, existing section files created with previous versions of VAXTPU are incompatible with this version. To create a section file compatible with this version, you may need to take one or more of the following steps:

- Recompile command files
- Rewrite and compile procedures that were added to a section file by using EXTEND TPU instead of by compiling a command file
- Re-create key definitions and learn sequences that were added to a section file by using EXTEND TPU

2 This version does not include the section file EDTSECINI, the EDT keypad emulator. The EDT keypad emulator is no longer supported. EVE now offers the option of activating an EDT-like keypad. To activate this keypad, invoke EVE and then use the EVE command SET KEYPAD EDT.

3 The section file logical name has been changed from TPUSECINI to TPU$SECTION. By default, TPU$SECTION is defined as the EVE section file, EVE$SECTION.

Some products layered on VAXTPU do not work with the newest version of VAXTPU unless a new logical name is defined for the system running these layered products.

The products requiring the systemwide logical definition are those having the following characteristics:

- Layered on VAXTPU or on an editor that is layered on VAXTPU
- Use the VAXTPU callable interface
- Use the section file TPUSECINI

DATATRIEVE is an example of a product with these characteristics.

To make such products work with the newest version of VAXTPU, first recompile the TPUSECINI section file. Then define a systemwide logical name TPU$SECTION pointing to the recompiled TPUSECINI section file.

4 The command file logical name TPUINI has been replaced by a new logical name, TPU$COMMAND. By default, TPU$COMMAND is undefined. VAXTPU checks the current directory to see if you have created a command file with that logical name.

5 This version of VAXTPU includes modifications to SET (DEBUG) and GET_INFO (DEBUG, ...). In addition, the default VAXTPU debugger has been completely redesigned and rewritten. As a result, the following debugger-related features are inconsistent with previous versions:

- If your code calls a user-written debugger created under a previous version of VAXTPU, the calls to the debugger will no longer work.
- In previous versions, by default, single-step execution was disabled. In this version, by default, single-step execution is turned on.
- If you are stepping through a list of parameters, local variables, or breakpoints using the call GET_INFO (DEBUG, ...), the signaled error indicating that there are no more elements is TPU$_NONAMES. In previous versions, this status was signaled with TPU$_FAILURE.

6 In this version, subpattern references are evaluated at pattern build time instead of at search time. Any code that relies on the previous method of pattern evaluation will be incompatible. For example, suppose your code contained the following sequence:

```
pat1 := "xyz";
pat2 := pat1 & "zy";
pat1 := "abc";
```

This sequence would match "xyzzy", not "abczy." In addition, this version signals an error if a pattern references a subpattern that has not yet been defined. For example, the following sequence would produce an error:

```
pat2 := pat1 & "xy";
pat1 := "xyz";
```

7 Depth first searches, referred to as "seek" searches in documentation of previous versions of VAXTPU, have been removed. All searches are now breadth first, referred to as "incremental" searches in documentation of previous versions of VAXTPU. For example, suppose your code contained the following statement:

```
SEARCH ("a" | "b" | "c", FORWARD);
```

This statement now matches c, not a, in the text "c b a".

8 If you use the /READ_ONLY qualifier to the EDIT/TPU command, you are not only prevented from writing the buffer contents to a file, but you are also prevented from making any editing changes in the buffer.

9.13.2 Problems in VAXTPU

The following problems are present in this version of VAXTPU. Most of these problems will be resolved in the next version of VAXTPU.

1 If you invoke VAXTPU with the /NODISPLAY modifier, the use of either the READ_CHAR or READ_KEY built-in causes a fatal internal error in VAXTPU. There is no way to avoid this problem except to avoid using these built-ins while working in NODISPLAY mode.

2 The alternation operator (|) does not work correctly in conjunction with the partial pattern assignment operator (@). You should avoid constructing pattern definitions in which the alternation operator is used in nested parentheses and the result is assigned using the partial pattern assignment operator. For example, VAXTPU does not correctly handle a pattern definition such as the following:

```
((a|b) + d) @ c
```

To avoid this problem, code such pattern definitions in distributed form. For example, VAXTPU correctly handles the following pattern definition:

```
((a + d) @ c) | ((b + d) @ c)
```

3 Free markers (markers not bound to text) behave differently than bound markers in certain cases. Since the visible behavior of free markers is supposed to be the same as that of bound markers, the discrepancy is considered a problem.

To understand the difference how free markers behave and how bound markers behave, note that certain built-ins (such as COPY_TEXT) cause VAXTPU to insert spaces into a buffer to fill the area between a free marker and the text closest to the free marker. This is referred to as *padding*. (For a list of all the built-ins that cause padding, see Section 7.4 of Chapter 7 of the *VAX Text Processing Utility Manual*.

When you insert text using a built-in that causes padding, any free marker to the right of the inserted text is pushed rightward by the number of padding spaces that VAXTPU inserted. However, if you use the same built-in to insert text on a line containing a bound marker instead of a free marker, no padding spaces are inserted. This is the difference in the behavior of the two kinds of markers.

The example in the following paragraphs illustrates what happens in the case of a free marker and a bound marker. In this example, a period (.) represents an unoccupied screen location and an underscore (_) represents a screen location filled by a space. An uppercase "M" represents a marker, either free or bound. The numerals represent screen columns.

Suppose you have two lines in a buffer. One line contains the text "FOOBAR", then ten unoccupied screen columns, and then a free marker. The other contains the text "FOOBAR", then ten spaces, and then a bound marker. The situation can be represented as follows:

```
Line with free marker:          Line with bound marker:

12345678901234567890            12345678901234567890
FOOBAR.........M                 FOOBAR__________M
```

Suppose the character "x" is placed in each line in the tenth screen column, using the COPY_TEXT built-in. (COPY_TEXT is one of the built-ins that causes VAXTPU to insert padding spaces in a line.) After the "x" is placed in each line, the free marker is pushed four screen columns to the right by the three padding spaces plus the "x". However, the bound marker is pushed only one screen column to the right since VAXTPU does not insert any padding spaces in front of the bound marker. The result can be represented as follows:

```
Line with free marker:          Line with bound marker:

12345678901234567890            12345678901234567890
FOOBAR___x.........M            FOOBAR___x_______M
```

In the next release of VAXTPU, this problem will be fixed so that free markers will behave like bound markers.

4 The VAXTPU debugger (supplied in TPU$DEBUG.TPU) prompts for the name of the file to be debugged even when the file is already in a buffer known to the debugger. You must supply the file name in response to the prompt.

5 The VAXTPU debugger HELP does not contain a topic on the debugger command QUIT.

6 The HELP on the CREATE_BUFFER built-in does not explain how to use the third parameter. The third parameter is of type buffer. You have the option of using it only if you specify the optional second parameter; however, you need not use the third parameter at all.

The third parameter is the buffer you want to use as a template for the buffer to be created. VAXTPU copies buffer attributes such as margin settings, tab stops, and modifiable status from the template buffer and gives these attributes to the newly-created buffer.

For more information on how to use the third parameter to the CREATE_BUFFER built-in, see the documentation of this built-in in Chapter 4 of the *VAX Text Processing Utility Manual*.

9.13.3 Incompatibility with Future Versions

The most recent version of VAXTPU contains two undocumented keywords, CHARACTERS and COORDINATES. In the EVE source code, these keywords are occasionally used as parameters to GET_INFO and a variety of other built-ins. Despite this usage, do not put these keywords in your code. In a future release, VAXTPU will not accept these as valid keywords.

9.13.4 VAXTPU — Restrictions

This section describes restrictions in the newest version of VAXTPU. It is not known at this time when these restrictions will be removed.

1 Journaling now works as documented for previous versions of VAXTPU. However, although the new journaling code now creates journal files that work as documented, any journal file that was broken under a previous version remains broken under the newest version.

2 VAXTPU manipulates data in a process' virtual memory space. If the sum of the VAXTPU images, data structures and files in memory exceeds the virtual address space, VAXTPU may abort with a fatal internal error. VAXTPU does not give any warning that you are approaching the virtual address space limit for your process.

You can avoid this fatal internal error by increasing the virtual address space available to a process. The virtual address space is controlled by the following two factors:

- The SYSGEN parameter VIRTUALPAGECNT
- The page file quota of the account you are using

The VIRTUALPAGECNT parameter controls the number of virtual pages that can be mapped for a process. For more information on VIRTUALPAGECNT, see the description of this parameter in the *System Generation Utility Manual*, Section A.2.

The page file quota controls the number of pages in the system paging file that can be allocated to your process. For more information on the page file quota, see the description of the /PGFLQUOTA qualifier in the *VMS Authorize Utility Manual*.

You may need to modify both the VIRTUALPAGECNT parameter and the page file quota to raise the virtual address space.

If VAXTPU exceeds the address space and generates the fatal error, you can increase the virtual address space and then recover your work by replaying the journal file.

Removal of this restriction is planned for a future version of VAXTPU.

9.13.5 VAXTPU Documentation — Notes

The documentation for VAXTPU contains the following errors and omissions, which will be corrected the next time the manual is revised:

1 The qualifier /[NO]DEBUG Table 1-1 in Chapter 1 of the *VAX Text Processing Utility Manual* should have the following syntax:
/[NO]DEBUG[=filespec]

2 In the *System Messages and Recovery Procedures Reference Manual*, 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.

3 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.

4 In Chapter 13 of the *VMS Utility Routines Manual*, Example 13-1 (Sample VAX BLISS Template for Callable VAXTPU) does not work. The following example should be substituted:

Example 9–1 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
```

Example 9–1 Cont'd. on next page

Example 9–1 (Cont.) Sample VAX BLISS Template for Callable VAXTPU

```
!
! 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$cleanup;                ! Have VAXTPU cleanup after itself
!
! Declare the VAXTPU literals
!
EXTERNAL LITERAL
    tpu$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 =
```

Example 9–1 Cont'd. on next page

Example 9–1 (Cont.) Sample VAX BLISS Template for Callable VAXTPU

```
    BEGIN
!++
! Main entry point of your program
!--
! Your_initialization_routine must be declared as a BPV

    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
```

Example 9–1 Cont'd. on next page

Example 9–1 (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),
```

Example 9–1 Cont'd. on next page

Example 9–1 (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);
```

Example 9–1 Cont'd. on next page

Example 9–1 (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
```

5 In Chapter 13 of the *VMS Utility Routines Manual,* Example 13-2 does not work. The following example should be substituted:

Example 9–2 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
C
C     Local Flags and Indices
C
      INTEGER*4       CLEANUP_FLAG       ! flag(s) for VAXTPU cleanup
      INTEGER*4       RET_STATUS
      INTEGER*4       MATCH_STATUS
```

Example 9–2 Cont'd. on next page

Example 9–2 (Cont.) Building a Callback Item List with VAX FORTRAN

```
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
```

Example 9–2 Cont'd. on next page

Example 9–2 (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'
```

Example 9–2 Cont'd. on next page

Example 9–2 (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
```

6 The *VAX Text Processing Utility Manual* does not explicitly mention that the VAXTPU callable interface is documented in Chapter 13 of the *VMS Utility Routines Manual*.

7 The preface of the *VAX Text Processing Utility Manual* incorrectly states that VAXTPU supports the use of mouse buttons on workstations running VWS Version 3.3 or higher. The correct version is VWS Version 4.0 or higher.

8 The topic of zero-length ranges was omitted from the discussion of the range data type in Chapter 2 of the *VAX Text Processing Utility Manual*.

VAXTPU does not support ranges of zero length unless the range is created at the end of a buffer. All other ranges contain at least one character (which could be a space character) or a line-end (if the range is created on a blank line).

If you assign to a variable a partial pattern whose definition does not include any characters, the partial pattern variable contains the character or line-end at the point in the file where the partial pattern was matched. For example, in any of the following patterns containing partial pattern assignments, the variable "partial_pattern_variable" contains the character or line-end at the point in the file where the partial pattern was matched:

```
"" @ partial_pattern_variable
ANCHOR @ partial_pattern_variable
UNANCHOR @ partial_pattern_variable
```

Note that if you use one of the preceding patterns when the cursor is free (that is, in an area that does not contain text, such as the area after the end of a line) the variable "partial_pattern_variable" contains the line-end or character nearest to the cursor.

9 The description of the COPY_TEXT built-in in Chapter 4 of the *VAX Text Processing Utility Manual* does not mention that COPY_TEXT can return the codes TPU$_LINETOOLONG and TPU$_TRUNCATE. TPU$_LINETOOLONG is a warning indicating that no more text can be added to the line. TPU$_TRUNCATE is a warning indicating that the line has been truncated.

10 The description of the CREATE_BUFFER built-in in Chapter 4 of the *VAX Text Processing Utility Manual* does not mention that CREATE_BUFFER can return the status TPU$_OPENIN. This status is an error indicating that the specified input file was not opened.

11 The description of the CURRENT_COLUMN built-in in Chapter 4 of the *VAX Text Processing Utility Manual* inaccurately describes the value returned by GET_INFO (window, "current_column"). The built-in description states that GET_INFO (window, "current_column") returns the number of the column in which the cursor is located on the screen, even if the cursor has been moved using CURSOR_HORIZONTAL since the last screen update.

However, the value that GET_INFO (window, "current_column") actually returns is the number of the column in which the cursor was located after the most recent screen update. In other words, if free cursor motion (as a result of CURSOR_HORIZONTAL) has occurred since the most recent update, the value returned by GET_INFO (window, "current_column") does not reflect the fact that the free cursor motion has occurred.

The description of CURRENT_COLUMN lists two methods of obtaining an accurate current column value. The first listed method is to use either the GET_INFO (window, "current_column") call or the GET_INFO (CURRENT_WINDOW, "current_column") call. This method does not work. The second method is to use the call GET_INFO (buffer, "offset_column"). This method does return an accurate current column value.

In a future release of VAXTPU, the value returned by CURRENT_COLUMN (and its corresponding GET_INFO call) and by CURRENT_ROW (and its corresponding GET_INFO call) will be the value that would be returned if you had caused an explicit screen update immediately prior to using the built-in or GET_INFO call.

12 The description of the FILE_SEARCH built-in in Chapter 4 of the *VAX Text Processing Utility Manual* should say that if FILE_SEARCH does not find a file specification matching the user's request, or if FILE_SEARCH has found one or more but cannot find another, FILE_SEARCH returns a null string but not an error status. If an unexpected error occurs during the search, FILE_SEARCH returns the status TPU$_SEARCHFAIL. In such a case, the reason for the error is displayed in the message buffer.

13 The description of the FILL built-in in Chapter 4 of the *VAX Text Processing Utility Manual* does not mention that FILL can return the status TPU$_INVRANGE. This status is a warning indicating that you specified an invalid range enclosure.

14 The description of the GET_INFO built-in in Chapter 4 of the *VAX Text Processing Utility Manual* has the following errors and omissions:

- The list of GET_INFO calls available for supplying information about a buffer is incomplete. The table describing these calls should include the following entries:

Parameter1	*Parameter2*	Return Value	Description of Return Value
A buffer variable	"before_bol"	Integer (1 or 0)[1]	Value that indicates whether the editing point is located before the beginning of a line.
	"beyond_eob"	Integer (1 or 0)[1]	Value that indicates whether the editing point is located beyond the end of a buffer.
	"beyond_eol"	Integer (1 or 0)[1]	Value that indicates whether the editing point is located beyond the end of a line.
	"bound"	Integer (1 or 0)[1]	Value that indicates whether the editing point is attached to a character or is detached (free).
	"middle_of_tab"	Integer (1 or 0)[1]	Value that indicates whether the editing point is attached to a character or is detached (free).

[1] An integer value of 1 indicates true and an integer value of 0 indicates false.

- The description of the call GET_INFO (window, "visible_length") should read as follows:
"Visible length" of the window (includes status line). This value differs from the value returned by GET_INFO (window, "original_length") in that the value returned by "visible_length" is the original length *minus* the number of window lines (if any) hidden by occluding windows.
- The description of the call GET_INFO (buffer, "right_margin_action") does not mention that this call returns TPU$K_UNSPECIFIED if the buffer does not have a right margin action routine.
- The description of the call GET_INFO (buffer, "next_marker") does not mention that you must use the call GET_INFO (buffer, "first_marker") before using the call GET_INFO (buffer, "next_marker").

- The description of the call GET_INFO (buffer, "next_range") does not mention that you must use the call GET_INFO (buffer, "first_range") before using the call GET_INFO (buffer, "next_range").
- The descriptions of the call GET_INFO (DEBUG, "next") and the call GET_INFO (DEBUG, "previous") neglect to say that these calls do not work unless you have already used one of the following calls:
  - GET_INFO (DEBUG, "local")
  - GET_INFO (DEBUG, "breakpoint")
  - GET_INFO (DEBUG, "parameter")
- The description of the call GET_INFO (DEBUG, "previous") incorrectly states that the call returns the next parameter, variable or breakpoint. The description should say that the call returns the previous parameter, variable or breakpoint.
- The description of the call GET_INFO (KEY_MAP, "current") incorrectly states that the call returns the first key map in the key map list. The description should say that the call returns the current key map in the key map list.
- The description of the call GET_INFO (key name, "key_type") incorrectly states that the call can return the keyword SHIFT. The correct syntax is SHIFT_KEY.
- The description of the call GET_INFO (SYSTEM, "pad_overstruck_tabs") incorrectly states that the integer returned indicates whether VAXTPU preserves a tab character. The description should say that the integer indicates whether VAXTPU preserves the white space created by a tab character.

15 The description of the POSITION built-in in Chapter 4 of the *VAX Text Processing Utility Manual* does not mention the parameter TEXT. The POSITION built-in still accepts only one parameter; TEXT is one of the keywords that can be used as a value for this single parameter. If the editing point is at a free-cursor location (a portion of the screen where there is no text), the statement POSITION (TEXT) establishes the editing point at the nearest location that has a text character in it. (The character may be a space or an end-of-line.)

16 The description of the SELECT built-in in Chapter 4 of the *VAX Text Processing Utility Manual* states that you can have only one visible select area at a time on the screen. This is incorrect. It is possible to have two visible select areas on the screen at the same time if each window is mapped to a different buffer. However, if there are two or more windows mapped to the same buffer, only one window (the current window) visibly displays the select area. If none of the windows on the screen is current, the visible window that was most recently current displays the select area.

17 Chapter 4 of the *VAX Text Processing Utility Manual* omits the description of the built-in SET (MODIFIED...). The syntax is as follows:

SET (MODIFIED, buffer, { ON / OFF }).

The parameters are as follows:

MODIFIED	A keyword indicating whether a buffer has been modified.
buffer	The buffer you want VAXTPU to mark as "modified" or "unmodified".
ON	A keyword indicating that VAXTPU is to mark the buffer as "modified".
OFF	A keyword indicating that VAXTPU is to mark the buffer as "unmodified".

18 The *VAX Text Processing Utility Manual* does not contain a necessary detail on the behavior of SET (STATUS_LINE). When you remove a status line by specifying a null string as the last parameter to SET (STATUS_LINE), the built-in does not require that the window be at least two lines high. That is, SET (STATUS_LINE) never returns TPU$_BADWINDLEN under these circumstances; instead, the status line is simply turned off, regardless of window length.

19 The description of the SPAWN built-in in Chapter 4 of *VAX Text Processing Utility Manual* states that the default for SPAWN's optional second parameter is ON. This statement is not correct. The default for the optional second parameter is ON if the string specified for the first parameter is something other than the null string. If you specify the null string as the first parameter to SPAWN, the default for the second parameter is OFF.

20 The description of the INT built-in in Chapter 4 of the *VAX Text Processing Utility Manual* has two errors in the syntax section. First, the syntax section does not show that INT can take either a keyword or a string as the first parameter. Second, the syntax section does not mention the second, optional, integer parameter. The correct syntax for the INT built-in is as follows:

integer2 := INT ({ keyword / string [, integer1] })

The return value and parameters are as follows:

integer2	The integer representation of the string or keyword you specify.
keyword	A keyword whose internal value you want.
string	A string consisting of one or more numeric characters, whose integer value you want INT to return
integer1	An integer specifying the radix (or base) of the string being converted. The default radix is 10. The other allowable values are 8 and 16. If you specify some other value, INT returns the status TPU$_BADVALUE.

21 The description of the STR built-in in Chapter 4 of the *VAX Text Processing Utility Manual* does not mention that STR can take a second, optional parameter of type integer. This optional parameter can be used only if the first parameter is also of type integer. The optional parameter specifies the radix (or base) you want VAXTPU to use when converting the integer to a string representation. The default radix is 10. The other allowable values are 8 and 16. If you specify some other value, STR returns the status TPU$_BADVALUE.

9.14 I/O Synchronization Using Symmetric Multiprocessing

Nearly every change made to VMS Version 5.0 for Symmetric Multiprocessing (SMP) support is invisible to general users and application programmers. However, there is one notable exception—the increased parallelism afforded by SMP can change the timing associated with I/O operations. This may expose previously unnoticed synchronization errors in incorrectly coded application programs. The following sections provide a detailed description of the changes and how they affect users and programmers.

9.14.1 Description of the SMP Change

In Version 4.n of VMS, certain I/O operations to particular devices were completed before control was passed back to the caller of the $QIO system service. For these operations, the $QIO (asynchronous) and $QIOW (synchronous) forms of the $QIO system service were basically the same. Programs that did not test for I/O completion and programs that incorrectly tested for I/O completion only worked on occasion.

In VMS Version 5.0, SMP allows the concurrent execution of programs and the system code that supports I/O completion. Parts of an I/O request may be in progress on one processor in the SMP configuration when control is passed back to the caller of a $QIO executing on a different processor.

Programs that perform proper synchronization (testing for I/O completion) with either the $QIOW form of the service or explicit use of the $SYNCH service will wait until the I/O operation is complete. Incorrectly coded programs may process adversely as a result of the change in timing.

Another method for I/O completion testing is the use of asynchronous system trap (AST) notification of completion. A program using AST synchronization recognizes a timing change with SMP in Version 5.0. In Version 4.n, if a $QIO request was issued and the data was immediately available, the AST routine would be executed before the $QIO returned to the caller. In the same situation using SMP on Version 5.0, it is possible for the $QIO to return immediately and the AST to be delivered very shortly thereafter.

Further examination of the effects of these changes on components of the VMS operating system, certain layered products, and a small number of third-party applications, required a second change to preserve some of the former capabilities. A small set of I/O operations have been artificially restricted to complete synchronously, even when the asychronous $QIO request is used. Specifically, I/O operations that use the Set or Sense Mode (or Characteristics) I/O function codes are completed when control is passed back to the caller. This second change preserves the behavior of at least some of the incorrectly coded programs.

I/O operations that specify a read or write or other function code to either a real device or a pseudo device (for example, a mailbox) are affected by the increased parallelism under SMP. These incorrectly coded programs may fail when runing on SMP processors.

9.14.2 General User Visibility

The actual behavior of a program with incorrect synchronization is difficult to predict but at least two modes of incorrect behavior can be detected.

The most common incorrect behavior is a premature test of the contents of the I/O status block. For example, the program may report an error on an I/O request when, in fact, the I/O request is still in progress. This error will often be a report that error "00000000" occurred, since the I/O status block (IOSB) contains 0 while an I/O request is in progress.

If the program does not look at the I/O status block, the program will probably continue to execute when, in fact, the I/O operation is still in progress. This may cause adverse behavior.

A less likely example of this change occurs when a program is affected by the change in timing while using AST synchronization. The program may be properly synchronized with respect to the I/O itself, however, it has a lack of coordination between the activity occurring in the AST routine and the activity in the mainline thread of execution. Shortly after the call to $QIO, the mainline thread manipulates a linked list that is also modified by the AST routine. The lack of synchronization between the two threads of execution may result in both threads manipulating the list at the same time.

9.14.3 Programmer Responsibility

All calls to the $QIO system service must properly check for I/O completion. This means that all calls must use the $QIOW service with an I/O status block or they must include an explicit call to the $SYNCH system service to properly test for the completion of a $QIO system service call. The VMS Version 5.0 documentation description of system services contains a section devoted to a discussion of proper synchronization.

Programs that issue read or write requests to real I/O devices or to pseudo devices (for example, mailboxes) and do not test for I/O completion before using the results must be changed immediately to work with VMS Version 5.0.

Programs that use the Set or Sense Mode (or Characteristics) function codes without correctly testing for completion continue to work for VMS Version 5.0. However, these programs are not correct and DIGITAL recommends that you change them. Because of continuous development of VMS symmetric multiprocessing, other changes to the I/O system will be made. A future release of the VMS operating system likely will change the internal operation of the SET/SENSE and MODE/CHARACTERISTICS function codes to be asynchronous.

Presently, there are no set guidelines to help programmers detect timing problems in their applications that were benign in Version 4.n, but are exposed by this change.

9.15 Driver Manipulation of I/O Postprocessing Queues

VMS Version 5.0 supplies several system routines that enable non-DIGITAL-supplied device drivers to initiate completion processing of an I/O request. Such device drivers should not manipulate the I/O postprocessing queues directly. Instead, they should call the appropriate system routine, or invoke a macro that does so, to insert an I/O request packet in the queue.

Among the system routines that initiate I/O postprocessing are the following:

Routine	Function
IOC$REQCOM	Completes an I/O operation on a device unit, requests I/O postprocessing of the current request, and starts the next I/O request waiting for the device
COM$POST	Initiates device-independent postprocessing of an I/O request independent of the status of the device unit and increments the unit's operations count
COM$POST_NOCNT	Initiates device-independent postprocessing of an I/O request independent of the status of the device unit and without incrementing the unit's operations count

The *VMS Device Support Manual* discusses each of these routines except COM$POST_NOCNT. A description of COM$POST_NOCNT follows.

COM$POST_NOCNT

Initiates device-independent postprocessing of an I/O request independent of the status of the device unit and without incrementing the unit's operations count.

module

COMDRVSUB

input

Location	Contents
R3	Address of IRP
IRP$L_MEDIA	Data to be copied to the I/O status block
IRP$L_MEDIA+4	Data to be copied to the I/O status block

output

Location	Contents
R0	Destroyed

synchronization

Drivers call COM$POST_NOCNT at any fork IPL. COM$POST_NOCNT executes at its caller's IPL and returns control at that IPL. The caller retains any spin locks it held at the time of the call.

DESCRIPTION

A driver fork process calls COM$POST_NOCNT after it has completed device-dependent I/O processing for an I/O request initiated by EXE$ALTQUEPKT. Because COM$POST_NOCNT, unlike COM$POST, does not increment the unit's operations count (UCB$L_OPCNT), the driver uses COM$POST_NOCNT to initiate completion processing for an I/O request when the associated UCB is not available.

COM$POST inserts the IRP into the I/O postprocessing queue, requests an IPL$_IOPOST software interrupt, and returns control to its caller. Unlike IOC$IOPOST, it does not attempt to dequeue any IRP waiting for the device or change the busy status of the device.

9.16 DELTA/XDELTA — Using XDELTA on SMP Systems

When using XDELTA on symmetrical multiprocessing (SMP) systems, only one processor can be in XDELTA at a time. If one processor attempts to enter XDELTA while another processor is using XDELTA, it waits until the other processor has exited XDELTA. If the processor using XDELTA sets a breakpoint, other SMP processors are aware of the breakpoint. Therefore, when the code with the XDELTA breakpoint is executed on another processor, that processor enters XDELTA and stops at the specified breakpoint.

XDELTA uses its own system control block (SCB) to direct all interrupt handling to an error handling routine in XDELTA. Therefore, an error encountered by XDELTA does not affect any of the other processors that share the standard system SCB.

9.17 EXE$BUFFRQUOTA and EXE$BUFQUOPRC Routines Replaced

For VMS Version 5.0, the EXE$BUFFRQUOTA and EXE$BUFQUOPRC routines have been removed. Avoid all direct manipulation of nonpaged pool quotas (JIB$L_BYTCNT and JIB$L_BYTLM) by programs. For more information, please refer to the *VMS Device Support Manual*.

9.18 LAT — LTDRIVER Changes

This section describes the changes made to the LTDRIVER for Version 5.0.

9.18.1 Problem Corrected

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.

9.18.2 New QIO Support

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.

9.19 LPA11–K Driver (LADRIVER) — Changing Timeouts Allowed

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 SYS$LOADABLE_IMAGES:LADRIVER.EXE/OUTPUT=SYS$LOADABLE_IMAGES:LADRIVER.EXE
PATCH> SET ECO 25
PATCH> REPLACE/INSTRUCTION LA$TIMEOUT_VALUE
OLD> 'PUSHL      I^#00000002'
OLD> EXIT
NEW> 'PUSHL      I^#0000003C'
NEW> EXIT
PATCH> UPDATE
PATCH> EXIT
```

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.

9.20 Modular Executive — Effect on System Services

This section describes the effect upon system services of changes to the modular executive.

9.20.1 New Description for the $MTACCESS

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.

9.20.2 Instructions for Loading a Site-Specific Executive Loaded Image

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 PSECT

DECLARE_PSECT   EXEC$NONPAGED_CODE   ; Nonpageable code PSCET
```

2 Assemble the source module by using the following command:

```
$ MACRO/OBJ=MTACCESS MTACCESS+SYS$LIBRARY:LIB.MLB/LIB
```

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).

```
$ SYSMAN SYS_LOADABLE REMOVE _LOCAL_ MTACCESS
```

2 Repeat steps c and d from instruction 4.

9.21 Modular Executive — Effect on Device Drivers

This section describes the effect of the Modular Executive upon device drivers.

9.21.1 Preparing a Driver for Loading into the Operating System

After the driver is linked, copy the driver into the SYS$LOADABLE_IMAGES directory. You must have the appropriate system privileges to put files into this directory.

Note that the SYSGEN LOAD and the SYSGEN CONNECT commands use SYS$LOADABLE_IMAGES or SYS$SYSTEM as the default directories. That is, if the driver image is not found in the SYS$LOADABLE_IMAGES directory, SYSGEN looks for the driver in the SYS$SYSTEM directory.

9.22 Modular Executive — Effect on System Dump Analyzer

This section describes the effect of the Modular Executive upon the System Dump Analyzer (SDA). Many of the topics in this section are also discussed in detail in the SDA section of the *VMS Version 5.0 New Features Manual*.

9.22.1 SHOW EXECUTIVE Command

Version 5.0 of the System Dump Analyzer includes a new command, SHOW EXECUTIVE, which displays information about all executive loaded images in the system.

Format	SHOW EXECUTIVE
Command parameter	None
Command qualifier	None

The SHOW EXECUTIVE command displays the following types of information:

- Name of the executive loaded image
- Starting address of the executive loaded image
- Ending address of the executive loaded image
- Length of the executive loaded image

9.22.2 READ /EXECUTIVE Qualifier

For VMS Version 5.0, the /EXECUTIVE qualifer was added to the READ command.

The READ/EXECUTIVE qualifier causes SDA to read the global symbols from all the executive loaded images and add those symbols to the SDA symbol table.

9.23 Modular Executive — Effect on DELTA/XDELTA

This section describes the effect of the Modular executive upon DELTA /XDELTA.

9.23.1 XDELTA ;L Command

This command displays the list of executive loaded images loaded in the system. It displays all executive loaded images loaded in the system with their starting and ending addresses.

The format of this command is as follows:

;L (List Executive loaded images)

9.24 Caution on Use of NOP Instruction as a Delay Mechanism

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.

9.25 Processor Register Definition Symbols

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 reference 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 reference 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.

9.26 SET HOST/DTE/DIAL Command — Problem and Solution

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.

10 Documentation Information

This chapter describes changes you should make to specific manuals in your VMS Version 5.0 documentation set.

10.1 *VMS DCL Dictionary*

In the *VMS DCL Dictionary*, 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).

10.2 *VMS VAXcluster Manual*

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.

10.3 VMS Device Support Manual

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*.

10.4 VMS Debugger Manual

On page CD-13 of the *VMS Debugger Manual*, the description of the CANCEL ALL command has the following errors:

The second sentence of the command overview at the top of the page should read as follows: "Restores some modes established with the SET MODE command to their default values."

The fourth item of the list in the Description section should read as follows:

> Restores some modes established with the SET MODE command to their default values. This is equivalent to entering the following command:
>
> ```
> DBG> SET MODE LINE,SYMBOLIC,NOG_FLOAT
> ```

10.5 VMS I/O User's Reference Manual: Part II

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.

10.6 VMS System Messages and Recovery Procedures Reference Manual: Part I

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.

10.7 VMS Mail Utility Manual

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.

10.8 VMS DECnet Test Sender/DECnet Test Receiver Utility Manual

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

10.9 *VMS Networking Manual*

In Section 3.6.2.2, Page 3-59 of the *VMS Networking Manual*, 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.

10.10 *Guide to VMS Software Installation*

Previously, the *Guide to VMS Software Installation* manual contained information on console subsystems, naming devices, VMSINSTAL, UETP, and various operations for all the VAX computers. For Version 5.0, this information has been incorporated into the installation and operations guides for each VAX computer and the *Guide to VMS Software Installation* has been discontinued.

10.11 *VMS Monitor Utility Manual*

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.

10.12 *VMS Network Control Program Manual*

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 (_).

10.13 VMS Convert and Convert/Reclaim Utility Manual

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.

10.14 VMS Linker Utility Manual

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.

10.15 VMS Record Management Services Manual

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)
```

10.16 VMS System Services Volume

The documentation for the $GETDVI system service in the *VMS System Services Volume* states erroneously that the $DCDEF macro defines symbols for teller terminal device types. The description for the DVI$_DEVTYPE item code should state that the $TTDEF macro defines the symbols for teller terminal devices.

10.17 VMS Run-Time Library Routines Volume

The following sections contain corrections you should make to the *VMS Run-Time Library Routines Volume.*

10.17.1 RTL LIB$ (Library) Facility

The following notes apply to the RTL LIB$ (Library) Facility routines.

10.17.1.1 Corrections to the LIB$ADAWI Routine Description

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.

10.17.1.2 Omissions from the LIB$SPAWN Routine Description

The *VMS RTL LIB$ (Library) Manual* omits the last argument to LIB$SPAWN. 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, these privileges are not available to the subprocess until a SET PROCESS /PRIVILEGE command or equivalent SYS$SETPRV call is performed in the subprocess to enable these 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.

The **cli** argument must be specified in all uppercase characters.

10.17.1.3 Correction to LIB$CREATE_VM_ZONE

In the LIB$CREATE_VM_ZONE routine description, the argument *number-of-areas* is listed; however, *number-of-areas* does not exist. LIB$CREATE_VM_ZONE has thirteen, not fourteen, arguments.

10.18 VAX RMS Journaling Manual

The following descriptions supersede the descriptions provided in the *VAX RMS Journaling Manual*.

10.18.1 Recovery Unit Flags Field XAB$W_RU_FLAGS

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).

10.18.2 SET FILE/RU_ACTIVE

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

```
$ SET FILE/RU_FACILITY=1/RU_ACTIVE=0 FINANCE_DISK:[PAYROLL]WEEKLY.DAT
```

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.

10.18.3 SET FILE /RU_FACILITY

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.

A Run-Time Library Argument Notation

This appendix documents the notation used to describe the arguments of a Run-Time Library routine in the VMS source files. This notation is commonly known as dot notation, and it specifies the following information about routine arguments:

- The argument name
- The type of access allowed for that argument
- The data type of the argument
- The mechanism used to pass the argument to or from the routine
- The form of the argument (if any)

The documentation for the Run-Time Library routines transforms this notation into the following format:

argument name

VMS usage:	VMS usage as interpreted from the data type and the usage of the argument in the routine
type:	Data type of the argument as documented in the source files
access:	Access to the argument as documented in the source files
mechanism:	Passing mechanism as documented in the source files

As shown in the previous table, most of the information in the Run-Time Library routines documentation is taken directly from the dot notation information listed in the source files. Only the VMS usage entry is interpreted from the data type and the actual usage of the argument in the routine.

The dot notation format for a Run-Time Library routine argument is as follows:

<name>.<access type> <data type>.<passing mechanism> <argument form>

The following sections discuss each of these fields in more detail.

A.1 Name

The *<name>* characteristic is a mnemonic for the parameter name or function value specifier. This name is arbitrary.

A.2 Access Type

The *<access type>* characteristic is a single letter denoting the type of access that the routine can make to the argument. The types of access and their corresponding notations are as follows:

a Reserved.

b Reserved.

c The argument is an address of a routine to be (optionally) called after unwinding the stack (return). No *<data type>* field is given, because the argument is a sequence of instructions.

f The argument is an address of a function to be (optionally) called without unwinding the stack. The *<data type>* field indicates the data type (ZEM or BPV) used to represent the function. Immediately following ZEM or BPV is the data type of the function value. For example, func.fzeml.r indicates that the argument list entry contains the address of a function that returns a signed longword value in R0.

j The argument is an address to which (optionally) to jump after unwinding the stack (return). No *<data type>* field is given, because the argument is a sequence of instructions, such as FORTRAN "ERR=".

m The argument can be modified; that is, read and written.

r The argument can be read only.

s The argument is an address of a subroutine to be (optionally) called without unwinding the stack. The *<data type>* field indicates the data type (ZEM or BPV) used to represent the subroutine.

v Reserved.

w The argument can be written only.

A.3 Data Type

The *<data type>* characteristic is a letter denoting the primary data type with trailing qualifier letters to further identify the data type. Note that the routine must reference only the size specified to avoid access violations.

The data types and their notations are given in the following table.

Code	Data Type
adt	Absolute date and time
b	Byte integer (signed)
blv	Bound label value
bpv	Bound procedure value
bu	Byte (unsigned)
cit	COBOL intermediate temporary
d	D_floating
dc	D_floating complex
dsc	Descriptor
f	F_floating
fc	F_floating complex
g	G_floating
gc	G_floating complex
h	H_floating
hc	H_floating complex
l	Longword integer (signed)
lu	Longword integer (unsigned)
nl	Numeric string, left separate sign
nlo	Numeric string, left overpunched sign
nr	Numeric string, right separate sign
nro	Numeric string, right overpunched sign
nu	Numeric string, unsigned
nz	Numeric string, zoned sign
o	Octaword integer (signed)
ou	Octaword integer (unsigned)
p	Packed decimal string
q	Quadword integer (signed)
qu	Quadword integer (unsigned)
t	Character string
v	Aligned bit string
vt	Varying character string
vu	Unaligned bit string
w	Word integer (signed)
wu	Word integer (unsigned)
z	Unspecified
zem	Procedure entry mask
zi	Sequence of instruction

A.4 Passing Mechanism

The *<passing mechanism>* characteristic is a single letter indicating the passing mechanism that the routine expects to be used for the argument. The three types of passing mechanisms and their notations are listed in the following table.

Code	Passing Mechanism
d	By descriptor; that is, the actual contents of the argument list entry are the longword address of a descriptor. The descriptor is two or more longwords that specify further information about the argument. Note that when *<passing mechanism>* is specified as *d*, the *<argument form>* field must also be specified to indicate the type of descriptor.
r	By reference; that is, the contents of the argument list entry are the longword address of the actual argument. If the argument is a scalar or a label, the *<argument form>* field must not be specified. However, if the argument is an array, *<argument form>* must be specified.
v	By value; that is, the contents of the argument list entry are the actual value of the argument to be passed. Note that arguments passed by value are always allocated a longword.

A.5 Argument Form

The *<argument form>* characteristic is a letter denoting the form of the argument. The types of argument forms and their notations are listed in the following table.

Code	Argument Form
	A null argument form indicates a scalar value of the specified data type.
x	This indicates any one of the supported string descriptors, as specified by the calling program in the DSC$B_CLASS field of the descriptor that it passes to the called routine.
x1	This indicates either a fixed-length or dynamic descriptor, as indicated by the calling program in the DSC$B_CLASS field of the descriptor that it passes to the called routine.
s	This indicates a fixed-length descriptor, where the contents of the argument list entry are the longword address of a two-longword scalar descriptor. When the data type field (DSC$B_DTYPE) indicates ASCII text (DSC$K_DTYPE_T), the descriptor contains the length, data type, and address of a fixed-length string. When the string is written, neither the length nor the address fields in the descriptor is modified, and the string is filled with trailing spaces or a separate argument is updated with the written length.
d	This indicates a dynamic string descriptor, where the contents of the argument list entry is the longword address of a two-longword string descriptor of the same format as that of *s*. However, when the string is written, both the length and address fields may be modified. Space is allocated dynamically by routines in the routine library.
a	This indicates an array reference or array descriptor, as indicated by the *<argument mechanism>* field. For array reference, the contents of the argument list entry is the address of an array of items of the indicated data type. The length of the array is either fixed, implied by the entries in the array, determined by another argument, or specified by prior agreement. For array descriptor, the contents of the argument list entry is the longword address of a descriptor block.
p	This indicates a routine descriptor, where the contents of the argument list entry is the longword address of a two-longword routine descriptor. The descriptor contains the address of the routine and the data type that the routine returns if it is a function. The *<access type>* must be *c*, *f*, *j*, or *s*.
sd	This indicates a decimal string descriptor. The first two longwords are similar to a descriptor; however, the third longword contains scale factor (one byte), and the number of decimal digits (one byte).
nca	This indicates a noncontiguous array descriptor. This is used when the array elements are not stored contiguously.
vs	This indicates a varying string descriptor. This is used for varying strings consisting of two fixed-length areas allocated contiguously with no padding between them.
vsa	This indicates a varying string array descriptor. This is used to specify an array of varying strings where each varying string has the same maximum length.

Code	Argument Form
ubs	This indicates an unaligned bit string descriptor. This is used to pass an unaligned bit string that starts and ends on an arbitrary bit boundary.
uba	This indicates an unaligned bit array descriptor. This is used to specify an array of unaligned bit strings. Elements of the array can be accessed directly using the VAX variable bit field instructions.
sb	This indicates a string with bounds descriptor.
ubsb	This indicates an unaligned bit string with bounds descriptor.

A.6 Examples of Dot Notation

If you are viewing the source code for the VMS Run-Time Library routines, you will see the following notation in almost every routine:

```
status.wlu.v
```

You should interpret this argument as follows:

status
Name of the argument. This argument represents the status or completion code of a routine.

w
Indicates an access of write only. The argument is to be written by the routine, not read.

lu
Indicates a data type of longword (unsigned). (Note that, because this longword integer is interpreted as a condition value and not as an integer value, the VMS usage for this argument would be *cond_value*, as shown in the routines documentation for the VMS Run-Time Library.)

v
Indicates that the argument is passed by value. That is, the routine passes this argument to the caller by placing its value in the argument list.

Index

A

Accounting logs
  processed during upgrade • 1–3
/ACL qualifier • 7–3
Alias directory entries • 7–17
  protecting • 8–66
ALL-IN-1
  installation restrictions • 9–4
ASMP
  See Asymmetric Multiprocessing
Asymmetric Multiprocessing (ASMP)
  discontinued support for VAX–11/782 • 8–6
Authorization files
  processed during upgrade • 1–3
Authorize Utility (AUTHORIZE) • 8–10 to 8–12
  UAF limits • 8–11
AUTOGEN • 8–12
  aborting during GENFILES phase • 8–12
  after Version 5.0 conversion • 4–13
  at end of phase upgrade • 3–16
  dump file size • 8–15
  error message • 8–6
  files not marked NOBACKUP • 8–14
  MSCP server mechanism • 8–14
  new feedback mechanism • 8–13
  selective crash dump files • 8–15
  swapping file size • 8–15
AUTOGEN.PAR
  creation of • 3–15

B

Backup
  build standalone BACKUP kit, after upgrade • 5–7
  console media, after upgrade • 5–7
  console media before upgrade • 2–1
  system disk, after upgrade • 5–7
  system disk before converting to B5.0 • 4–9
  system disk before upgrade • 2–1
Backup Utility (BACKUP)
  files marked for after-image journaling • 8–36
  files marked for before-image journaling • 8–36
Backup Utility (BACKUP) (cont'd.)
  files marked for recovery unit journaling • 8–36
  files with active recovery units • 8–36
BI Unibus Adapter
  installing • 8–50
Block mode
  RMS • 9–11
Boot command
  See DEFBOO and dddGEN
Boot during upgrade
  See Conversational boot
Bootstrapping
  VAX–11/750 • 8–54
    example • 8–54
BUA board
  See BI Unibus Adapter
Buffer
  allocating a physically contiguous • 10–2

C

Cautions, for upgrade • 1–2
CDROM
  Phase 2 of upgrade • 3–13
  Phase 6 of upgrade • 3–15
CI750 • 3–4
CIBOO.CMD • 8–1
  editing • 8–3
  example • 8–4
  requirement for upgrade • 3–7
Cluster upgrade
  See Concurrent upgrade
  See Rolling upgrade
COM$POST • 10–2
COM$POST_NOCNT • 9–44
Command procedures
  labels • 7–15
  processed during upgrade • 5–1
Concurrent upgrade
  description • 4–1
  pre-upgrade procedure • 4–1 to 4–2
  procedure • 4–1 to 4–2
  summary • 4–1
CONFIGURE phase
  Startup phase • 8–8

Configuring devices for upgrade • 2–6
CONSCOPY
  after upgrade • 5–7
  before upgrade • 2–1
Console media
  backing up after upgrade • 5–7
  requirement for upgrade • 2–1
  requirement for upgrade, VAX–11/725, 11/730, 11/780, 11/785, 8600, 8650 • 3–10
  update after upgrade for VAX 8530, 8550, 8700, 8800 • 5–3 to 5–5
Conversational boot
  restriction during Phase 1 of upgrade for VAX–11/750 • 3–5
  restriction during Phase 1 of upgrade for VAX 8200, 8250, 8300, 8350 • 3–7
  using for insufficient memory (8530, 8550, 8700, 8800) • 3–9
  using for insufficient memory (VAX–11/750) • 3–5
  using for insufficient memory (VAX 8200, 8250, 8300, 8350) • 3–7
Conversion program
  A5.0 to B5.0 • 4–9
  B5.0 to Version 5.0 • 4–11
  B5.0 to Version 5.0, description of • 4–10
  converting primary MAIL database • 4–12
  converting primary queue file • 4–11
  converting secondary MAIL database • 4–13
  converting secondary queue file • 4–12
  DECnet proxy database • 5–2
Copy operations
  RMS • 9–11
/CPU=[NO]ATTACHED qualifier • 7–2
Customer application software
  mixed-version limitation • 4–6
CVTNAFV5.COM • 5–2

D

DCL (Digital Command Language) • 7–1 • 7–3
dddGEN
  change required for upgrade • 2–6
  modifying for upgrade • 2–6
  restoring original versions during upgrade • 3–14
Debugger • 9–5 to 9–9
  CALL command • 9–9
Debugger (cont'd.)
  linking shareable images • 9–6
  obsolete commands • 9–6
  screen management facility (SMG) restriction • 9–5
  screen-mode • 9–7
  SET IMAGE command • 9–7
  SET SCOPE command • 9–8
  shareable images • 9–6
  supporting MACRO • 9–6
  using on VAXstation • 9–8
  Version 4.6 notes • 9–8
  Version 4.7 notes • 9–8
DECnet permanent database
  sharing local • 5–3
  sharing remote • 5–3
DECnet proxy database
  automatic conversion • 3–15
  manual conversion • 5–2
  mixed-version limitation • 4–4
  name change • 1–3, 4–3
DECnet–VAX, requirement for upgrade • 2–4
Decompressing
  help libraries after upgrade • 5–7
  object libraries after upgrade • 5–7
  space requirement • 5–7
DEFBOO
  change required for upgrade • 2–6
  modifying for upgrade • 2–6
  restoring original versions during upgrade • 3–14
DEFINE/FORM command
  /SHEET_FEED qualifier • 7–1
DELETE command
  removing directory entries • 8–66
DELETE/ENTRY command
  mixed-version limitation • 4–4
DELTA/XDELTA
  using symmetric multiprocessing systems • 9–45
Device
  format for upgrade • 3–2
  HSC format for upgrade • 3–2
  requirement for upgrade • 2–6
Digital Command Language
  See DCL
Directories
  protecting • 8–66
  requirement for upgrade • 1–2
  user, processed during upgrade • 3–13

Disks
restriction during rolling upgrade • 4–10
Distribution volume
use during upgrade • 3–1
Dump files
modifying size after upgrade • 5–6
processed during upgrade • 1–3

E

EDT editor
delete access requirement • 7–3
EDTSECINI
mixed-version limitation • 4–4
/ENTRY qualifier • 7–3
Error
cluster hang during shutdown • 4–7
during console update • 5–5
during queue file conversion to Version 5.0 • 4–10
hang during queue file conversion • 4–13
incorrect value for INTSTKPAGES causing crash • 2–4
%JBCUPGRAD-E-JOBNOTSAVED • 4–11
%JBCUPGRAD-W-NOTSTOPQUE • 4–10
recovering from reboot failure (MicroVAX and VAXstation) • 3–12
recovering from TPU failure on a mixed-version cluster • 4–4
while entering device code • 3–2
EVE editor • 7–3 to 7–15
documentation errors • 7–15
future version incompatibilities • 7–9
previous version incompatibilities • 7–4
problems • 7–10
restrictions • 7–14
EXCHANGE procedure
updating console after upgrade • 5–5
EXE$ALOPHYCNTG • 10–2
EXE$ALTQUEPKT • 9–44
EXE$BUFFRQUOTA routine • 9–45
EXE$BUFQUOPRC routine • 9–45
EXE$DEBIT_BYTCNT_ALO • 10–2
EXE$DEBIT_BYTCNT_BYTLM_ALO • 10–2
/EXECUTIVE qualifier • 9–49
EXPECTED_VOTES parameter • 5–2
Extensible VAX Editor
See EVE editor

F

F$VERIFY lexical function
symbol substitution • 7–18
File Definition Language Facility (FDL)
processing files with comment lines containing semicolons • 7–17
File names
restriction on maximum character length • 7–16
Files
not recommended to be journaled • 8–37
protecting • 8–66
File types
restriction on maximum character length • 7–16
$FREE
RMS restriction • 9–11
Full duplex device driver
I/O completion for • 9–44, 10–2

G

$GETLKI system service
mixed-version limitation • 4–4

H

HELPLIB.HLB • 2–3
HSC
device format for upgrade • 3–2
requirement during Phase 1 of upgrade for 8200, 8250, 8300, 8350 • 3–7

I

I/O postprocessing
for full duplex device driver • 9–44, 10–2
queue • 9–44
I/O postprocessing queue • 10–2
I/O postprocessing queues • 9–43
I/O synchronization
using SMP • 9–41
IF command
nested levels • 7–1

INITIAL phase
  Startup phase • 8–8
Installation procedure
  for VAX 8200 • 8–1 to 8–5
  RK07 distribution kits • 8–5
Installation Verification Procedure (IVP)
  description • 8–32
Internal processor registers (IPRs)
  definition symbols • 9–50
Interrupt
  resuming after, (upgrade) • 1–4
  upgrade before Phase 1 • 3–3
INTSTKPAGES
  requirement for upgrade • 2–4
IOSB (I/O status block) • 9–44
IPL$_IOPOST • 9–44

J

JBC$UPGRADE • 4–12
JBCSYSQUE.DAT • 4–11
%JBCUPGRAD-E-JOBNOTSAVED message • 4–11
%JBCUPGRAD-W-NOTSTOPQUE message • 4–10
Job controller
  mixed-version limitation • 4–4
  restriction during rolling upgrade • 4–10

L

LADRIVER • 9–45
LAT • 9–45
LAT service ratings
  mixed-version limitation • 4–5
Layered products
  after upgrade • 5–7
  mixed-version limitation • 4–5
  restriction for upgrade • 1–4
LIB$
  changes • 9–15
  new translation tables • 9–15
LIB$ routine • 9–14
LIB$SYS_TRNLOG • 9–16
LIB$SYS_TRNLOG routine • 9–16
LIBRARY save sets • 3–13
License
  definition of an activity for a VMS • 6–30
  error messages • 6–3
License (cont'd.)
  examples of registration • 6–1, 6–7, 6–13, 6–19
  managing VMS licenses provided for service customers • 6–24
  provided with a VMS W-KIT • 6–24
  registering a System Integrated Product • 6–19
  registering a VMS activity • 6–13
  registering a VMS availability • 6–7
  registering a VMS license in a VAXcluster environment • 6–7
  registering on a standalone VAX • 6–13
  registration with LMF$CONFIG.COM • 6–24
  types for VMS • 6–29
LICENSE command • 6–31
LICENSE database
  common, with multiple system disks • 6–29
  special location for VMS service customers • 6–29
License Management Facility (LMF) • 6–1 to 6–33
  DECnet–VAX notes • 6–31
  information messages during upgrade • 3–3
  notes about the • 6–28
  upgrade requirement • 1–3, 5–1
  VAXcluster notes • 6–30
  VAX RMS Journaling notes • 6–31
  VAX Volume Shadowing notes • 6–31
License Management Utility (LICENSE) • 6–1 to 6–33
  codes for license types • 6–26
  License Unit Requirement Table (LURT) • 6–26
  modifying license units • 6–25
  MOD_UNITS option • 6–25
License Unit Requirement Table (LURT)
  used with the License Management Utility (LICENSE) • 6–26
LMF
  See License Management Facility
LMF$CONFIG.COM • 6–24
Local Area Terminal
  See LAT
Local Area VAXcluster, upgrade requirement • 1–3
Logical name • 9–16
Log in
  requirement for upgrade • 2–6
  restriction after upgrade • 1–3
LPA11–K driver • 9–45
LTDRIVER • 9–45

M

MACRO
  debugger support • 9–6
MAIL
  converting database, description • 4–11
  converting primary database to Version 5.0 • 4–12
  converting secondary database to Version 5.0 • 4–13
  database name change • 1–3, 4–3, 4–11
  database upgrade requirement • 1–3
  mixed-version limitation • 4–4
  SET CC_PROMPT command
    mixed-version limitation • 4–4
  SET FORM command
    mixed-version limitation • 4–4
  SET QUEUE command
    mixed-version limitation • 4–4
  specifying database file • 4–12
  terminating sessions • 4–11
MAIL$UPGRADE.COM • 4–13
Mandatory update
  automatic • 3–16
  LIBRARY saveset • 3–16
  manual • 5–1
  OPTIONAL saveset • 3–16
  RD52 systems • 3–16
Message Router
  installation restriction • 9–9
MicroVAX system
  Phase 1 upgrade procedure • 3–11 to 3–12
  recovering from reboot failure • 3–12
Mixed-interconnect configuration, upgrade requirement • 1–3
Mixed-version state A5.0
  description • 4–3
  restriction • 4–3
  upgrading to • 4–7 to 4–8
Mixed-version state B5.0
  converting to • 4–9 to 4–10
  restriction • 4–3
MODPARAMS.DAT
  modifying for cluster • 5–2
  modifying for single system • 5–2
  requirement for upgrade • 2–3
MSCP server
  mixed-cluster restriction • 4–4
MSCP_BUFFER
  change in value • 2–4
MSCP_BUFFER (cont'd.)
  requirement for upgrade • 2–4
MULTIPROCESSING parameter • 8–44

N

National Character Set
  See NCS • 9–19
NCP
  use for upgrade • 2–4
NCS
  diacriticals • 9–19
NETNODE_REMOTE.DAT • 5–3
NETOBJECT.DAT • 5–3
NETPROXY.DAT • 1–3, 4–3
NETUAF.DAT
  conversion • 3–15
Network
  requirement for upgrade • 2–4
  restriction on upgrade • 1–2
NISCS_CONV_BOOT parameter • 5–2
NISCS_LOAD_PEA0 parameter • 5–2
Nonpaged dynamic memory
  using conversational boot for insufficient (8530, 8550, 8700, 8800) • 3–9
  using conversational boot for insufficient (VAX–11/750) • 3–5
  using conversational boot for insufficient (VAX 8200, 8250, 8300, 8350) • 3–7
NOP instruction • 9–50
NOTICE.TXT
  processed during upgrade • 1–3

O

/OBJECT_TYPE qualifier • 7–3
OPEN command • 7–1
Operator
  requirement for VAX–11/750 upgrade • 3–6
  requirement for upgrade • 1–4
  requirement for VAX 8200, 8250, 8300, 8350 upgrade • 3–8
  requirement for VAX 8530, 8550, 8700, 8800 upgrade • 3–9
Operator logs
  processed during upgrade • 1–3
OPTIONAL save sets • 3–13
OTS$ routines • 9–17

P

PAGEFILE.SYS, see Paging file
/PAGES qualifier • 7–1
Paging file
  location requirement for upgrade • 2–5
  modifying size after upgrade • 5–6
  processed during upgrade • 1–3
  size required for upgrade • 2–2
Password
  modifying for upgrade, VAX–11/725, 11/730, 11/780, 11/785, 8600, 8650 • 3–10
  modifying for upgrade, VAX–11/750 • 3–4
  modifying for upgrade, MicroVAX and VAXstation systems • 3–11
  modifying for upgrade, VAX 8200, 8250, 8300, 8350 • 3–6
  modifying for upgrade, VAX 8530, 8550, 8700, 8800 • 3–8
PE3 parameter
  name change • 5–2
PE6 parameter
  name change • 5–2
PPL$CREATE_SHARED_MEMORY routine • 9–12
Pre-upgrade procedure • 2–1 to 2–6
Primary directory entries • 7–17
  protecting • 8–66
PRINT command
  /PAGES qualifier • 7–1
Private system disk format • 1–3
Product Authorization Key (PAK)
  registering • 6–7, 6–14, 6–19
Products (VMSINSTAL)
  specification for upgrade • 3–2
  specifying for mandatory update • 5–1
Proxy database for DECnet
  See DECnet proxy database

Q

Queue commands
  mixed-version limitation • 4–4
Queue file
  conversion description • 4–10
  converting primary to Version 5.0 • 4–11
  converting secondary to Version 5.0 • 4–12
  mixed-version limitation • 4–4
  specifying • 4–11
Queue file (cont'd.)
  upgrade requirement • 1–3
Queues
  requirement for upgrade • 2–4
QUORUM parameter
  setting for concurrent upgrade • 4–2
  setting for rolling upgrade • 4–8, 4–9
  superseded after upgrade • 5–2

R

RA60
  Phase 2 of upgrade • 3–13
RC25
  Phase 2 of upgrade • 3–13
  restriction on upgrade • 1–2
RD52
  restriction on upgrade • 1–2
  space required for upgrade • 2–3
  upgrade restriction • 3–12
READ
  /EXECUTIVE qualifier • 9–49
Record Management Services
  See RMS
Recoverable facility
  numbers associated with • 10–9
Recovery units
  modifying the recoverable facility for • 10–7
  specifying a facility for • 10–8
Registering license after upgrade • 5–1
RENAME command • 7–2
  removing directory entries • 8–66
Restart
  requirement for upgrade • 2–5
Restore console media for upgrade • 2–1
Restrictions, for upgrade • 1–2
RIGHTSLIST.DAT
  processed during upgrade • 1–3
RK07
  Phase 2 of upgrade • 3–13
  restriction on upgrade • 1–2
RK07 kits
  installing • 8–5
RL02
  Phase 2 of upgrade • 3–13
RMS • 9–11
  appending sequential files • 9–11
  block mode for copy operations • 9–11
  extended asynchronous interface • 9–11

RMS (cont'd.)
$FREE restriction • 9–11
task network operations • 9–11
XAB$V_NUL option • 9–11
Rolling upgrade
description • 4–1
procedure • 4–7 to 4–13
summary • 4–3
VMS version requirement • 1–2, 4–6
Root directory
for booting during upgrade • 2–6
RTL • 9–12
• 9–19
creating multiple program copies • 9–12
language support • 9–19
LIB$ADAWI routine • 9–13
LIB$ changes • 9–15
LIB$CREATE_VM_ZONE correction • 9–13
LIB$ routine • 9–14
LIB$SPAWN routine • 9–13
LIB$SYS_TRNLOG • 9–16
MTH$ routines • 9–16
new LIB$ translation tables • 9–15
obsolete SMG$ routines • 9–18
OTS$ routines • 9–17
PPL$CREATE_SHARED_MEMORY routine • 9–12
PPL$ENABLE_EVENT_SIGNAL restriction • 9–12
SMG$CREATE_PASTEBOARD restriction • 9–17
SMG$CREATE_VIRTUAL_KEYBOARD restriction • 9–17
SMG$ features • 9–17, 9–18
string procedures • 9–14
Run-Time Library
See RTL
/RU_ACTIVE qualifier
overview • 10–7
/RU_FACILITY qualifier
examples • 10–9
overview • 10–8
RX50
booting from during upgrade • 3–6
requirement for upgrade • 3–7

S

Satellites, upgrade requirement • 1–3
Screen Management Facility • 9–5
Section files
requirement for mixed-version cluster • 4–4
SET ACL command
/OBJECT_TYPE qualifier • 7–3
SET DEVICE command
/ACL qualifier • 7–3
SET DIRECTORY command
/ACL qualifier • 7–3
SET ENTRY command • 7–3
SET FILE
/REMOVE qualifier
removing directory entries • 8–66
SET FILE/AI_JOURNAL command
errors when creating duplicate journal files • 8–39
SET FILE/BI_JOURNAL command
errors when creating duplicate journal files • 8–39
SET FILE command
/ACL qualifier • 7–3
/RU_ACTIVE qualifier
examples • 10–8
overview • 10–7
/RU_facility qualifier
examples • 10–9
/RU_FACILITY qualifier
overview • 10–8
SET HOST/DTE/DIAL command • 9–51
SET IMAGE command • 9–7
SET PROCESS command
/CPU=[NO[ATTACHED qualifier • 7–2
SET QUEUE command
/ENTRY qualifier • 7–3
SET SCOPE command • 9–8
SET TIME/CLUSTER command • 7–2
SET TIME command • 7–2
/SHEET_FEED qualifier • 7–1
SHOW CLUSTER/CONTINUOUS
mixed-version limitation • 4–4
Show Cluster Utility
HW_TYPE field • 8–41
SHOW ENTRY command
mixed-version limitation • 4–4
SHOW EXECUTIVE command • 9–49
SHOW LICENSE command • 6–32
SHOW QUEUE command • 7–2
Single system upgrade
procedure • 3–1 to 3–16
SMG
See Screen Management Facility

SMG$
obsolete routines • 9–18
SMP
See Symmetric Multiprocessing
Space required for upgrade • 2–3
Standalone BACKUP for upgrade • 2–1
STARLET.OLB • 2–3
STARTUP.COM procedure • 8–7
Startup phases
BASEENVIRON phase • 8–8
CONFIGURE phase • 8–8
INITIAL phase • 8–8
SYSFILES phase • 8–8
Startup procedure • 8–7 to 8–10
STARTUP_P1 parameter
requirement for upgrade • 2–4
STOP/CPU command
entering while operating system is running • 8–50
Streamlined synchronization image
loading • 8–44
/SUMMARY qualifier • 7–2
Swapping file
modifying size after upgrade • 5–6
processed during upgrade • 1–3
use during upgrade • 2–2
SYENVIRON.COM procedure
obsolete • 8–10
Symbol names
assigning symbols that are DCL command names • 7–18
Symbol substitution
performing for F$VERIFY function • 7–18
Symmetric Multiprocessing
I/O synchronization • 9–41
Symmetric Multiprocessing (SMP) • 8–6
Synchronization image
streamlined • 8–44
uniprocessing • 8–44
SYS$STARTUP directory • 8–8
SYSDUMP.DMP, see Dump files
SYSERR directory, processed during upgrade • 1–3
SYSF
root directory • 2–6
SYSFILES phase
Startup phase • 8–8
SYSGEN
parameter requirement for upgrade • 2–4
SYSGEN parameter
requirement for upgrade • 2–3
SYSMAN • 8–44
CONFIGURATION command • 7–2
STARTUP subfunction • 8–9
SYSTARTUP.COM, see System startup command procedure
SYSTARTUP_V5.COM • 1–3, 5–2
SYSTARTUP_V5.COM procedure • 8–9
SYSTEM account, requirement for pre-upgrade • 2–1
System disk
analyze for upgrade • 2–2
backing up after upgrade • 5–7
backup before B5.0 • 4–9
backup before upgrade • 2–1
correct errors for upgrade • 2–2
free space required for upgrade • 2–3
restriction during upgrade • 1–3
System Management Utility
See SYSMAN
System manager's account, see SYSTEM
System startup command procedure
name change • 1–3, 5–2
SYSUAF.DAT
processed during upgrade • 1–3
restoring after upgrade • 5–1
upgrade requirement • 2–2, 2–5

T

Tailored system
after upgrade • 5–6
restriction for upgrade • 1–2
Tape device names
for MicroVAX computers • 8–5
for VAXserver 3600 computers • 8–5
for VAXstation computers • 8–5
TPU
error recovery on a mixed-version cluster • 4–4
logical names for mixed-version cluster • 4–4
mixed-version limitation • 4–4
TU58
booting from during upgrade • 3–4
requirement for Phase 1 upgrade, VAX–11 /725, 11/730, 11/780, 11/785, 8600, 8650 • 3–10
requirement for upgrade • 3–5

U

UETP
  refer to User Environment Test Package
Uniprocessing synchronization image
  loading • 8–44
Update, mandatory
  See Mandatory update
UPGGEN.CMD
  Phase 1 (VAX–11/725, 11/730, 11/780, 11/785, 8600, 8650) • 3–11
  Phase 1 (VAX–11/750) • 3–5
  Phase 1 (VAX 8200, 8250, 8300, 8350) • 3–7
  Phase 1 (VAX 8530, 8550, 8700, 8800) • 3–9
  Phase 1 for VAX–11/725, 11/730, 11/780, 11/785, 8600, 8650 • 3–11
  Phase 1 for VAX–11/750 • 3–5
  Phase 1 for VAX 8200, 8250, 8300, 8350 • 3–7
Upgrade procedure
  See also Cluster upgrade
  See also Concurrent upgrade
  See also Single system upgrade
  driver version error • 8–6
  overview • 1–1
  Phase 1 • 3–4
  Phase 2 • 3–12
  Phase 3 • 3–13
  Phase 4 • 3–13
  Phase 5 • 3–15
  Phase 6 • 3–15
  post-upgrade procedure • 5–1 to 5–8
  pre-upgrade procedure • 2–1
  summary • 1–2
User Environment Test Package • 5–7
User files
  processed during upgrade • 3–13

V

VAX–11/725 computer
  Phase 1 upgrade procedure • 3–10 to 3–11
VAX–11/730 computer
  Phase 1 upgrade procedure • 3–10 to 3–11
  recovery from boot failure • 3–11
VAX–11/750 computer
  Phase 1 upgrade procedure • 3–4 to 3–6
VAX–11/750 computer
  bootstrapping command procedure • 8–54
VAX–11/780 computer
  Phase 1 upgrade procedure • 3–10 to 3–11
VAX–11/782 computer
  discontinued support for ASMP • 8–6
VAX–11/785 computer
  Phase 1 upgrade procedure • 3–10 to 3–11
VAX 8200 computer
  Phase 1 upgrade procedure • 3–6 to 3–8
VAX 8250 computer
  Phase 1 upgrade procedure • 3–6 to 3–8
VAX 8300 computer
  Phase 1 upgrade procedure • 3–6 to 3–8
VAX 8350 computer
  Phase 1 upgrade procedure • 3–6 to 3–8
VAX 8530 computer
  Phase 1 upgrade procedure • 3–8 to 3–9
  requirement for pre-upgrade procedure • 2–6
  stopping the CPU • 8–50
  using the SET TIME/CLUSTER command • 7–2
  using the SET TIME command • 7–2
VAX 8550 computer
  Phase 1 upgrade procedure • 3–8 to 3–9
  requirement for pre-upgrade procedure • 2–6
  stopping the CPU • 8–50
  using the SET TIME/CLUSTER command • 7–2
  using the SET TIME command • 7–2
VAX 8600 computer
  Phase 1 upgrade procedure • 3–10 to 3–11
VAX 8650 computer
  Phase 1 upgrade procedure • 3–10 to 3–11
VAX 8670 computer
  nonexistence of • 8–56
VAX 8700 computer
  Phase 1 upgrade procedure • 3–8 to 3–9
  requirement for pre-upgrade procedure • 2–6
  stopping the CPU • 8–50
  using the SET TIME/CLUSTER command • 7–2
  using the SET TIME command • 7–2
VAX 8800 computer
  Phase 1 upgrade procedure • 3–8 to 3–9
  requirement for pre-upgrade procedure • 2–6
  stopping the CPU • 8–50
  using the SET TIME/CLUSTER command • 7–2
  using the SET TIME command • 7–2
VAX 8810 computer
  installation information • 8–56
VAX 8820 computer
  installing BUA board • 8–50
  stopping the CPU • 8–50

VAX 8820 computer (cont'd.)
using the SET TIME/CLUSTER command • 7–2
using the SET TIME command • 7–2
VAX 8820-N computer
installation information • 8–56
VAX 8830 computer
installing BUA board • 8–50
stopping the CPU • 8–50
using the SET TIME/CLUSTER command • 7–2
using the SET TIME command • 7–2
VAX 8840 computer
installing BUA board • 8–50
stopping the CPU • 8–50
using the SET TIME/CLUSTER command • 7–2
using the SET TIME command • 7–2
VAX Ada • 9–1 to 9–3
asynchronous file operations • 9–2
closing default files • 9–1
I/O files • 9–1
re-opening default files • 9–1
restrictions • 9–2
temporary files • 9–2
VAX BASIC • 9–3, 9–5
Run-Time Library • 9–3
VAX C
Run-Time Library changes • 9–20
VAXcluster configuration
upgrade requirement • 1–3
VAXcluster upgrade
See Concurrent upgrade
See Rolling upgrade
VAX DIBOL
new routines • 9–3
VAX MACRO • 9–20 to 9–23
restrictions • 9–21
VAX Pascal
installation restriction • 9–10
VAX RMS Journaling • 8–30
Installation Verification Procedure (IVP)
sample output • 8–32
VAXstation system
Phase 1 upgrade procedure • 3–11 to 3–12
recovering from reboot failure • 3–12
VAXTPU • 9–23 to 9–40
documentation notes • 9–27
errors • 9–25
future version incompatibilities • 9–26
previous version incompatibilities • 9–23
restrictions • 9–26
VAXVMSSYS.PAR parameter files
conversion • 3–15
Version 5.0, converting to
procedure • 4–11 to 4–13
VMB.EXE
update after upgrade for 8530, 8550, 8700, 8800 • 5–3
updating for VAX–11/750 • 3–4
updating for VAX 8200, 8250, 8300, 8350 • 3–7
VMS050 • 3–2
VMSIMAGES.DAT • 5–7
VMSINSTAL
invoke for mandatory update • 5–1
invoke for upgrade • 3–1
VMS Installation • 8–1 to 8–6
VMSLICENSE.COM
registering a System Integrated Product license with • 6–20
registering a VMS activity license with • 6–14
registering a VMS availability license with • 6–8
VMSMAIL.DAT • 4–12
name change • 4–3
VMSMAIL_PROFILE.DATA • 1–3, 4–3, 4–11
VMSTAILOR • 5–6
VMS Upgrade • 8–1 to 8–6
VMS version required, for upgrade • 1–2
VMS_ATOB050 • 4–9
VMS_BTOC050 • 4–11
Volume label names
requirement for rolling upgrade • 4–7
Volume-shadowed disks • 2–1
VOTES parameter
checking for rolling upgrade • 4–7
setting for concurrent upgrade • 4–2
upgrade requirement • 4–1

W

Workstation applications
linking on nonworkstation systems • 9–9
WRTJNL_BIJ error message
returns incorrect completion status value • 8–37

X

XAB$V_NUL option • 9–11

Reader's Comments

VMS Version 5.0
Release Notes
AA–LB22A–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 5.0
Release Notes
AA–LB22A–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 ----