MicroVAX 3100 Model 85, 90, 95, 96

KA50/51/55/56 CPU System Maintenance

Order Number: EK–M3100–SM. B01

June 1995

This manual gives maintenance information for systems that use the KA50, KA51, KA55 or KA56 CPU module.

Digital Equipment Corporation
Maynard, Massachusetts

First Printing, February 1995
Revised June, 1995

Digital Equipment Corporation makes no representations that the use of its products in the manner described in this publication will not infringe on existing or future patent rights, nor do the descriptions contained in this publication imply the granting of licenses to make, use, or sell equipment or software in accordance with the description.

© Digital Equipment Corporation 1995. All Rights Reserved.

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

The following are trademarks of Digital Equipment Corporation: DEC, Digital, MicroVAX, OpenVMS, VAX, VAXsimPLUS, and the DIGITAL logo.

All other trademarks and registered trademarks are the property of their respective holders.

S2869

This document was prepared using VAX DOCUMENT Version 2.1.

Contents

Preface xi

1 KA50/51/55/56 CPU Module Description

1.1 KA50/51/55/56 CPU Module 1–1
1.1.1 Physical Description 1–2
1.1.2 Functional Description 1–3
1.2 MS44 and MS44L Memory Modules 1–9
1.3 MS44 or MS44L Memory Option Installation 1–11
1.4 Memory Tests 1–13

2 Configuration

2.1 Memory Configurations 2–1
2.2 Mass Storage Devices 2–1
2.2.1 Internal Mass Storage Devices 2–2
2.2.2 External Mass Storage Devices 2–2
2.2.3 SCSI ID Numbers 2–4
2.3 Communications Options 2–4
2.3.1 Asynchronous Communications Options 2–4
2.3.2 Synchronous Communications Options 2–5

3 KA50/51/55/56 Firmware Commands

3.1 Console I/O Mode Control Characters 3–2
3.1.1 Command Syntax 3–3
3.1.2 Address Specifiers 3–3
3.1.3 Symbolic Addresses 3–4
3.1.4 Console Numeric Expression Radix Specifiers 3–8
3.1.5 Console Command Qualifiers 3–9
3.1.6 Console Command Keywords 3–10
3.2 Console Commands 3–13
3.2.1 BOOT 3–13

3.2.2 CONTINUE ... 3–15
3.2.3 DEPOSIT ... 3–15
3.2.4 EXAMINE ... 3–16
3.2.5 FIND ... 3–17
3.2.6 HALT ... 3–18
3.2.7 HELP ... 3–18
3.2.8 INITIALIZE ... 3–20
3.2.9 LOGIN ... 3–21
3.2.10 MOVE ... 3–22
3.2.11 NEXT ... 3–23
3.2.12 REPEAT ... 3–24
3.2.13 SEARCH ... 3–25
3.2.14 SET ... 3–27
3.2.15 SHOW ... 3–28
3.2.16 START ... 3–31
3.2.17 TEST ... 3–31
3.2.18 UNJAM ... 3–35
3.2.19 X—Binary Load and Unload ... 3–35
3.2.20 ! (Comment) ... 3–38

4 System Initialization and Acceptance Testing (Normal Operation)

4.1 Basic Initialization Flow ... 4–1
4.2 Power-On Self-Tests (POST) ... 4–2
4.2.1 Power-Up Tests for Kernel ... 4–3
4.2.2 Power-Up Tests for Mass Storage Devices ... 4–5
4.3 CPU ROM-Based Diagnostics ... 4–6
4.3.1 Diagnostic Tests ... 4–6
4.3.2 Scripts ... 4–11
4.4 Basic Acceptance Test Procedure ... 4–13
4.5 Machine State on Power-Up ... 4–14
4.6 Main Memory Layout and State ... 4–14
4.6.1 Reserved Main Memory ... 4–15
4.6.1.1 PFN Bitmap ... 4–15
4.6.1.2 Scatter/Gather Map ... 4–16
4.6.1.3 Firmware "Scratch Memory" ... 4–16
4.6.2 Contents of Main Memory ... 4–16
4.6.3 Memory Controller Registers ... 4–17
4.6.4 On-Chip and Backup Caches ... 4–17
4.6.5 Translation Buffer ... 4–17
4.6.6 Halt-Protected Space ... 4–17
4.7 Operating System Bootstrap ... 4–17

4.7.1 Preparing for the Bootstrap 4–18
4.7.2 Primary Bootstrap Procedures (VMB) 4–20
4.7.3 Device Dependent Secondary Bootstrap Procedures 4–23
4.7.3.1 Disk and Tape Bootstrap Procedure 4–23
4.7.3.2 MOP Ethernet Functions and Network Bootstrap Procedure 4–24
4.7.3.3 Network "Listening" 4–30
4.8 Operating System Restart 4–31
4.8.1 Locating the RPB 4–32

5 System Troubleshooting and Diagnostics

5.1 Basic Troubleshooting Flow 5–1
5.2 Product Fault Management and Symptom-Directed Diagnosis . . . 5–3
5.2.1 General Exception and Interrupt Handling 5–3
5.2.2 OpenVMS Error Handling 5–4
5.2.3 OpenVMS Error Logging and Event Log Entry Format 5–6
5.2.4 OpenVMS Event Record Translation 5–14
5.2.5 Interpreting CPU Faults Using ANALYZE/ERROR 5–15
5.2.6 Interpreting Memory Faults Using ANALYZE/ERROR 5–18
5.2.6.1 Uncorrectable ECC Errors 5–18
5.2.6.2 Correctable ECC Errors 5–22
5.2.7 Interpreting System Bus Faults Using ANALYZE/ERROR . . . 5–26
5.2.8 Interpreting DMA ⇔ Host Transaction Faults Using ANALYZE/ERROR 5–28
5.2.9 VAXsimPLUS and System-Initiated Call Logging (SICL) Support 5–32
5.2.9.1 Converting the SICL Service Request MEL File 5–37
5.2.9.2 VAXsimPLUS Installation Tips 5–38
5.2.9.3 VAXsimPLUS Post-Installation Tips 5–39
5.2.10 Repair Data for Returning FRUs 5–41
5.3 Power-On Self-Test (POST) and ROM-Based Diagnostic (RBD) Failures 5–41
5.3.1 FE Utility 5–47
5.3.2 Overriding Halt Protection 5–48
5.3.3 Isolating Memory Failures 5–48
5.4 Using MOP Ethernet Functions to Isolate Failures 5–53
5.5 Interpreting User Environmental Test Package (UETP) OpenVMS Failures 5–56
5.5.1 Interpreting UETP Output 5–56
5.5.1.1 UETP Log Files 5–56
5.5.1.2 Possible UETP Errors 5–57
5.6 Using Loopback Tests to Isolate Failures 5–58

5.6.1 Testing the Console Port 5–58
5.6.2 Embedded Ethernet Loopback Testing 5–58

6 FEPROM Firmware Update

6.1 Preparing the Processor for a FEPROM Update 6–2
6.2 Updating Firmware via Ethernet 6–3
6.3 Updating Firmware via Tape 6–6
6.4 FEPROM Update Error Messages 6–7

A Address Assignments

A.1 KA50/51/55/56 General Local Address Space Map A–1
A.2 KA50/51/55/56 Detailed Local Address Space Map A–3
A.3 External, Internal Processor Registers A–8
A.4 Global Q22–bus Address Space Map A–9
A.5 Processor Registers A–9
A.6 IPR Address Space Decoding A–21

B ROM Partitioning

B.1 Firmware EPROM Layout B–1
B.1.1 System Identification Registers B–3
B.1.1.1 PR$_SID (IPR 62) B–3
B.1.1.2 SIE (20040004) B–3
B.1.2 Call-Back Entry Points B–4
B.1.2.1 CP$GETCHAR_R4 ..... B–5
B.1.2.2 CP$MSG_OUT_NOLF_R4 ..... B–6
B.1.2.3 CP$READ_WTH_PRMPT_R4 ..... B–6
B.1.3 Boot Information Pointers B–7

C Data Structures and Memory Layout

C.1 Halt Dispatch State Machine C–1
C.2 Restart Parameter Block C–5
C.3 VMB Argument List C–9

D Configurable Machine State

E NVRAM Partitioning

E.1 SSC RAM Layout E–1
E.1.1 Public Data Structures E–2
E.1.1.1 Console Program MailBoX (CPMBX) E–2
E.1.1.2 Terminal Status E–3
E.1.1.3 Keyboard Status E–3
E.1.2 Service Vectors E–4
E.1.3 Firmware Stack E–4
E.1.4 Diagnostic State E–4
E.1.5 USER Area E–4

F MOP Counters

G Error Messages

G.1 Machine Check Register Dump G–1
G.2 Halt Code Messages G–1
G.3 VMB Error Messages G–3
G.4 Console Error Messages G–4

H Related Documents

Glossary

Index

Examples

1–1 Successful Running of Memory Test Script A8 1–13
1–2 Typical Failure After Running Memory Test Script A8 1–14
4–1 Successful Diagnostic Countdown 4–2
4–2 Successful Power-Up to List of Bootable Devices 4–5
4–3 Test 9E 4–8
5–1 Error Log Entry Indicating CPU Error 5–16
5–2 SHOW ERROR Display Using the OpenVMS Operating System 5–17

5–3	Error Log Entry Indicating Uncorrectable ECC Error	5–20
5–4	SHOW MEMORY Display Under the OpenVMS Operating System	5–21
5–5	Using ANALYZE/SYSTEM to Check the Physical Address in Memory for a Replaced Page	5–22
5–6	Error Log Entry Indicating Correctable ECC Error	5–25
5–7	Error Log Entry Indicating Q-Bus Error	5–27
5–8	Error Log Entry Indicating Polled Error	5–29
5–9	Device Attention Entry	5–31
5–10	SICL Service Request with Appended MEL File	5–38
5–11	Sample Output with Errors	5–41
5–12	FE Utility Example	5–48
5–13	Failure Due to a Missing SIMM (One 16 Mbyte Set)	5–49
5–14	Failure Due to a Missing SIMM (Two 16 Mbyte Sets)	5–50
5–15	Failure Due to a Bad SIMM	5–51
5–16	SIMM Wrong Size	5–52
6–1	FEPROM Update via Ethernet	6–5
6–2	FEPROM Update via Tape	6–7

Figures

1–1	KA50/51/55/56 CPU Module	1–2
1–2	KA50/51/55/56 CPU Module Block Diagram	1–4
1–3	KA50/51/55/56 Controls, Indicators, Ports, and Connectors	1–6
1–4	Memory Expansion Connectors	1–10
1–5	Memory Module Installation	1–12
2–1	SZ Expansion Box Numbering System	2–3
4–1	Console Banner	4–3
4–2	Memory Layout After Power-Up Diagnostics	4–15
4–3	Memory Layout Prior to VMB Entry	4–20
4–4	Memory Layout at VMB Exit	4–22
4–5	Boot Block Format	4–24
4–6	Locating the Restart Parameter Block	4–32
5–1	Event Log Entry Format	5–8
5–2	Machine Check Stack Frame Subpacket	5–9
5–3	Processor Register Subpacket	5–10

5–4	Memory Subpacket for ECC Memory Errors	5–11
5–5	Memory SBE Reduction Subpacket (Correctable Memory Errors)	5–11
5–6	CRD Entry Subpacket Header	5–12
5–7	Correctable Read Data (CRD) Entry	5–13
5–8	Trigger Flow for the VAXsimPLUS Monitor	5–34
5–9	Five-Level VAXsimPLUS Monitor Display	5–36
6–1	Firmware Update Utility Layout	6–2
6–2	W4 Jumper Setting for Updating Firmware	6–3
B–1	KA50/51/55/56 FEPROM Layout	B–2
B–2	SID : System Identification Register	B–3
B–3	SIE : System Identification Extension (20040004)	B–4
B–4	Boot Information Pointers	B–8
E–1	KA50/51/55/56 SSC NVRAM Layout	E–2
E–2	NVR0 (20140400) : Console Program MailBoX (CPMBX)	E–2
E–3	NVR1 (20140401)	E–3
E–4	NVR2 (20140402)	E–3

Tables

1–1	Functions of Controls, Indicators, Connectors	1–6
1–2	KA50/51/55/56 CPU Module Memory Configurations	1–10
2–1	KA50/51/55/56 Internal Mass Storage Devices	2–2
2–2	Supported Asynchronous Communications Options	2–4
2–3	Supported Synchronous Communications Options	2–5
2–4	DSW42-AA Communications Support	2–5
3–1	Console Symbolic Addresses	3–4
3–2	Symbolic Addresses Used in Any Address Space	3–8
3–3	Console Radix Specifiers	3–8
3–4	Console Command Qualifiers	3–9
3–5	Command Keywords by Type	3–11
3–6	Console Command Summary	3–11
4–1	LED Codes	4–4
4–2	Scripts Available to Customer Services	4–12
4–3	Network Maintenance Operations Summary	4–26
4–4	Supported MOP Messages	4–27
4–5	MOP Multicast Addresses and Protocol Specifiers	4–31

5–1	OpenVMS Error Handler Entry Types	5–7
5–2	Conditions That Trigger VAXsimPLUS Notification and Updating	5–33
5–3	Five-Level VAXsimPLUS Monitor Screen Displays	5–35
5–4	KA50/51/55/56 Console Displays as Pointers to FRUs	5–44
5–5	Loopback Connectors for Common Devices	5–60
A–1	Processor Registers	A–9
A–2	IPR Address Space Decoding	A–21
B–1	System Identification Register	B–3
B–2	System Identification Extension	B–4
B–3	Call-Back Entry Points	B–5
C–1	Firmware State Transition Table	C–3
C–2	Restart Parameter Block Fields	C–6
C–3	VMB Argument List	C–9
E–1	Bit Functions for NVR0	E–2
E–2	Bit Functions for NVR1	E–3
E–3	Bit Functions for NVR2	E–4
F–1	MOP Counter Block	F–1
G–1	HALT Messages	G–2
G–2	VMB Error Messages	G–3
G–3	Console Error Messages	G–4

Preface

This manual describes the KA50 CPU module used in the MicroVAX 3100 Model 90, the KA51 CPU module used in the MicroVAX 3100 Model 95, the KA55 CPU module used in the MicroVAX 3100 Model 85, and the KA56 CPU module used in the MicroVAX 3100 Model 96 system. It provides the configuration guidelines, ROM-based diagnostic information, and troubleshooting information for systems containing the KA50/51/55/56 CPU modules.

Audience

This manual is for Digital Services personnel who provide support and maintenance for systems that use the KA50/51/55/56 CPU module. It is also for customers who have a self-maintenance agreement with Digital Equipment Corporation.

Structure of This Manual

This manual is divided into six chapters, eight appendixes, a glossary, and an index:

- Chapter 1 describes the KA50/51/55/56 CPU module.
- Chapter 2 describes the KA50/51/55/56 system configurations.
- Chapter 3 describes the console commands that you can enter at the console prompt.
- Chapter 4 describes the system initialization, testing and bootstrap process that occurs at power-up.
- Chapter 5 describes the error log interpretation of diagnostic testing, the ROM-based diagnostic testing, and troubleshooting procedures for the KA50/51/55/56 systems. Also, this chapter provides information on testing DSSI storage devices, using MOP Ethernet functions to isolate errors, and interpreting UETP failures.
- Chapter 6 describes the FEPROM firmware.

- Appendix A gives the address assignments.
- Appendix B describes ROM partitioning and subroutine entry points.
- Appendix C gives definitions of the key global data structures used by the CPU firmware.
- Appendix D gives the normal state of all configurable bits in the CPU module as they are left after the successful completion of power-up ROM diagnostics.
- Appendix E describes how the CPU firmware partitions the SCC 1 KB battery-backed-up (BBU) RAM.
- Appendix F gives MOP counters.
- Appendix G describes the error codes and messages that the system exerciser test generates.
- Appendix H gives a list of related documents.

Note

Examples in this manual may vary slightly from your particular MicroVAX 3100 system, since they are from various VAX and MicroVAX systems which share common features, options, diagnostics, and so on.

Conventions

The following conventions are used in this manual:

Convention	Description
Ctrl/*x*	Ctrl/*x* indicates that you hold down the Ctrl key while you press another key or mouse button (indicated here by *x*).
x	A lowercase italic *x* indicates the generic use of a letter. For example, *xxx* indicates any combination of three alphabetic characters.
n	A lowercase italic *n* indicates the generic use of a number. For example, 19*nn* indicates a 4-digit number in which the last 2 digits are unknown.
{}	In format descriptions, braces indicate required elements. You must choose one of the elements.
[ ]	In format descriptions, brackets indicate optional elements. You can choose none, one, or all of the options.
( )	In format descriptions, parentheses delimit the parameter or argument list.
...	In format descriptions, horizontal ellipsis points indicate one of the following: • An item that is repeated • An omission such as additional optional arguments • Additional parameters, values, or other information that you can enter
|	In format descriptions, a vertical bar separates similar options, one of which you can choose.
italic type	Italic type emphasizes important information, indicates variables, and indicates the complete titles of manuals.
boldface type	Boldface type in examples indicates user input. Boldface type in text indicates the first instance of terms defined either in the text, in the glossary, or both.
n nnn.nnn nn	A space character separates groups of 3 digits in numerals with 5 or more digits. For example, *10 000* equals *ten thousand.*
n.nn	A period in numerals signals the decimal point indicator. For example, *1.75* equals *one and three-fourths.*
MONOSPACE	Text displayed on the screen is shown in monospace type.

Convention	Description
Radix indicators	The radix of a number is written as a word enclosed in parentheses, for example, 23(decimal) or 34(hexadecimal).
>>>	Three right angle brackets indicate the console prompt.
UPPERCASE	A word in uppercase indicates a command.
Note	A note contains information that is of special importance to the user.
Caution	A caution contains information to prevent damage to the equipment.
Warning	A warning contains information to prevent personal injury.

1

KA50/51/55/56 CPU Module Description

This chapter describes the KA50 **central processing unit** (CPU) module that is used in the MicroVAX 3100 Model 90, the KA51 CPU module that is used in the MicroVAX 3100 Model 95, the KA55 CPU module that is used in the MicroVAX 3100 Model 85 system, and the KA56 CPU module that is used in the MicroVAX 3100 Model 96. It gives information on the following:

- KA50/51/55/56 CPU modules
- MS44 or MS44L memory modules

The KA50, KA51, KA55 and KA56 are similar in design, and the information in this document is applicable for each of them except where noted. The differences between the KA50, KA51, KA55, and KA56 CPUs are as follows:

	KA50	KA51	KA55	KA56
Speed	286Mhz (14ns)	333Mhz (12ns)	250Mhz (16ns)	400Mhz (10ns)
VIC	2Kb	2Kb	disabled	2Kb
P-cache	8Kb	8Kb	8Kb	8Kb
B-cache	128Kb	512Kb	128Kb	512Kb

1.1 KA50/51/55/56 CPU Module

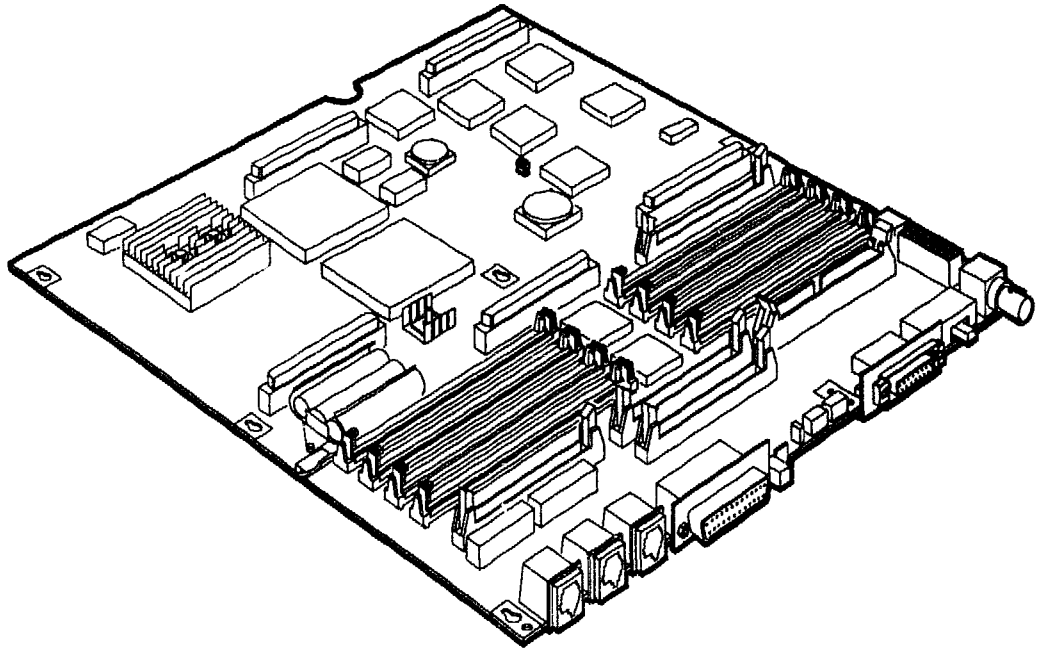
The KA50/51/55/56 CPU module is based on the NVAX chip set. It uses MS44 or MS44L memory modules and a set of supported **small computer system interface** (SCSI) devices. Figure 1–1 shows the KA50 CPU module; the KA51, KA55 and KA56 modules are similar.

1.1.1 Physical Description

The KA50/51/55/56 CPU module is the primary component of the MicroVAX 3100 system in which it is installed. The KA50/51/55/56 CPU module contains the following components:

- The NVAX processor chip—This chip is a complementary metal oxide semiconductor (CMOS) virtual memory microprocessor. The key features of the chip are as follows:
  - Support for the MicroVAX chip subset of the VAX instruction set
  - Support for the MicroVAX chip subset of the VAX data types
  - Full VAX memory management
  - 30-bit physical memory addressing

Figure 1–1 KA50/51/55/56 CPU Module



MLO-009931

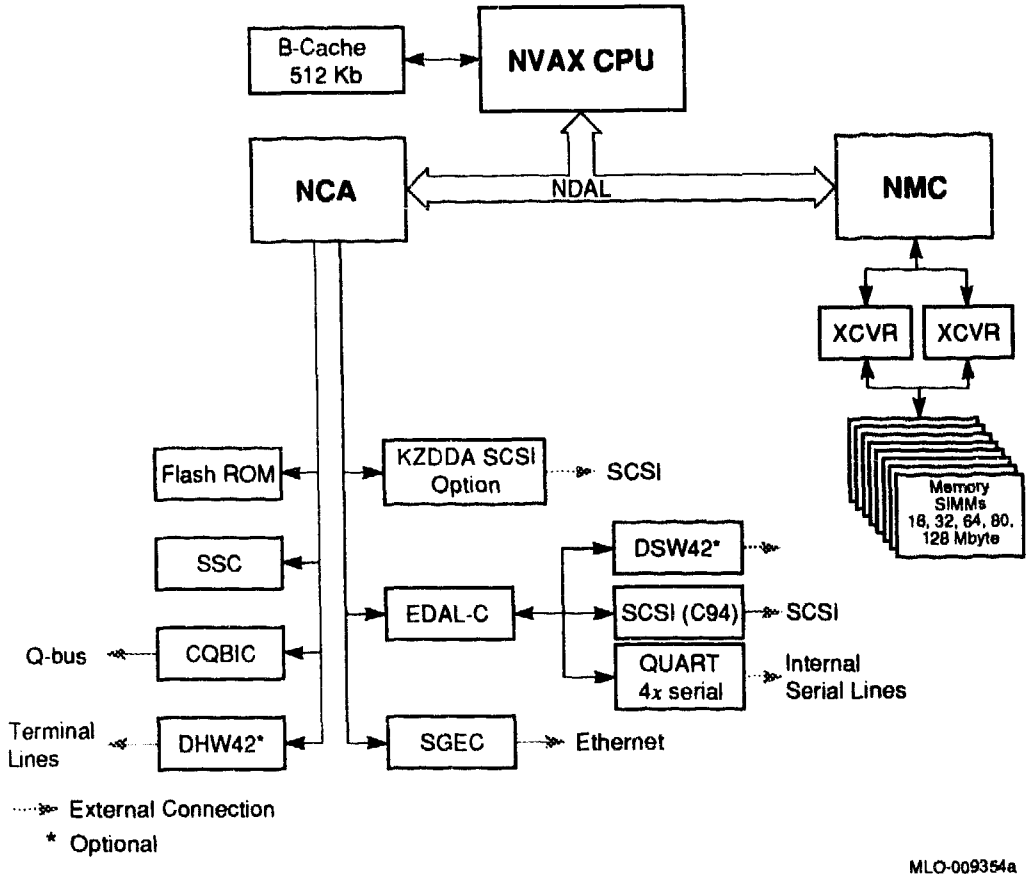
- DC244 NVAX memory controller (NMC) memory controller chip
- DC243 NVAX CP bus adapter (NMA) and input/output (I/O) control chip
- SCSI controller and SQWF buffer chip
- Time-of-year (TOY) clock SSC chip

- DC541 SGEC chip Ethernet controller for standard or ThinWire Ethernet
- DC7085 (QUART) serial line controller (4 serial lines, one with modem control)
- 128K bytes (KA50/55) or 512K bytes (KA51/56) of second level write-back cache memory
- Basic system memory (16M bytes of **random-access memory** (RAM) consisting of four MS44L-AA memory modules or 64M bytes of RAM consisting of four MS44-CA)
- Support for up to 128M bytes of RAM
- 512K bytes of **read-only memory** (ROM)—This ROM contains the boot and diagnostic firmware for the system.
- 32-byte network address ROM
- Four asynchronous communications ports as follows:
  - Three DEC423 ports—These ports are **modified modular jack** (MMJ) connectors.
  - One modem control port—This port is a D-sub 25-way connector.
- Provision for asynchronous communications options that provide one of the following:
  - Eight or 16 additional DEC423 ports
  - Eight additional modem ports
- Provision for synchronous communications options that provide:
  - Two synchronous ports

1.1.2 Functional Description

Figure 1–2 is block diagram of the CPU module. This example shows a KA51 and KA56. The diagrams for the KA50 and KA55 are the same except that there are only 128Kb of B-cache on those modules instead of the 512Kb shown.

Figure 1–2 KA50/51/55/56 CPU Module Block Diagram



The KA50/51/55/56 CPU module supports the following MicroVAX data types:

- Byte, word, longword, and quadword
- Character string
- Variable-length bit field
- Absolute queues
- Self-relative queues
- f_floating-point, d_floating-point, and g_floating-point

The operating system uses software emulation to support other MicroVAX data types. The KA50/51/55/56 CPU module supports the following MicroVAX instructions:

- Integer, arithmetic and logical
- Address
- Variable-length bit field
- Control
- Procedure call
- Miscellaneous
- Queue
- Character string instructions
- MOVC3/MOVC5
- CMPC3/CMPC5
- LOCC
- SCANC
- SKPC
- SPANC
- Operating system support
- f_floating-point, d_floating-point, and g_floating-point

The NVAX processor chip provides special microcode assistance to aid the macrocode emulation of the following instruction groups:

- Character string (other than those mentioned previously)
- Decimal string
- CRC
- EDITPC

The operating system uses software emulation to support other VAX instructions. Figure 1–3 shows the controls, indicators, ports, and connectors on the KA50/51/55/56 CPU module. Table 1–1 describes the functions of the controls, indicators, ports, and connectors.

Figure 1–3 KA50/51/55/56 Controls, Indicators, Ports, and Connectors

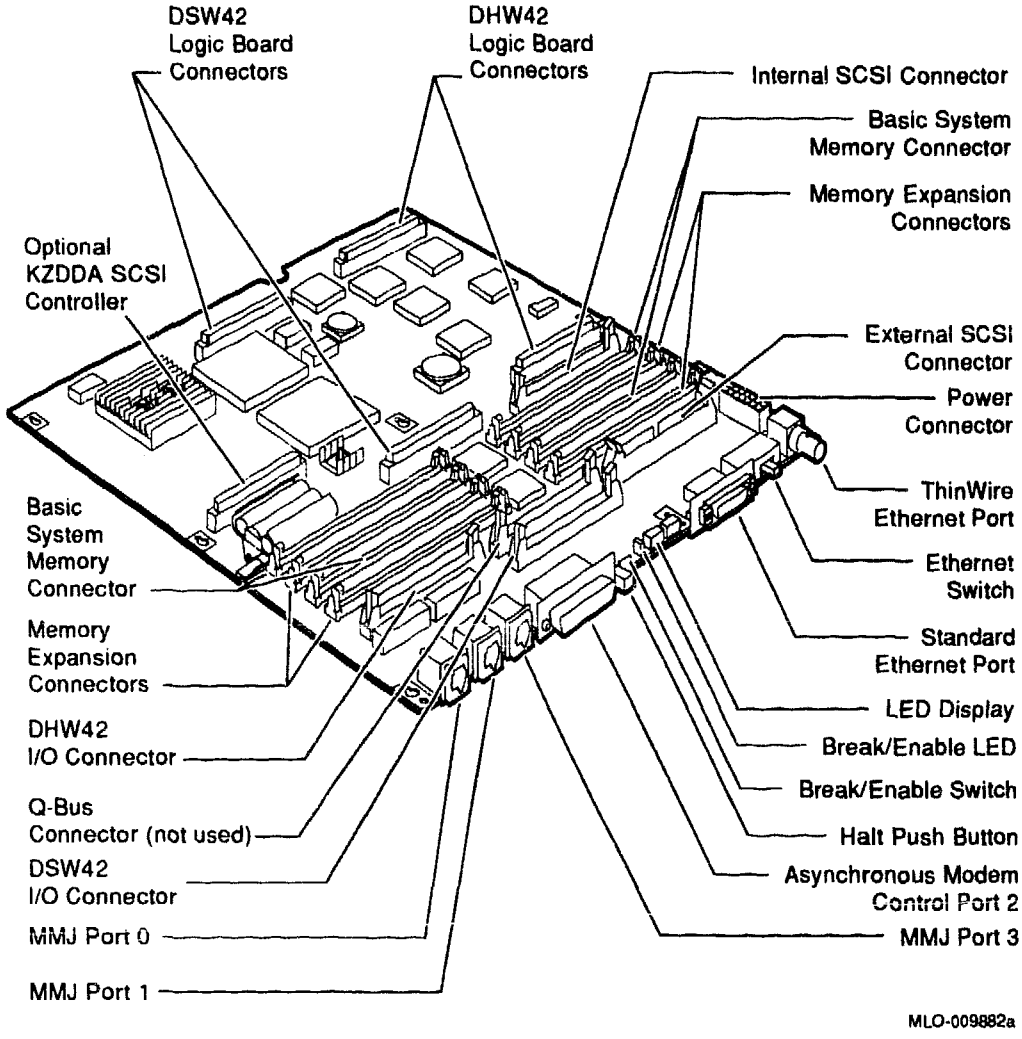


Table 1–1 Functions of Controls, Indicators, Connectors

Component	Description
Internal SCSI connector	A connector that provides a connection for SCSI devices mounted inside the system enclosure.

(continued on next page)

Table 1–1 (Cont.) Functions of Controls, Indicators, Connectors

Component	Description
Basic system memory connectors	Four connectors for the basic system memory modules.
Memory expansion connectors	Four connectors for an additional memory option.
External SCSI connector	A connector that provides a connection to SCSI devices that are external to the system enclosure. (Only functional when the *internal* SCSI connector has a cable installed.)
Power connector	A connector for dc power.
ThinWire Ethernet port	A port that provides a connection to a ThinWire Ethernet network.
Ethernet switch	A two-position switch that determines the type of Ethernet that the system uses as follows: • Left position—selects the standard Ethernet type • Right position—selects the ThinWire Ethernet type
Standard Ethernet port	A port that provides a connection to a standard Ethernet network.
LED display	A set of six LEDs that provide power-up and self-test diagnostic code information.
Break/Enable LED	A LED indicator that shows the function of MMJ port 3 as follows: • On—Break enable • Off—Break disable on port 3
Break/Enable switch[1]	A two-position switch that determines the function of MMJ port 3 as follows: • Up position—MMJ port 3 functions as a console port; in this state, you can press the Break key on the keyboard of a terminal connected to MMJ port 3 to put the system in console mode. • Down position—MMJ port 3 functions as a console port only, and the Break key is disabled.

[1]The system recognizes the position of this switch only when the system is turned on.

(continued on next page)

Table 1–1 (Cont.) Functions of Controls, Indicators, Connectors

Component	Description
Halt button	A momentary-contact push button that puts the system in console mode.
Asynchronous modem control port 2	EIA-232 compatible asynchronous port with modem control.
MMJ port 3	DEC423 compatible asynchronous port. This port functions as the primary console port.
MMJ port 1	DEC423 compatible asynchronous port.
MMJ port 0	DEC423 compatible asynchronous port.
DSW42 I/O connector	A connector that provides a connection for the DSW42 input/output cable.
DHW42 I/O connector	A connector that provides a connection for the DHW42 input/output cable.
DSW42 logic board connectors	Two connectors that provide connections for a DSW42 logic board.
DHW42 logic board connectors	Two connectors that provide connections for a DHW42 logic board.
KZDDA SCSI connector option	Connector which provides a physical interface between the CPU module and external SCSI devices on an optional second SCSI bus (SCSI-B).

1.2 MS44 and MS44L Memory Modules

The MS44 and the MS44L memory modules provide memory expansion for the KA50/51/55/56 CPU module. The KA50/51/55/56 CPU module supports one variant of the MS44 memory option and one variant of the MS44L option as follows:

- The MS44L-BC (16M bytes), which contains four MS44L-AA (4M bytes) memory modules
- The MS44-DC (64M bytes), which contains four MS44-CA (16M bytes) memory modules

Note

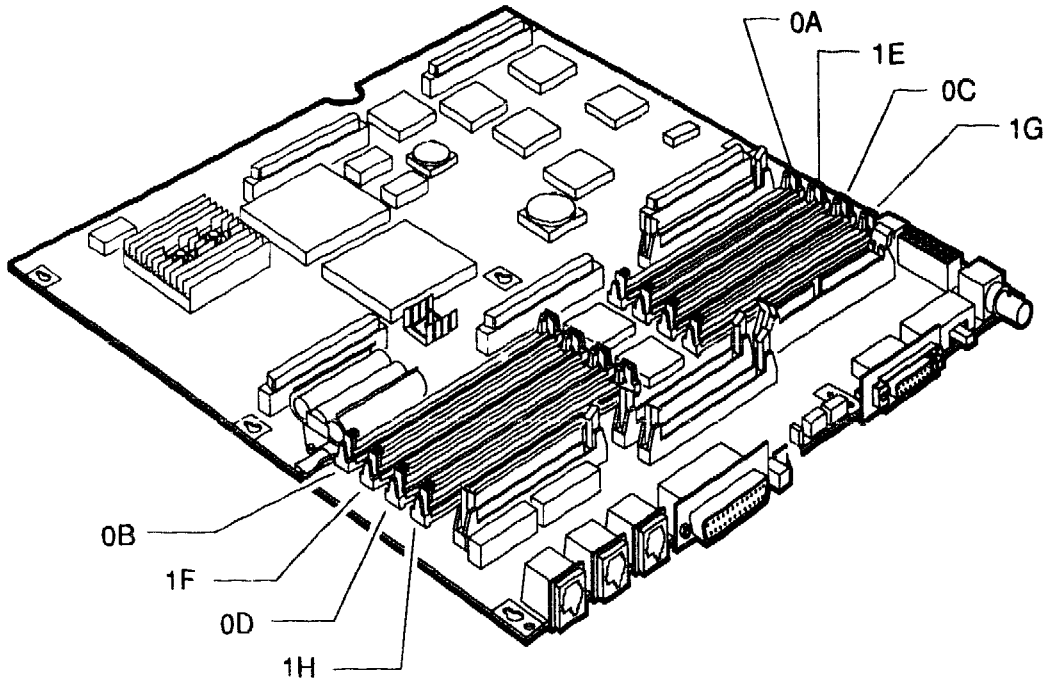
Use only MS44 or MS44L memory modules qualified by Digital.

The rules for adding MS44 or MS44L memory options are as follows:

- You must install all four of the memory modules contained in a memory option. This means that you can expand memory in 16M byte or 64M byte increments only.
- You can install memory options only in a set of connectors that have the same numeral in the connector label. The sets are identified by the following labels:
  - 0A, 0B, 0C, 0D
  - 1E, 1F, 1G, 1H

Figure 1–4 shows the location of the basic memory (16M bytes or 64M bytes) and the memory expansion connectors. Table 1–2 lists the memory configurations.

Figure 1–4 Memory Expansion Connectors



Note: 0A 0B 0C and 0D are identifiers for the basic system memory connectors.

GA_EN00083A_92A

Table 1–2 KA50/51/55/56 CPU Module Memory Configurations

Total Memory (bytes)	Increment 1[1] (0A + 0B + 0C + 0D)[2]	Increment 2 (1E + 1F + 1G + 1H)[2]
16M	MS44L-BC	
32M	MS44L-BC	MS44L-BC
64M	MS44-DC	
80M	MS44-DC	MS44L-BC
128M	MS44-DC	MS44-DC

[1]Basic system memory.

[2]0A, 0B, 0C, 0D, 1E, 1F, 1G, and 1H are connector identifiers (see Figure 1–4).

1.3 MS44 or MS44L Memory Option Installation

The MS44 and MS44L memory options consist of four memory modules each. Install an MS44 or MS44L memory option on the KA50/51/55/56 CPU module as follows:

1. Position the KA50/51/55/56 CPU module, component side up, so that the edge connectors are facing away from you.
2. Identify the connectors on the KA50/51/55/56 CPU module into which you must install the memory option (see Figure 1–4 and Table 1–2).
3. Insert the first memory module, with the side containing the bar code facing away from you, into the connector on the KA50/51/55/56 CPU module (see Figure 1–5).

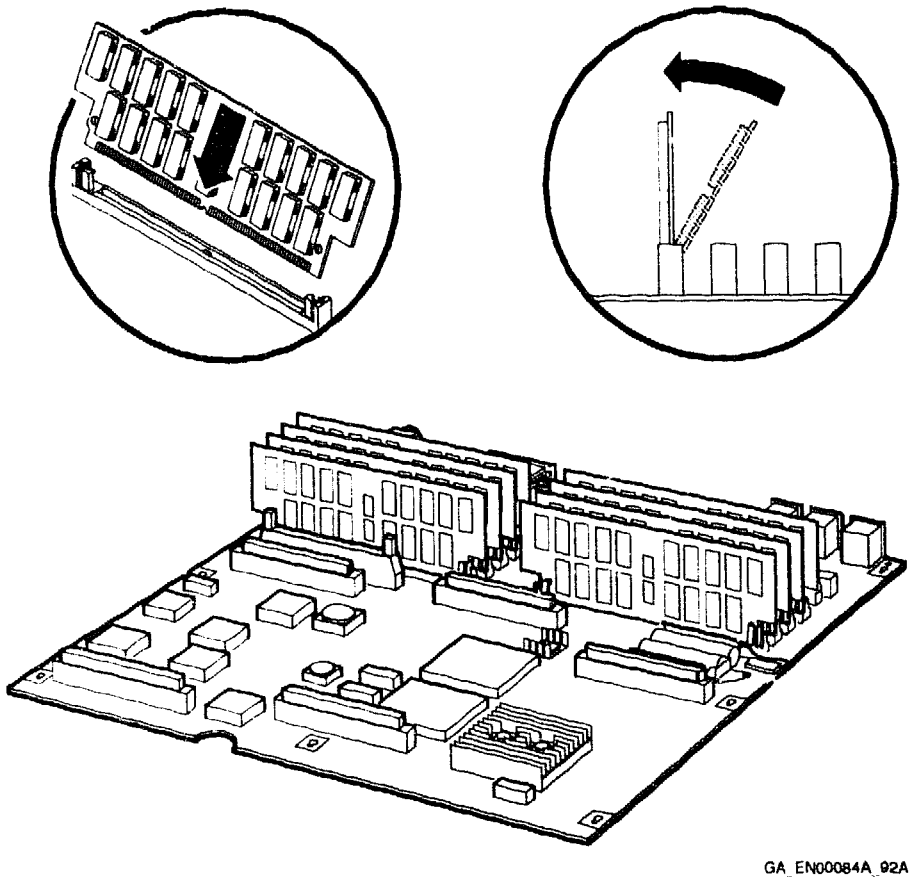
Caution

The connectors are keyed to ensure that you install the memory modules with the correct orientation. Do not force the modules into the connectors with an incorrect orientation.

Caution

Make sure that you fully install the memory module into the connector before you tilt the module toward the front of the enclosure.

Figure 1–5 Memory Module Installation



4. Tilt the memory module toward the front of the enclosure until the metal locking clips on the connector lock the memory module in position.
5. Repeat the procedure in step 1 for the subsequent memory modules. Insert them into the other connectors in the set on the KA50/51/55/56 CPU module.
6. Run the MEM diagnostic test, refer to Section 1.4 after you reinstall the KA50/51/55/56 CPU module into the system enclosure to check that the memory is working correctly.

Caution

When removing memory modules, you must release the metal clips on the connectors of the CPU module.

1.4 Memory Tests

The memory tests check the system memory contained on the MS44 and/or MS44L memories. The tests run automatically as part of the power-up tests and initialization, when you turn on the system. The memory tests are a group of individual tests which can be called individually or normally as a group under a specific script number.

The recommended method to verify a new memory installation is to run the memory test script A8 which will call all of the memory tests and run them on all memory present.

Examples of successful and unsuccessful runs of memory test script A8 are shown in Example 1–1 and Example 1–2.

The individual memory tests are listed following the examples.

Example 1–1 Successful Running of Memory Test Script A8

```
>>>T A8
9D..31..30..4F..4E..4D..4C..4B..4A..48..48..48..48..48..48..48..
48..48..48..47..40..80..
>>>
```

The failure is reported by the count bad pages test 40 at end of the script. Issuing the SHOW MEMORY command shows which memory set caused the failure. Bad pages were detected in memory set 0.

```
>>>SHOW MEMORY

16 MB RAM, SIMM Set (0A,0B,0C,0D) present
Memory Set 0: 00000000 to 00FFFFFF, 16MB, 32256 good pages, 512 bad pages

16 MB RAM, SIMM Set (1E,1F,1G,1H) present
Memory Set 1: 01000000 to 01FFFFFF, 16MB, 32768 good pages, 0 bad pages

Total of 32MB, 65024 good pages, 512 bad pages, 112 reserved pages
>>>
```

Example 1–2 Typical Failure After Running Memory Test Script A8

```
>>>T A8
9D..31..30..4F..4E..4D..4C..4B..4A..48..48..48..48..48..48..48..
48..48..48..47..40..

? Test_Subtest_40_06  Loop_Subtest=00  Err_Type=FF  DE_Memory_count_pages.lis
Vec=0000  Prev_Errs=0004  P1=00000001  P2=00000002  P3=00000001
P4=00000000 P5=00000020  P6=00008000  P7=00000020  P8=00000000
P9=00000000 P10=00FCD44B r0=00FF4008  r1=00000007  r2=00000000
r3=FFFFFFFF  r4=00000068  r5=00000000 r6=00000000  r7=00000002
r8=00FF4000  r9=20140758  r10=FFFFFFFE r11=FFFFFFFF
dser=0000 cesr=00000000 intmsk=00 icsr=01 pcsts=FC00 pcadr=FFFFFFF8 pcctl=FC13
cctl=00000021 bcetsts=0000 bcedsts=0000 cefsts=00000200 nests=00
mmcdsr=01111000 mesr=00080000
```

Test DC - Check for No Memory Present

The only purpose of this test is to check for the specific condition of no valid memory present in the system. This occurs if no memory is present, or if memory is present and one or more SIMMs is missing or not plugged in correctly.

Test 31 - Size and Setup Memory CSRs

Find out how much memory is available and configure into consecutive memory starting at address 00000000. Verify proper configuration data in the CSRs.

Test 30 - Build a Bitmap in Memory

Set up a bitmap in RAM to be used by the memory tests. Test the area before setting up a bitmap.

This test looks for a 1 MB KB section of memory to be used for the bitmap, busmap and reserved console area and structures to run diagnostics. The test starts at the top of available memory and tests one section of memory at the top of each 4 MB section of memory until a good section is found for the maps or the bottom of memory is reached, in which case the test fails.

Test 4F - Data Pattern Tests

Verifies that each bit in the data path can be written to a one and a zero individually. This test also checks for shorts between individual paths. The test needs to be run once for each array of memory chips.

This test uses various fix patterns and also floating 1's and 0's patterns across all 72 data bits (64 data, 8 ECC). The test always checks both even and odd QWs of data so that all four SIMMs in a memory set are tested.

Test 4E - Masked Write Cycles with No Errors, BYTE, WORD

This test verifies masked write cycles to memory.

Test 4D - Address Uniqueness Test

The main purpose of the test is to verify that each set on each board can be uniquely addressed. The test writes a unique pattern to each location to be tested then verifies all locations.

Test 4C - MEMORY ECC, Verify Error Detection and Reporting

The main purpose of this test is to test ECC logic. It is not intended to test the memory RAMs explicitly.

The test verifies that single and double bit errors are reported and logged correctly in the MESR. It also verifies that single bit errors cause interrupts through vector 54 when enabled and that double bit errors cause a machine check.

In addition, the test also verifies that multiple bit errors can be detected using data patterns that generate all of the syndrome values for multiple bit errors.

Test 4B - MEMORY Verify Masked Write Cycles with Errors

The test verifies operation of masked write cycles when the location contains errors. In addition, it verifies that errors are reported and that single bit errors are corrected.

Test 4A - MEMORY ECC, Verify Ability to Correct Single Bit Errors

This test verifies the correct operation of the error correction logic (ECC). It does this by verifying that single bit errors can be detected and corrected in any of the 64 data bits and that single bit errors are detected in the eight check bits.

Test 48 - MEMORY Address/Shorts Test

This test verifies that all locations in each set can be uniquely written to and that each of the 64 data bits in each QW can be written to a one and to a zero. This test also writes all locations in memory with good ECC.

The test runs on a hexaword basis with all caches enabled to fully utilize caching to speed up the test. Two primary data patterns of AAAAAAAA_AAAAAAAA and 55555555_55555555 are used by the test. The ECC checkbits for these patterns are complements of each other. By running this test, all data and ECC bits in all locations in memory will be written as a 1 and a 0. The test also detects addressing errors.

Test 47 - MEMORY Data Retention, Verify Refresh Logic

This test verifies that the refresh logic is working for all memory boards. The test loads patterns into memory, waits a specified amount of time, then verifies the patterns.

Test 40 - MEMORY Count Bad Pages Marked In Bitmap

This test is normally run last in a script of memory tests. Its only purpose is to read the bitmap when done and check to see if any pages in memory were marked bad, if so, report an error.

Note

If this test fails, do SHOW MEMORY to see which set has bad pages in it.

2
Configuration

This chapter describes the KA50/51/55/56 system configurations. It gives information on the following:

- Memory configurations
- Mass storage devices
- Communications options

2.1 Memory Configurations

A KA50/51/55/56 system has a basic memory of 16M bytes or 64M bytes. This consists of four MS44L-AA memory modules or four MS44-CA memory modules. You can add memory in 16M byte or 64M byte increments, up to a maximum of 128M bytes. See Section 1.2 for information on the memory configurations.

2.2 Mass Storage Devices

A KA50/51/55/56 system supports mass storage devices in the following categories:

- Internal mass storage devices—These devices are mounted inside the system enclosure.
- External mass storage devices—These devices are self-contained units that you can connect to the system externally.

2.2.1 Internal Mass Storage Devices

Table 2–1 shows some of the internal mass storage devices that a KA50/51/55 /56 system supports.

Table 2–1 KA50/51/55/56 Internal Mass Storage Devices

Option Name	Description	Size[1] (in)	Capacity
RZ23L	Disk drive	3.5	120-MB
RZ24	Disk drive	3.5	209-MB
RZ24L	Disk drive	3.5	245-MB
RZ25	Disk drive	3.5	400-MB
RZ25L	Disk drive	3.5	535-MB
RZ25M	Disk drive	3.5	545-MB
RZ26	Disk drive	3.5	1.05-GB
RZ26L	Disk drive	3.5	1.05-GB
RZ28	Disk drive	3.5	2.10-GB
TZ30[2]	Tape drive	5.25	95-MB cartridge
TZK10/TZK11[2]	Tape drive	5.25	Range of cartridges
TLZ06/TLZ07[2]	Tape drive	5.25	Range of cassettes
RX23/RX26[2]	Diskette drive	3.5	Range of diskettes
RRD42[2]	CDROM drive	5.25	600-MB CDROM
RRD43[2]	CDROM drive	5.25	600-MB CDROM

[1]Size of half-height device.

[2]Removable media device.

The system enclosure determines the combinations of internal mass storage devices in a KA50/51/55/56 system. See the *MicroVAX 3100 BA42B Enclosure Maintenance* manual for more information.

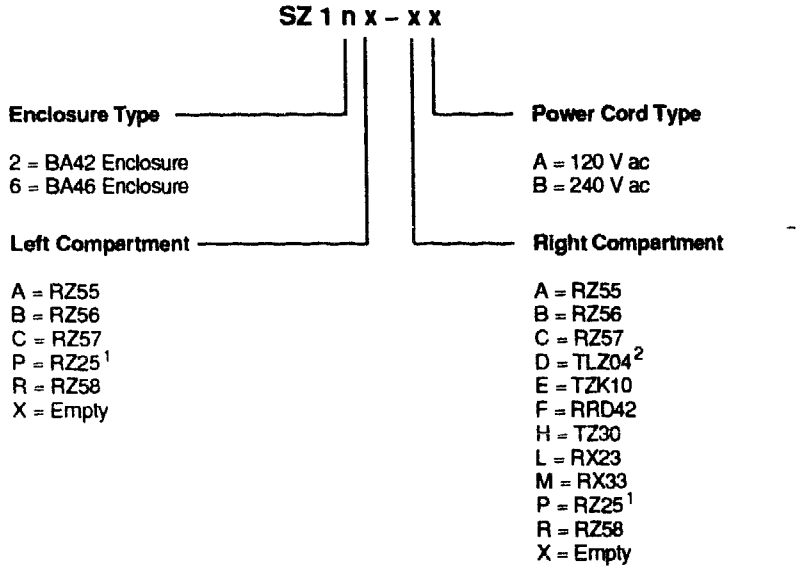
2.2.2 External Mass Storage Devices

The external mass storage devices connect to KA50/51/55/56 systems through the SCSI connector on the back of the system enclosure. In KA50/51/55/56 systems, the SCSI bus supports a maximum of seven mass storage devices. Therefore, the number of external mass storage devices that you can connect depends on the number of mass storage devices that are mounted inside the system enclosure.

The maximum number of mass storage devices in the system enclosure is five. This means that you can connect at least two external mass storage devices.

A KA50/51/55/56 system supports the SZ series of mass storage expansion boxes. The SZ number defines the contents of each expansion box. Figure 2–1 shows the numbering system for SZ expansion boxes.

Figure 2–1 SZ Expansion Box Numbering System



[1] The RZ25 disk drive fits in the BA42 enclosure only.
[2] The TLZ04 tape drive fits in the BA46 enclosure only.

With the KZDDA SCSI option, a second SCSI connector, a KA50/51/55/56 system can support seven additional external devices on a second (external) SCSI bus.

A KA50/51/55/56 system also supports other types of external mass storage devices. See the latest Systems and Options Catalog (SOC) for a listing of supported external SCSI devices.

When you are adding mass storage devices, use these guidelines. Also, refer to documentation for your SCSI expander, if any.

- You can add a maximum of four external SCSI devices. A fully configured SZ12 enclosure contains two SCSI devices.
- You can add a maximum of two SCSI tape devices. Depending on the configuration, the system may support two TLZ04 tape drives.

- The BA40 single drive expansion box contains one SCSI device.
- The RRD42 CDROM drive is a single SCSI device. You can add a maximum of three RRD42 CDROM drives.
- Terminate the SCSI bus correctly. Failure to do this can cause a system failure or corrupt data.
- Digital recommends that you connect all SCSI devices to the same ac power source.
- Do not add or remove devices that are connected to the SCSI bus while the power is on.
- Digital does not guarantee the correct operation of a SCSI bus that does not use the cables supplied by Digital or is not configured in accordance with Digital recommendations.

2.2.3 SCSI ID Numbers

Each mass storage device must have a unique SCSI ID number. SCSI ID 6 is typically used for the SCSI controller.

2.3 Communications Options

A KA50/51/55/56 system supports the following types of communications options:

- Asynchronous communications options
- Synchronous communications options

Each communications option has components that are installed in the system enclosure and components that connect to the system externally.

2.3.1 Asynchronous Communications Options

Table 2–2 lists the asynchronous communications options that KA50/51/55/56 systems support.

Table 2–2 Supported Asynchronous Communications Options

Option	Description
DHW42-AA	Eight-line DEC423 asynchronous option
DHW42-BA	Sixteen-line DEC423 asynchronous module option
DHW42-CA	Eight-line EIA-232 modem asynchronous module option
DHW42-UP	Eight-line to 16-line DEC423 asynchronous upgrade option

2.3.2 Synchronous Communications Options

Table 2–3 lists the synchronous communications options that KA50/51/55/56 systems support.

Table 2–3 Supported Synchronous Communications Options

Option	Description
Model 100	
DSW42-AA[1]	Two-line EIA-232/V.24 synchronous option with two external cables, BC19D-02 (17-01110-01)

[1]This option is supplied with two external cables that support the EIA-232/V.24 interface.

The the DSW42-AA option also supports the communications interfaces listed in Table 2–4, but you must order the external cables separately.

Table 2–4 DSW42-AA Communications Support

Communications Interface	External Cable
EIA-423/V.10	BC19E-02[1] (17-01111-01)
EIA-422/V.11	BC19B-02[1] (17-01108-01)

[1]Two required for DSW42-AA.

3

KA50/51/55/56 Firmware Commands

This chapt r describes the console mode control characters, the command syntax, the command modifiers, and all of the console commands. You can enter these commands when the system is in console mode. Console mode is indicated when the console prompt (>>>) is displayed. If the system is running the operating system software, refer to the *MicroVAX 3100 Model 85 Customer Technical Information* manual, the *MicroVAX 3100 Model 90 Customer Technical Information* manual, the *MicroVAX 3100 Model 95 Customer Technical Information* manual, or the *MicroVAX 3100 Model 96 Customer Technical Information* manual, for information on returning the system to console mode.

If the console security feature is enabled and a security password is set, you must log in to privileged console mode before using most of these commands. Refer to the appropriate MicroVAX 3100 Customer Technical Information manual (above) for information on the console security feature.

Note

The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95, and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, will appear on the console and/or printouts from time to time.

3.1 Console I/O Mode Control Characters

In console I/O mode, several characters have special meaning:

RETURN	Also <CR>. The carriage return ends a command line. No action is taken on a command until after it is terminated by a carriage return. A null line terminated by a carriage return is treated as a valid, null command. No action is taken, and the console prompts for input. Carriage return is echoed as carriage return, line feed (<CR><LF>).
<X	When you press <X, the console deletes the previously typed character. The resulting display differs, depending on whether the console is a video or a hardcopy terminal. For hardcopy terminals, the console echoes a backslash (\), followed by the deletion of the character. If you press additional rubouts, the additional deleted characters are echoed. If you type a nonrubout character, the console echoes another backslash, followed by the character typed. The result is to echo the characters deleted, surrounding them with backslashes. For example: EXAMI;E <X<XNE<CR> The console echoes: EXAMI;E\E;\NE<CR> The console sees the command line: EXAMINE<CR> For video terminals, the previous character is erased and the cursor is restored to its previous position. The console does not delete characters past the beginning of a command line. If you press more rubouts than there are characters on the line, the extra rubouts are ignored. A rubout entered on a blank line is ignored.
CTRL/A and F14	Toggle insertion/overstrike mode for command line editing. By default, the console powers up to overstrike mode.
CTRL/B or up_ arrow (or down_ arrow)	Recalls previous command(s). Command recall is only operable if sufficient memory is available. This function may then be enabled and disabled using the SET RECALL command.
CTRL/D and left arrow	Move cursor left one position.
CTRL/E	Moves cursor to the end of the line.
CTRL/F and right arrow	Move cursor right one position.
CTRL/H, backspace, and F12	Move cursor to the beginning of the line.
CTRL/U	Echoes ^U<CR> and deletes the entire line. Entered but otherwise ignored if typed on an empty line.
CTRL/S	Stops output to the console terminal until CTRL/Q is typed Not echoed.
CTRL/Q	Resumes output to the console terminal. Not echoed.

CTRL/R	Echoes <CR><LF>, followed by the current command line. Can be used to improve the readability of a command line that has been heavily edited.
CTRL/C	Echoes ^C<CR> and aborts processing of a command. When entered as part of a command line, deletes the line.
CTRL/O	Ignores transmissions to the console terminal until the next CTRL/O is entered. Echoes ^O when disabling output, not echoed when it re-enables output. Output is re-enabled if the console prints an error message, or if it prompts for a command from the terminal. Output is also enabled by entering console I/O mode, by pressing the BREAK key, and by pressing CTRL/C.

3.1.1 Command Syntax

The console accepts commands up to 80 characters long. Longer commands produce error messages. The character count does not include rubouts, rubbed-out characters, or the RETURN at the end of the command.

You can abbreviate a command by entering only as many characters as are required to make the command unique. Most commands can be recognized from their first character. See Table 3–5.

The console treats two or more consecutive spaces and tabs as a single space. Leading and trailing spaces and tabs are ignored. You can place command qualifiers after the command keyword or after any symbol or number in the command.

All numbers (addresses, data, counts) are hexadecimal (hex), but symbolic register names contain decimal register numbers. The hex digits are 0 through 9 and A through F. You can use uppercase and lowercase letters in hex numbers (A through F) and commands.

The following symbols are qualifier and argument conventions:

[ ] An optional qualifier or argument

{ } A required qualifier or argument

3.1.2 Address Specifiers

Several commands take one or more addresses as arguments. An address defines the address space and the offset into that space. The console supports five address spaces:

Physical memory
Virtual memory
General purpose registers (GPRs)
Internal processor registers (IPRs)
The PSL

The address space that the console references is inherited from the previous console reference, unless you explicitly specify another address space. The initial address space is physical memory.

3.1.3 Symbolic Addresses

The console supports symbolic references to addresses. A symbolic reference defines the address space and the offset into that space. Table 3–1 lists symbolic references supported by the console, grouped according to address space. You do not have to use an address space qualifier when using a symbolic address.

Table 3–1 Console Symbolic Addresses

Symb	Addr	Symb	Addr	Symb	Addr	Symb	Addr
/G—General Purpose Registers							
R0	00	R4	04	R8	08	R12 (AP)	0C
R1	01	R5	05	R9	09	R13 (FP)	0D
R2	02	R6	06	R10	0A	R14 (SP)	0E
R3	03	R7	07	R11	0B	R15 (PC)	0F
/M—Processor Status Longword							
PSL	—						
/I—Internal Processor Registers							
pr$_ksp	00	pr$_pcbb	10	pr$_rxcs	20	—	30
pr$_esp	01	pr$_scbb	11	pr$_rxdb	21	—	31
pr$_ssp	02	pr$_ipl	12	pr$_txcs	22	—	32
pr$_usp	03	pr$_astlv	13	pr$_txdb	23	—	33
pr$_isp	04	pr$_sirr	14	—	24	—	34
—	05	pr$_sisr	15	—	25	—	35
—	06	—	16	pr$_mcesr	26	—	36
—	07	—	17	—	27	pr$_ ioreset	37

Note: All symbolic values in this table are in hexadecimal.

(continued on next page)

Table 3–1 (Cont.) Console Symbolic Addresses

Symb	Addr	Symb	Addr	Symb	Addr	Symb	Addr
			/i—Internal Processor Registers				
pr$_p0br	08	pr$_iccs	18	—	28	pr$_ mapen	38
pr$_p0lr	09	pr$_nicr	19	—	29	pr$_tbia	39
pr$_p1br	0A	pr$_icr	1A	pr$_savpc	2A	pr$_tbis	3A
pr$_p1lr	0B	pr$_todr	1B	pr$_savpsl	2B	—	3B
pr$_sbr	0C	—	1C	—	2C	—	3C
pr$_slr	0D	—	1D	—	2D	—	3D
—	0E	—	1E	—	2E	pr$_sid	3E
—	0F	—	1F	—	2F	pr$_ tbchk	3F
pr$_ecr	7D						
pr$_cctl	A0	pr$_neoadr	B0	pr$_vmar	D0	—	F0
—	A1	—	B1	pr$_vtag	D1	—	F1
pr$_bcdecc	A2	pr$_ neocmd	B2	pr$_vdata	D2	pr$_pcadr	F2
pr$_bcetsts	A3	—	B3	pr$_icsr	D3	—	F3
pr$_bcetidx	A4	pr$_ nedathi	B4	—	D4	pr$_pcsts	F4
pr$_bcetag	A5	—	B5	—	D5	—	F5
pr$_ bcedsts	A6	pr$_ nedatlo	B6	—	D6	—	F6
pr$_ bcedidx	A7	—	B7	pr$_ pamode	E7	—	F7
pr$_ bcedecc	A8	pr$_neicmd	B8	—	E8	pr$_pcctl	F8
pr$_cefadr	AB	—	B9	—	E9	—	F9
pr$_cefsts	AC	—	BA	pr$_tbadr	EC	—	FA
pr$_nests	AE	—	BB	pr$_tbsts	ED	—	FB
pr$_bctag	01000000	pr$_bcflush	01400000	pr$_pctag	01800000	pr$_ pcdap	01C00000

(continued on next page)

Table 3–1 (Cont.) Console Symbolic Addresses

Symb	Addr	Symb	Addr	Symb	Addr	Symb	Addr
			/P—Physical (VAX I/O Space)				
qbio	20000000	qbmem	30000000	qbmbr	20080010	—	—
rom or feprom	20040000	—	—	bdr	20084000	—	—
scr	20080000	dser	20080004	qbear	20080008	dear	2008000C
ipcr0	20001f40	ipcr1	20001f42	ipcr2	20001f44	ipcr3	20001f46
sscram/ nvr	20140400	ssccr	20140010	cbtcr	20140020	dledr	20140030
ad0mat	20140130	ad0msk	20140134	ad1mat	20140140	ad1msk	20140144
tcr0	20140100	tir0	20140104	tnir0	20140108	tivr0	2014010c
tcr1	20140110	tir1	20140114	tnir1	20140118	tivr1	2014011c
nicsr0	20008000	nicsr1	20008004	nicsr2	20008008	nicsr3	2000800C
nicsr4	20008010	nicsr5	20008014	nicsr6	20008018	nicsr7	2000801C
—	20008020	nicsr9	20008024	nicsr10	20008028	nicsr11	2000802C
nicsr12	20008030	nicsr13	20008034	nicsr14	20008038	nicsr15	2000803C
sgec_setup	20008000	sgec_txpoll	20008004	sgec_rxpoll	20008008	sgec_rba	2000800C
sgec_tba	20008010	sgec_status	20008014	sgec_mode	20008018	sgec_sbr	2000801C
—	20008020	sgec_wdt	20008024	sgec_mfc	20008028	sgec_ verlo	2000802C
sgec_verhi	20008030	sgec_proc	20008034	sgec_bpt	20008038	sgec_cmd	2000803C
shac_sswcr	20004230	shac_ sshma	20004244	shac_pqbbr	20004248	shac_psr	2000424c
shac_pesr	20004250	shac_pfar	20004254	shac_ppr	20004258	shac_ pmcsr	2000425C
shac_ pcq0cr	20004280	shac_ pcq1cr	20004284	shac_ pcq2cr	20004288	shac_ pcq3cr	2000428C
shac_ pdfqcr	20004290	shac_ pmfqcr	20004294	shac_psrcr	20004298	shac_pecr	2000429C
shac_pdcr	200042A0	shac_picr	200042A4	shac_pmtcr	200042A8	shac_ pmtecr	200042AC

(continued on next page)

Table 3–1 (Cont.) Console Symbolic Addresses

Symb	Addr	Symb	Addr	Symb	Addr	Symb	Addr
			/P—Physical (VAX I/O Space)				
nmccwb	21000110	modr	21010000	—	—	—	—
memcon0	21018000	memcon1	21018004	memcon2	21018008	memcon3	2101800c
memcon4	21018010	memcon5	21018014	memcon6	21018018	memcon7	2101801c
memsig8	21018020	memsig9	21018024	memsig10	21018028	memsig11	2101802c
memsig12	21018030	memsig13	21018034	memsig14	21018038	memsig15	2101803c
mear	21018040	mser	21018044	nmcdsr	21018048	moamr	2101804C
cesr	21020000	cmcdsr	21020004	csear1	21020008	csear2	2102000c
cioear1	21020010	cioear2	21020014	cnear	21020018	—	—
scdadrB	21C00000	scddirB	21C00004	scsicsr0B	22000080	scsicsr1B	22000084
scsicsr2B	22000088	scsicsr3B	2200008c	scsicsr4B	22000090	scsicsr5B	22000094
scsicsr6B	22000098	scsicsr7B	2200009c	scsicsr8B	22000A0	scsicsr9B	220000A4
scsicsraB	220000A8	scsicsrbB	220000Ac	scsicsrcB	220000B0	scsimapB	23000000
intmskB	21C00008	intreqB	21C0000c	—	—	—	—
csr	25000000	rbuf	25000004	lpr	25000004	tcr	26000008
msr	2500000C	tdr	2500000C	ssr	25800000	—	—
scdadr	25C00000	scddir	25C00004	intmsk	25C00008	intreq	25C0000C
scsicsr0	26000080	scsicsr1	26000084	scsicsr2	26000088	scsicsr3	2600008C
scsicsr4	26000090	scsicsr5	26000094	scsicsr6	25c00098	scsicsr7	25C0009C
scsicsr8	260000A0	scsicsr9	260000A4	scsicsra	260000A8	scsicsrb	260000AC
scsicsrc	260000BU	scsimap	27000000	—	—	—	—

Table 3–2 lists symbolic addresses that you can use in any address space.

Table 3–2 Symbolic Addresses Used In Any Address Space

Symbol	Description
*	The location last referenced in an EXAMINE or DEPOSIT command.
+	The location immediately following the last location referenced in an EXAMINE or DEPOSIT command. For references to physical or virtual memory spaces, the location referenced is the last address, plus the size of the last reference (1 for byte, 2 for word, 4 for longword, 8 for quadword). For other address spaces, the address is the last address referenced plus one.
-	The location immediately preceding the last location referenced in an EXAMINE or DEPOSIT command. For references to physical or virtual memory spaces, the location referenced is the last address minus the size of this reference (1 for byte, 2 for word, 4 for longword, 8 for quadword). For other address spaces, the address is the last address referenced minus one.
@	The location addressed by the last location referenced in an EXAMINE or DEPOSIT command.

3.1.4 Console Numeric Expression Radix Specifiers

By default, the console treats any numeric expression used as an address or a datum as a hexadecimal integer. The user may override the default radix by using one of the specifiers listed in Table 3–3.

Table 3–3 Console Radix Specifiers

Form 1	Form 2	Radix
%b	^b	Binary
%o	^o	Octal
%d	^d	Decimal
%x	^x	Hexadecimal, default

For instance, the value 19 is by default hexadecimal, but it may also be represented as %b11001, %o31, %d25, and %x19 (or in the alternate form as ^b11001, ^o31, ^d25, and ^x19).

3.1.5 Console Command Qualifiers

You can enter console command qualifiers in any order on the command line after the command keyword. The three types of qualifiers are data control, address space control, and command specific. Table 3–4 lists and describes the data control and address space control qualifiers. Command specific qualifiers are listed in the descriptions of individual commands.

Table 3–4 Console Command Qualifiers

Qualifier	Description
Data Control	
/B	The data size is byte.
/W	The data size is word.
/L	The data size is longword.
/Q	The data size is quadword.
/N:{count}	An unsigned hexadecimal integer that is evaluated into a longword. This qualifier determines the number of additional operations that are to take place on EXAMINE, DEPOSIT, MOVE, and SEARCH commands. An error message appears if the number overflows 32 bits.
/STEP:{size}	Step. Overrides the default increment of the console current reference. Commands that manipulate memory, such as EXAMINE, DEPOSIT, MOVE, and SEARCH, normally increment the console current reference by the size of the data being used.
/WRONG	Wrong. On writes, 3 is used as the value of the ECC bits, which always generates double bit errors. Ignores ECC errors on main memory reads.

(continued on next page)

Table 3–4 (Cont.) Console Command Qualifiers

Qualifier	Description
Address Space Control	
/G	General purpose register (GPR) address space, R0–R15. The data size is always longword.
/I	Internal processor register (IPR) address space. Accessible only by the MTPR and MFPR instructions. The data size is always longword.
/V	Virtual memory address space. All access and protection checking occur. If access to a program running with the current PSL is not allowed, the console issues an error message. Deposits to virtual space cause the PTE<M> bit to be set. If memory mapping is not enabled, virtual addresses are equal to physical addresses. Note that when you examine virtual memory, the address space and address in the response is the physical address of the virtual address.
/P	Physical memory address space.
/M	Processor status longword (PSL) address space. The data size is always longword.
/U	Access to console private memory is allowed. This qualifier also disables virtual address protection checks. On virtual address writes, the PTE<M> bit is not set if the /U qualifier is present. This qualifier is not inherited; it must be respecified on each command.

3.1.6 Console Command Keywords

Table 3–5 lists command keywords by type. Table 3–6 lists the parameters, qualifiers, and arguments for each console command. Parameters, used with the SET and SHOW commands only, are listed in the first column along with the command.

You should not use abbreviations in programs. Although it is possible to abbreviate by using the minimum number of characters required to uniquely identify a command or parameter, these abbreviations may become ambiguous at a later time if an updated version of the firmware contains new commands or parameters.

Table 3–5 Command Keywords by Type

Processor Control	Data Transfer	Console Control
BOOT	DEPOSIT	CONFIGURE
CONTINUE	EXAMINE	FIND
HALT	MOVE	REPEAT
INITIALIZE	SEARCH	SET
NEXT	X	SHOW
START		TEST
		LOGIN
UNJAM		!

Table 3–6 Console Command Summary

Command	Qualifiers	Argument	Other(s)
BOOT	/R5:{boot_flags} /{boot_flags}	[{boot_device}[,{boot_device}]...]	—
CONFIGURE	—	—	—
CONTINUE	—	—	—
DEPOSIT	/B /W /L /Q — /G /I /V /P /M /U /N:{count} /STEP:{size} /WRONG	{address}	{data} [{data}]
EXAMINE	/B /W /L /Q — /G /I /V /P /M /U /N:{count} /STEP:{size} /WRONG /INSTRUCTION	[{address}]	—
FIND	/MEM /RPB	—	—
HALT	—	—	—
HELP	—	—	—
INITIALIZE	—	—	—
LOGIN	—	—	—

(continued on next page)

Table 3–6 (Cont.) Console Command Summary

Command	Qualifiers	Argument	Other(s)
MOVE	/B /W /L /Q — /V /P /U /N:{count} /STEP:{size} /WRONG	{src_address}	{dest_ address}
NEXT	—	[{count}]	—
REPEAT	—	{command}	—
SEARCH	/B /W /L /Q — /V /P /U /N:{count} /STEP:{size} /WRONG /NOT	{start_address}	{pattern} [{mask}]
SET BFLAG	—	{bitmap}	—
SET BOOT	—	[{boot_device}[,{boot_ device}]...	—
SET CONTROLP	—	{0/1}	—
SET HALT	—	{halt_action}	—
SCSI_ID	—	{bus}[1] {id}	—
SET HOST	/DUP /DSSI /BUS:{0/1}	{node_number}	[{task}]
SET HOST	/DUP /UQSSP {/DISK ! /TAPE } /DUP /UQSSP	{controller_number} {csr_address}	[{task}] [{task}]
SET HOST	/MAINTENANCE /UQSSP /SERVICE /MAINTENANCE /UQSSP	{controller_number} {csr_address}	
SET LANGUAGE	—	{language_type}	—
SET RECALL	—	{0/1}	—
SHOW BFL(A)G	—	—	—
SHOW BOOT	—	—	—
SHOW CONTROLP	—	—	—
SHOW DSSI	—	—	—
SHOW HALT	—	—	—
SHOW LANGUAGE	—	—	—

[1]For Open VMS version 1.3 and earlier, only one argument, the id, is used. For later versions, two arguments are accepted; the first refers to the bus, the second to the id; if only one argument is supplied, the system defaults to bus 0, and the argument is taken as the id.

(continued on next page)

Table 3–6 (Cont.) Console Command Summary

Command	Qualifiers	Argument	Other(s)
SHOW MEMORY	/FULL	—	—
SHOW QBUS	—	—	—
SHOW RECALL	—	—	—
SHOW RLV12	—	—	—
SHOW SCSI	—	—	—
SHOW SCSI_ID	—	—	—
SHOW TRANSLATION		{phys_address}	—
SHOW UQSSP	—	—	—
SHOW VERSION	—	—	—
START	—	{address}	—
TEST	—	{test_number}	[{parameters}]
UNJAM	—	—	—
X	—	{address}	{count}

3.2 Console Commands

The following sections describe all the console commands, give the command formats with their qualifiers, and describe the significance of each qualifier.

3.2.1 BOOT

The BOOT command initializes the processor and transfers execution to Virtual Memory Boot (VMB). VMB attempts to boot the operating system from the specified device or list of devices, or from the default boot device if none is specified. The console qualifies the bootstrap operation by passing a boot flags bitmap to VMB in R5.

Format:

BOOT [qualifier-list] [{boot_device},{boot_device},...]

If you do not enter either the qualifier or the device name, the default value is used. Explicitly stating the boot flags or the boot device overrides, but does not permanently change, the corresponding default value.

When specifying a list of boot devices (up to 32 characters, with devices separated by commas and no spaces), the system checks the devices in the order specified and boots from the first one that contains bootable software.

Note

If included in a string of boot devices, the Ethernet device, EZA0, should be placed only as the last device of the string. The system will continuously attempt to boot from EZA0.

Set the default boot device and boot flags with the SET BOOT and SET BFLAG commands. If you do not set a default boot device, the processor times out after 30 seconds and attempts to boot from the Ethernet device, EZA0.

Qualifiers:

Command specific:

/R5:{boot_flags}	A 32-bit hex value passed to VMB in R5. The console does not interpret this value. Use the SET BFLAG command to specify a default boot flags longword. Use the SHOW BFLAG command to display the longword.
/{boot_flags}	Same as /R5:{boot_flags}
[device_name]	A character string of up to 32 characters. When specifying a list of boot devices, the device names should be separated by commas and no spaces. Apart from checking the length, the console does not interpret or validate the device name. The console converts the string to uppercase, then passes VMB a string descriptor to this device name in R0. Use the SET BOOT command to specify a default boot device or list of devices. Use the SHOW BOOT command to display the default boot device. The factory default device is the Ethernet device, EZA0. Refer to the *MicroVAX 3100 Customer Technical Information* manuals for a list of the boot devices supported by the system.

Examples:

```
>>>SHOW BOOT
DKA300
>>>SHOW BFLAG
00000000
>>>B    !Boot using default boot flags and device.
(BOOT/R5:0 DKA300)

  2..
-DKA300
```

3.2.2 CONTINUE

The CONTINUE command causes the processor to begin instruction execution at the address currently contained in the program counter (PC). This address is the address stored in the PC when the system entered console mode or an address that the user specifies using the DEPOSIT command. The CONTINUE command does not perform a processor initialization. The console enters program I/O mode.

Format:

CONTINUE

Example:

```
>>>CONTINUE
$       !OpenVMS DCL prompt
```

3.2.3 DEPOSIT

The DEPOSIT command deposits data into the address specified. If you do not specify an address space or data size qualifier, the console uses the last address space and data size used in a DEPOSIT, EXAMINE, MOVE, or SEARCH command. After processor initialization, the default address space is physical memory and the default data size is longword. If you specify conflicting address space or data sizes, the console ignores the command and issues an error message.

Format:

DEPOSIT [qualifier-list] {address} {data} [data...]

Qualifiers:

Data control: /B, /W, /L, /Q, /N:{count}, /STEP:{size}, /WRONG

Address space control: /G, /I, /M, /P, /V, /U

Arguments:

{address} A longword address that specifies the first location into which data is deposited. The address can be an actual address or a symbolic address.

{data} The data to be deposited. If the specified data is larger than the deposit data size, the firmware ignores the command and issues an error response. If the specified data is smaller than the deposit data size, it is extended on the left with zeros.

[{data}] Additional data to be deposited (as much as can fit on the command line).

Examples:

```
>>>D/P/B/N:1FF 0 0          ! Clear first 512 bytes of
                            ! physical memory.
>>>D/V/L/N:3 1234 5         ! Deposit 5 into four longwords
                            ! starting at virtual memory address
                            ! 1234.
>>>D/N:8 R0 FFFFFFFF        ! Loads GPRs R0 through R8 with -1.
>>>D/L/P/N:10/ST:200 0 8    ! Deposit 8 in the first longword of
                            ! the first 17 pages in physical
                            ! memory.
>>>D/N:200 - 0              ! Starting at previous address, clear
                            ! 513 longwords or 2052 bytes.
```

3.2.4 EXAMINE

The EXAMINE command examines the contents of the memory location or register specified by the address. If no address is specified, + is assumed. The display line consists of a single character address specifier, the physical address to be examined, and the examined data.

EXAMINE uses the same qualifiers as DEPOSIT. However, the /WRONG qualifier causes EXAMINE to ignore ECC errors on reads from physical memory. The EXAMINE command also supports an /INSTRUCTION qualifier, which will disassemble the instructions at the current address.

Format:

EXAMINE [qualifier-list] [address]

Qualifiers:

Data control: /B, /W, /L, /Q, /N:{count}, /STEP:{size}, /WRONG

Address space control: /G, /I, /M, /P, /V, /U

Command specific:

/INSTRUCTION Disassembles and displays the VAX MACRO–32 instruction at the specified address.

Arguments:

[{address}] A longword address that specifies the first location to be examined. The address can be an actual or a symbolic address. If no address is specified, + is assumed.

Examples:

```
>>>EX PC                          ! Examine the PC.
  G 0000000F FFFFFFFC
>>>EX SP                          ! Examine the SP.
  G 0000000E 00000200
>>>EX PSL                         ! Examine the PSL.
  M 00000000 041F0000
>>>E/M                            ! Examine PSL another way.
  M 00000000 041F0000
>>>E R4/N:5                       ! Examine R4 through R9.
  G 00000004 00000000
  G 00000005 00000000
  G 00000006 00000000
  G 00000007 00000000
  G 00000008 00000000
  G 00000009 801D9000

>>>EX PR$_SCBB                    !Examine the SCBB, IPR 17
  I 00000011 2004A000             ! (decimal).

>>>E/P 0                          ! Examine local memory 0.
  P 00000000 00000000

>>>EX /INS 20040000               ! Examine 1st byte of ROM.
  P 20040000   11 BRB     20040019

>>>EX /INS/N:5 20040019           ! Disassemble from branch.
  P 20040019   D0 MOVL    I^#20140000,@#20140000
  P 20040024   D2 MCOML   @#20140030,@#20140502
  P 2004002F   D2 MCOML   S^#0E,@#20140030
  P 20040036   7D MOVQ    R0,@#201404B2
  P 2004003D   D0 MOVL    I^#201404B2,R1
  P 20040044   DB MFPR    S^#2A,B^44(R1)

>>>E/INS                          ! Look at next instruction.
  P 20040048   DB MFPR    S^#2B,B^48(R1)

>>>
```

3.2.5 FIND

The FIND command searches main memory, starting at address zero for a page-aligned 128-Kbyte segment of good memory, or a restart parameter block (RPB). If the command finds the segment or RPB, its address plus 512 is left in Stack Pointer (SP) R14. If it does not find the segment or RPB, the console issues an error message and preserves the contents of SP. If you do not specify a qualifier, /RPB is assumed.

Format:

FIND [qualifier-list]

Qualifiers:

Command specific:

/MEMORY — Searches memory for a page-aligned block of good memory, 128K bytes in length. The search looks only at memory that is deemed usable by the bitmap. This command leaves the contents of memory unchanged.

/RPB — Searches all physical memory for an RPB. The search does not use the bitmap to qualify which pages are looked at. The command leaves the contents of memory unchanged.

Examples:

```
>>>EX SP                    ! Check the SP.
  G 0000000E 00000000
>>>FIND /MEM                ! Look for a valid 128 Kbytes.
>>>EX SP                    ! Note where it was found.
  G 0000000E 00000200
>>>FIND /RPB                ! Check for valid RPB.
?2C FND ERR 00C00004        ! None to be found here.
>>>
```

3.2.6 HALT

The HALT command has no effect. It is included for compatibility with other VAX consoles.

Format:

HALT

Example:

```
>>>HALT     ! Pretend to halt.
>>>
```

3.2.7 HELP

The HELP command provides information about command syntax and usage.

Format:

HELP

Example:

```
>>>HELP
Following is a brief summary of all the commands supported by the
console:

UPPERCASE  denotes a keyword that you must type in
|          denotes an OR condition
[]         denotes optional parameters
<>         denotes a field specifying a syntactically correct value
..         denotes one of an inclusive range of integers
...        denotes that the previous item may be repeated

Valid qualifiers:
    /B /W /L /Q /INSTRUCTION
    /G /I /V /P /M
    /STEP: /N: /NOT
    /WRONG /U

Valid commands:
    BOOT [[/R5:]<boot_flags>] [<boot_device>]
    CONTINUE
    DEPOSIT [<qualifiers>] <address> <datum> [<datum>...]
    EXAMINE [<qualifiers>] [<address>]
    FIND [/MEMORY | /RPB]
    HALT
    HELP
    INITIALIZE
    LOGIN
    MOVE [<qualifiers>] <address> <address>
    NEXT [<count>]
    REPEAT <command>
    SEARCH [<qualifiers>] <address> <pattern> [<mask>]
    SET BFLG <boot_flags>
    SET BOOT <boot_device>
    SET DSSI_ID <bus_number> <id>
    SET HALT <0..4 |DEFAULT|RESTART|REBOOT|HALT|RESTART_REBOOT>
    SET HOST/DUP/DSSI/BUS:<0..3> <node_number> [<task>]
    SET LANGUAGE <1..15>
    SET PSE <0..1 |DISABLED | ENABLED>
    SET PSWD <password>
    SET RECALL <0..1 | DISABLED | ENABLED>
    SET SCSI_ID  <0..7>
    SHOW BFLG
    SHOW BOOT
    SHOW CONFIG
    SHOW DEVICE
    SHOW DSSI  [0..3]
    SHOW DSSI_ID
    SHOW ERRORS
    SHOW ESTAT
    SHOW ETHERNET
    SHOW HALT
    SHOW LANGUAGE
    SHOW MEMORY [/FULL]
```

```
        SHOW PSE
        SHOW RECALL
        SHOW SAVED_STATE
        SHOW SCSI
        SHOW SCSI_ID
        SHOW TESTS
        SHOW TRANSLATION <physical_address>
        SHOW VERSION
        START <address>
        TEST [<test_code> [<parameters>]]
        UNJAM
        X <address> <count>

>>>
```

3.2.8 INITIALIZE

The INITIALIZE command performs a processor initialization.

Format:

INITIALIZE

The following registers are initialized:

Register	State at Initialization
PSL	041F0000
IPL	1F
ASTLVL	4
SISR	0
ICCS	Bits <6> and <0> clear; the rest are unpredictable.
RXCS	0
TXCS	80
MAPEN	0
Caches	Flushed
Instruction buffer	Unaffected
Console previous reference	Longword, physical, address 0
TODR	Unaffected
Main memory	Unaffected
General registers	Unaffected
Halt code	Unaffected

Register	State at Initialization
Bootstrap-in-progress flag	Unaffected
Internal restart-in-progress flag	Unaffected

The firmware clears all error status bits and initializes the following:

- CDAL bus timer
- Address decode and match registers
- Programmable timer interrupt vectors
- The QUART LPR register is set to 9600 baud
- All error status bits are cleared

Example:

```
>>>INIT
>>>
```

3.2.9 LOGIN

Allows you to put the system in privileged console mode. When the console security feature is enabled and when you put the system in secure console mode, the system operates in unprivileged console mode. You can access only a subset of the console commands. To access the full range of console commands, you must use this command. This command may only be executed in secure console mode. The format of this command is as follows:

LO[GIN]

When you enter the command, the system prompts you for a password as follows:

```
Password:
```

You must enter the current console security password. If you do not enter the correct password, the system displays the error message, INCORRECT PASSWORD. When you enter the console security password, the system operates in privileged console mode. In this mode, you can use all the console commands. The system exits from privileged console mode when you enter one of the following console commands:

- BOOT
- CONTINUE
- HALT

- START

3.2.10 MOVE

The MOVE command copies the block of memory starting at the source address to a block beginning at the destination address. Typically, this command has an /N qualifier so that more than one datum is transferred. The destination correctly reflects the contents of the source, regardless of the overlap between the source and the data.

The MOVE command actually performs byte, word, longword, and quadword reads and writes as needed in the process of moving the data. Moves are supported only for the physical and virtual address spaces.

Format:

MOVE [qualifier-list] [src_address] [dest_address]

Qualifiers:

Data control: /B, /W, /L, /Q, /N:[count], /STEP:[size], /WRONG

Address space control: /V, /U, /P

Arguments:

[src_address] A longword address that specifies the first location of the source data to be copied.

[dest_address] A longword address that specifies the destination of the first byte of data. These addresses may be an actual address or a symbolic address. If no address is specified, + is assumed.

Examples:

```
>>>EX/N:4 0              ! Observe destination.
P 00000000 00000000
P 00000004 00000000
P 00000008 00000000
P 0000000C 00000000
P 00000010 00000000

>>>EX/N:4 200            ! Observe source data.
P 00000200 58DD0520
P 00000204 585E04C1
P 00000208 00FF8FBB
P 0000020C 5208A8D0
P 00000210 540CA8DE

>>>MOV/N:4 200 0         ! Move the data.
```

```
>>>EX/N:4 0                ! Observe moved data.
P 00000000 58DD0520
P 00000004 585E04C1
P 00000008 00FF8FBB
P 0000000C 5208A8D0
P 00000010 540CA8DE
>>>
```

3.2.11 NEXT

The NEXT command executes the specified number of macro instructions. If no count is specified, 1 is assumed.

After the last macro instruction is executed, the console reenters console I/O mode.

Format:

NEXT [count]

The console implements the NEXT command, using the trace trap enable and trace pending bits in the PSL and the trace pending vector in the SCB.

The console enters the "Spacebar Step Mode". In this mode, subsequent spacebar strokes initiate single steps and a carriage return forces a return to the console prompt.

The following restrictions apply:

- If memory management is enabled, the NEXT command works only if the first page in SSC RAM is mapped in S0 (system) space.
- Overhead associated with the NEXT command affects execution time of an instruction.
- The NEXT command elevates the IPL to 31 for long periods of time (milliseconds) while single-stepping over several commands.
- Unpredictable results occur if the macro instruction being stepped over modifies either the SCBB or the trace trap entry. This means that you cannot use the NEXT command in conjunction with other debuggers.

Arguments:

[count] A value representing the number of macro instructions to execute.

Examples:

```
>>>DEP 1000 50D650D4                ! Create a simple program.
>>>DEP 1004 125005D1
>>>DEP 1008 00FE11F9
>>>EX /INSTRUCTION /N:5 1000        ! List it.
  P 00001000   D4 CLRL    R0
  P 00001002   D6 INCL    R0
  P 00001004   D1 CMPL    S^#05,R0
  P 00001007   12 BNEQ    00001002
  P 00001009   11 BRB     00001009
  P 0000100B   00 HALT
>>>DEP PR$ SCBB 200                ! Set up a user SCBB...
>>>DEP PC 1000                      ! ...and the PC.
>>>
>>>N                               ! Single step...
  P 00001002   D6 INCL    R0         ! SPACEBAR
  P 00001004   D1 CMPL    S^#05,R0   ! SPACEBAR
  P 00001007   12 BNEQ    00001002   ! SPACEBAR
  P 00001002   D6 INCL    R0         ! CR
>>>N 5                             ! ...or multiple step the program.
  P 00001004   D1 CMPL    S^#05,R0
  P 00001007   12 BNEQ    00001002
  P 00001002   D6 INCL    R0
  P 00001004   D1 CMPL    S^#05,R0
  P 00001007   12 BNEQ    00001002
>>>N 7
  P 00001002   D6 INCL    R0
  P 00001004   D1 CMPL    S^#05,R0
  P 00001007   12 BNEQ    00001002
  P 00001002   D6 INCL    R0
  P 00001004   D1 CMPL    S^#05,R0
  P 00001007   12 BNEQ    00001002
  P 00001009   11 BRB     00001009
>>>N
  P 00001009   11 BRB     00001009
>>>
```

3.2.12 REPEAT

The REPEAT command repeatedly displays and executes the specified command. Press Ctrl/C to stop the command. You can specify any valid console command except the REPEAT command.

Format:

REPEAT {command}

Arguments:

(command) A valid console command other than REPEAT.

Examples:

```
>>>REPEAT EX PR$_TODR !Watch the clock.
  I 0000001B 5AFE78CE
  I 0000001B 5AFE78D1
  I 0000001B 5AFE78FD
  I 0000001B 5AFE7900
  I 0000001B 5AFE7903
  I 0000001B 5AFE7907
  I 0000001B 5AFE790A
  I 0000001B 5AFE790D
  I 0000001B 5AFE7910
  I 0000001B 5AFE793C
  I 0000001B 5AFE793F
  I 0000001B 5AFE7942
  I 0000001B 5AFE7946
  I 0000001B 5AFE7949
  I 0000001B 5AFE794C
  I 0000001B 5AFE794F
  I 0000001B 5^C
>>>
```

3.2.13 SEARCH

The SEARCH command finds all occurrences of a pattern and reports the addresses where the pattern was found. If the /NOT qualifier is present, the command reports all addresses in which the pattern did not match.

Format:

SEARCH [qualifier-list] [address] [pattern] [[mask]]

SEARCH accepts an optional mask that indicates bits to be ignored (*don't care* bits). For example, to ignore bit 0 in the comparison, specify a mask of 1. The mask, if not present, defaults to 0.

A match occurs if (pattern and not mask) = (data and not mask), where:

Pattern is the target data.
Mask is the optional don't care bitmask (which defaults to 0).
Data is the data at the current address.

SEARCH reports the address under the following conditions:

/NOT Qualifier	Match Condition	Action
Absent	True	Report address
Absent	False	No report
Present	True	No report
Present	False	Report address

The address is advanced by the size of the pattern (byte, word, longword, or quadword), unless overridden by the /STEP qualifier.

Qualifiers:

Data control: /B, /W, /L, /Q, /N:{count}, /STEP:{size}, /WRONG

Address space control: /P, /V, /U

Command specific:

/NOT Inverts the sense of the match.

Arguments:

{start_address} A longword address that specifies the first location subject to the search. This address can be an actual address or a symbolic address. If no address is specified, + is assumed.

{pattern} The target data.

[{mask}] A mask of the bits desired in the comparison.

Examples:

```
>>>DEP /P/L/N:1000 0 0                ! Clear some memory.
>>>
>>>DEP 300 12345678                   ! Deposit some search data.
>>>DEP 401 12345678
>>>DEP 502 87654321
>>>
>>>SEARCH /N:1000 /ST:1 0 12345678    ! Search for all occurrences
  P 00000300 12345678                 ! of 12345678 on any byte
  P 00000401 12345678                 ! boundary.  Then try on
>>>SEARCH /N:1000 0 12345678          ! longword boundaries.
  P 00000300 12345678                 ! Search for all non-zero
>>>SEARCH /N:1000 /NOT 0 0            ! longwords.
  P 00000300 12345678
  P 00000400 34567800
  P 00000404 00000012
  P 00000500 43210000
  P 00000504 00008765
>>>SEARCH /N:1000 /ST:1 0 1 FFFFFFFE  ! Search for odd-numbered
                                      ! longwords on any boundary.
  P 00000502 87654321
  P 00000503 00876543
  P 00000504 00008765
  P 00000505 00000087
>>>SEARCH /N:1000 /B 0 12             ! Search for all occurrences
  P 00000303 12                       ! of the byte 12.
  P 00000404 12
>>>SEARCH /N:1000 /ST:1 /w 0 FE11     ! Search for all words that
>>>                                   ! could be interpreted as
>>>                                   ! a spin (10$: brb 10$).
>>>                                   ! Note that none were found.
```

3.2.14 SET

The SET command sets the parameter to the value you specify.

Format:

SET {parameter} {value}

Parameters:

BFLAG	Sets the default R5 boot flags. The value must be a hex number of up to eight digits.
BOOT	Sets the default boot device. The value must be a valid device name or list of device names as specified in the BOOT command description in Section 3.2.1.
HALT	Sets the user-defined halt action. Acceptable values are the keywords "default", "restart", "reboot", "halt", "restart_reboot", or a number in the range 0 to 4 inclusive.

HOST	Invoke the DUP or MAINTENANCE driver on the selected node. Only SET HOST/DUP accepts a value parameter. The hierarchy of the SET HOST qualifiers listed below suggests the appropriate usage. Each qualifier only supports additional qualifiers at levels below it.
LANGUAGE	Sets console language and keyboard type. If the current console terminal does not support the multinational character set (MCS), then this command has no effect and the console message appears in English. Values are 1 through 15.
PSE	Allows you to enable or disable the console security feature of the system. The SET PSE command accepts the following values: • 0—Console security disabled • 1—Console security enabled When the console security feature is enabled, only a subset of the console commands is available to the user. To enable the complete set of console commands once the console security feature is enabled, you must use the LOGIN command (see Section 3.2.9).
PSWD	Allows you to set or change the console security password.
RECALL	Sets command recall state to either ENABLED (1) or DISABLED (0).
SCSI_ID	Sets the SCSI ID of the SCSI controller to a number in the range 0 to 7. The SCSI ID of the SCSI controller is set to 6 before the system is shipped. For the KZDDA option second SCSI bus, You must enter two arguments; the bus, then the id.

Qualifiers: Listed in the parameter descriptions above.

Examples:

```
>>>
>>>SET BFLAG 220
>>>
>>>SET BOOT DUA0
>>>
>>>SET LANGUAGE 5
>>>
>>>SET HALT RESTART
>>>
```

3.2.15 SHOW

The SHOW command displays the console parameter you specify.

Format:

SHOW [parameter]

Parameters:

BFLAG	Displays the default R5 boot flags.
BOOT	Displays the default boot device.
CONFIG	Displays the system configuration. The command displays information about the devices that the firmware has tested. It also displays the device errors that the most recent device test detected.
DEVICE	Displays all devices in the system.
HALT	Shows the user-defined halt action.
ESTAT	Shows results from last run of the system exerciser, tests 100 to 107. Data is volatile and is destroyed by running other tests or boots, etc. SHOW ESTAT normally done immediately after running the system test.
ERRORS	Shows saved data on tests which failed.
ETHERNET	Displays hardware Ethernet address for all Ethernet adapters that can be found. Displays as blank if no Ethernet adapter is present.
LANGUAGE	Displays console language and keyboard type. Refer to the corresponding SET LANGUAGE command for the meaning.
MEMORY	Displays main memory configuration. /FULL—Additionally, displays the normally inaccessible areas of memory, such as the PFN bitmap pages, and the console scratch memory pages. Also reports the addresses of bad pages, as defined by the bitmap.
PSE	Displays the condition of the console security feature of the system.
RECALL	Shows the current state of command recall, either ENABLED or DISABLED.

This information is obtained from the media type field of the MSCP command GET UNIT STATUS. The console does not display device information if a node is not running (or cannot run) an MSCP server.)

SCSI	Shows any SCSI devices in the system.
TRANSLATION	Shows any virtual addresses that map to the specified physical address. The firmware uses the current values of page table base and length registers to perform its search; it is assumed that page tables have been properly built.
VERSION	Displays the current firmware version.

Qualifiers: Listed in the parameter descriptions above.

Examples:

```
>>>
>>>SHOW BFLAG
00000220
>>>
>>>SHOW BOOT
DUA0
>>>SHOW CONTROLP
>>>
>>>SHOW ETHERNET
Ethernet Adapter
-EZA0 (08-00-2B-0B-29-14)
>>>
>>>SHOW HALT
restart
>>>
>>>SHOW LANGUAGE
English (United States/Canada)
>>>

>>>show memory

16 MB RAM, SIMM Set (0A,0B,0C,0D) present
Memory Set 0: 04000000 to 04FFFFFF, 16MB, 32768 good pages, 0 bad pages

64 MB RAM, SIMM Set (1E,1F,1G,1H) present
Memory Set 1: 00000000 to 03FFFFFF, 64MB, 131072 good pages, 0 bad pages

Total of 80MB, 163840 good pages, 0 bad pages, 136 reserved pages
>>>


;show memory / full
>>>show mem/full

16 MB RAM, SIMM Set (0A,0B,0C,0D) present
Memory Set 0: 00000000 to 00FFFFFF, 16MB, 32768 good pages, 0 bad pages

Total of 16MB, 32768 good pages, 0 bad pages, 104 reserved pages

Memory Bitmap
-00FF3000 to 00FF3FFF, 8 pages

Console Scratch Area
-00FF4000 to 00FF7FFF, 32 pages

Scan of Bad Pages
>>>
```

```
>>>SHOW SCSI
SCSI Adapter 0 (761300), SCSI ID 7
-DKA100 (DEC TLZ04)
>>>
>>>SHOW TRANSLATION 1000
 V 80001000
>>>
>>>SHOW VERSION
KA50 Vn.n VMBn.n

>>>
```

3.2.16 START

The START command starts instruction execution at the address you specify. If no address is given, the current PC is used. If memory mapping is enabled, macro instructions are executed from virtual memory, and the address is treated as a virtual address. The START command is equivalent to a DEPOSIT to PC, followed by a CONTINUE. It does not perform a processor initialization.

Format:

START [[address]]

Arguments:

[address] The address at which to begin execution. This address is loaded into the user's PC.

Example:

```
>>>START 1000
```

3.2.17 TEST

The TEST command invokes a diagnostic test program specified by the test number. If you enter a test number of 0 (zero), the power-up diagnostics are executed. The console accepts an optional list of up to five additional hexadecimal arguments.

Refer to Chapter 5 for a detailed explanation of the diagnostics.

Format:

TEST [[test_number] [[test_arguments]]]

Arguments:

[test_number] A two-digit hex number specifying the test to be executed. No meaning to console, but meaning to tests themselves. **T 9E** lists arguments used by applicable tests.

(test_arguments) Up to five additional test arguments. These arguments are accepted, but they have no meaning to the console.

Example:

```
>>>TEST 0
72..71..70..69..68..67..66..65..64..63..62..61..60..59..58..57..
56..55..54..53..52..51..50..49..48..47..46..45..44..43..42..41..
40..39..38..37..36..35..34..33..32..31..30..29..28..27..26..25..
24..23..22..21..20..19..18..17..16..15..14..13..12..11..10..09..
08..07..06..05..04..03..
Tests completed.
>>>
```

Example:

```
>>>              ! Display the CPU registers.

>>>T 9C

  savpc=20048C68 savpsl=20048C68    sbr=03FA0000    slr=00003040
   p0br=80000000   p0lr=00182000   p1br=00000000   p1lr=00000000
    sid=13001401    sie=03020801  mapen=00000000
   tcr0=00000000   tir0=00000000  tnir0=00000000  tivr0=00000078
   tcr1=00000001   tir1=02AF768E  tnir1=0000000F  tivr1=0000007C
    bdr=3FFB08FF  ssccr=00D05070   scbb=20053400
DZ  csr=0020        tcr=0008        msr=0F75
    scr=0000D000   dser=00000000  qbear=0000000F   dear=00000000
  qbmbr=03FF8000   ipcr=0000
nicsr0=1FFF0003  3=00004030   4=00004050   5=8039FF00   6=83E0F000  7=00000000
nicsr9=04E204E2 10=00040000  11=00000000  12=00000000  13=00000000 15=0000FFFF
NISA=08-00-2B-29-1C-7A    intmsk=00 intreq=00 scdadr=00000000 scddir=0
SCSI_CSRs 0=00 1=00 2=00 3=00 4=00 6=05 5=05 7=00 8=16 9=5B A=5B B=00 C=04
VIC.....icsr=00000001    vmar=000007E0     ecr=000000CA
PC......pcctl=FFFFFC13   pcsts=FFFFF800   pcadr=FFFFFFF8
BC_128K..cctl=00000007 bcetsts=000003E0 bcetidx=FFFFFFE0  bcetag=FFFFFE00
......bcedsts=00000F00 bcedidx=001FFFF8 bcedecc=00000000
........nests=00000000  neoadr=E0055F70  neocmd=8000FF04  neicmd=000003FF
......nedathi=FFFFFFFF nedatlo=FF7F9FFF  cefsts=00019200  cefadr=E00002C0
MEMORY...mesr=00006000    mear=08406010     Add=21018040  mmcdsr=01111000
......memcon0=80000005 memcon1=00000007   moamr=00000000     ssr=C0CE
NCA......cesr=00000000  cmcdsr=0000C108   cnear=00000000
.......csear1=00000000  csear2=00000000 cioear1=00000000 cioear2=000002C0
.........iccs=00000000    nicr=FFFFD8F0     icr=FFFFD8F0    todr=00000000
>>>
```

Example:

```
>>>
>>>       ! list diagnostics and scripts
>>>TEST

Test
#   Address   Name                Parameters
-------------------------------------------------------------------------------
    20052200  SCB
    20055850  De_executive
30  2006A53C  Memory_Init_Bitmap  *** mark_Hard_SBEs ******
31  2006AB34  Memory_Setup_CSRs   **********
32  2005D148  NMC_registers       **********
33  2005D324  NMC_powerup         **
34  2005E6D8  SSC_ROM             ***
35  2005FB90  B_Cache_diag_mode   bypass_test_mask *********
37  20061590  Cache_w_Memory      bypass_test_mask *********
40  2006B5E0  Memory_count_pages  SIMM_set0 SIMM_set1 Soft_errs_allowed *****
41  20068CEC  Board_Reset         *
42  20061880  Chk_for_Interrupts  **********
46  200610C4  P_Cache_diag_mode   bypass_test_mask *********
47  2006AD04  Memory_Refresh      start_a end_incr cont_on_err time_seconds *****
48  2006B028  Memory_Addr_shorts start_add end_add * cont_on_err pat2 pat3 ****
4A  2006A23C  Memory_ECC_SBEs    start_add end_add add_incr cont_on_err ******
4B  2006940C  Memory_Byte_Errors start_add end_add add_incr cont_on_err ******
4C  20069BA0  Memory_ECC_Logic   start_add end_add add_incr cont_on_err ******
4D  20068FE8  Memory_Address     start_add end_add add_incr cont_on_err ******
4E  20069188  Memory_Byte        start_add end_add add_incr cont_on_err ******
4F  2006B7F4  Memory_Data        start_add end_add add_incr cont_on_err ******
51  2005803C  FPA                **********
52  20058530  SSC_Prog_timers    which_timer wait_time_us ***
53  20058818  SSC_TOY_Clock      repeat_test_250ms_ea Tolerance ***
54  20057C18  Virtual_Mode       ********
55  20058E6C  Interval_Timer     *****
56  2006507C  SHAC_LPBCK         From_bus To_bus passes *******
58  20065D24  SHAC_RESET         dssi_bus port_number time_secs not_pres
59  20062778  SGEC_LPBCK_ASSIST  time_secs **
5C  20062D10  SHAC               SHAC_number *********
5F  200619B8  SGEC               loopback_type no_ram_tests ******
62  20058B1C  console_QDSS       mark_not_present selftest_r0 selftest_r1 ******
63  20058CA4  QDSS_any           input_csr selftest_r0 selftest_r1 ******
80  2005D3C0  CQBIC_memory       bypass_test_mask *********
81  200596CC  Qbus_MSCP          IP_csr ******
82  200598AC  Qbus_DELQA         device_num_addr ****
83  2005A85C  QZA_Intlpbck1      controller_number ********
84  2005BF1C  QZA_Intlpbck2      controller_number *********
85  20059A9C  QZA_memory         incr test_pattern controller_number *******
86  20059F44  QZA_DMA            Controller_number main_mem_buf ********
90  20058494  CQBIC_registers    *
```

```
91  20058410  CQBIC_powerup       **
99  2005DC4C  Flush_Ena_Caches    dis_flush_VIC dis_flush_BC dis_flush_PC
9A  20063FB0  INTERACTION         pass_count disable_device ****
9B  20068E48  Init_memory         ***
9C  2006631C  List_CPU_registers  *
9D  2006C250  Utility             Flags *********
9E  2005903C  List_diagnostics    script_number *
9F  200681CC  Create_A0_Script    **********
C1  20057888  SSC_RAM_Data        *
C2  20057A78  SSC_RAM_Data_Addr   *
C5  200589E8  SSC_registers       *
D0  20060C70  V_Cache_diag_mode   bypass_test_mask *********
D2  2005DE90  O_Bit_diag_mode     bypass_test_mask *********
DA  2006139C  PB_Flush_Cache      **********
DB  2005E850  Speed               print_speed *********
DC  2006C060  NO_Memory_present   *
DD  2005F0DC  B_Cache_Data_debug  start_add end_add add_incr *******
DE  2005EC64  B_Cache_Tag_Debug   start_add end_add add_incr *******
DF  2005E2A8  O_BIT_DEBUG         start_add end_add add_incr seg_incr ******
E0  2006D4D4  SCSI                environment reset_bus time_s *******
E1  2006D7CC  SCSI_Utility        environment util_nbr target_ID lun ******
E2  2006DA2C  SCSI_MAP            bypass_test addr_incr_data_tst ********
E4  2006DFC8  DZ                  environment *********
E8  2006E1DC  SYNC                environment *********
E9  2006E2B4  SYNC_Utility        environment *********
EC  2006E398  ASYNC               environment *********
F0  2006D638  SCSI_option         environment reset_bus time_s *******
F1  2006D900  SCSI_Opt_Utility    environment util_nbr target_ID lun ******
F2  2006DA40  SCSI_MAP_Option     bypass_test addr_incr_data_tst ********

Scripts
#   Description

A0  User defined scripts
A1  Powerup tests, Functional Verify, continue on error, numeric countdown
A3  Functional Verify, stop on error, test # announcements
A4  Loop on A3 Functional Verify
A6  Memory tests, mark only multiple bit errors
A7  Memory tests
A8  Memory acceptance tests, mark single and multi-bit errors, call A7
A9  Memory tests, stop on error
B2  Extended tests plus BF, then loop
B5  Extended tests, then loop
BF  DZ, SYNC, ASYNC with loopbacks
```

```
Load & start system exerciser
100 Customer mode, 2 passes
101 CSSE mode, 2 passes
102 CSSE mode, continous until ^C
103 Manuf mode, continous until ^C
104 Manuf TINA mode, continous until ^C
105 Manuf mode, 2 passes
106 CSSE mode, select tests, continous until ^C
107 Manuf mode, select tests, continous until ^C

>>>
>>>
```

3.2.18 UNJAM

The UNJAM command performs an I/O bus reset, by writing a 1 (one) to IPR 55 (decimal). SHAC and SGEC are explicitly reset, EDAL_INTREQ register error bits are cleared and SCSI_DMA map registers are cleared.

Format:

UNJAM

Example:

```
>>>UNJAM
>>>
```

3.2.19 X—Binary Load and Unload

The X command is for use by automatic systems communicating with the console.

The X command loads or unloads (that is, writes to memory, or reads from memory) the specified number of data bytes through the console serial line (regardless of console type) starting at the specified address.

Format:

X {address} {count} CR {line_checksum} {data} {data_checksum}

If bit 31 of the count is clear, data is received by the console and deposited into memory. If bit 31 is set, data is read from memory and sent by the console. The remaining bits in the count are a positive number indicating the number of bytes to load or unload.

The console accepts the command upon receiving the carriage return. The next byte the console receives is the command checksum, which is not echoed. The command checksum is verified by adding all command characters, including the checksum and separating space (but not including the terminating carriage return, rubouts, or characters deleted by rubout), into an 8-bit register initially set to zero. If no errors occur, the result is zero. If the command checksum

is correct, the console responds with the input prompt and either sends data to the requester or prepares to receive data. If the command checksum is in error, the console responds with an error message. The intent is to prevent inadvertent operator entry into a mode where the console is accepting characters from the keyboard as data, with no escape mechanism possible.

If the command is a load (bit 31 of the count is clear), the console responds with the input prompt (>>>), then accepts the specified number of bytes of data for depositing to memory, and an additional byte of received data checksum. The data is verified by adding all data characters and the checksum character into an 8-bit register initially set to zero. If the final content of the register is nonzero, the data or checksum are in error, and the console responds with an error message.

If the command is a binary unload (bit 31 of the count is set), the console responds with the input prompt (>>>), followed by the specified number of bytes of binary data. As each byte is sent, it is added to a checksum register initially set to zero. At the end of the transmission, the two's complement of the low byte of the register is sent.

If the data checksum is incorrect on a load, or if memory or line errors occur during the transmission of data, the entire transmission is completed, then the console issues an error message. If an error occurs during loading, the contents of the memory being loaded are unpredictable.

The console represses echo while it is receiving the data string and checksums.

The console terminates all flow control when it receives the carriage return at the end of the command line in order to avoid treating flow control characters from the terminal as valid command line checksums.

You can control the console serial line during a binary unload using control characters (Ctrl/C, Ctrl/S, Ctrl/O, and so on). You cannot control the console serial line during a binary load, since all received characters are valid binary data.

The console has the following timing requirements:

- It must receive data being loaded with a binary load command at a rate of at least one byte every 60 seconds.
- It must receive the command checksum that precedes the data within 60 seconds of the carriage return that terminates the command line.
- It must receive the data checksum within 60 seconds of the last data byte.

If any of these timing requirements are not met, then the console aborts the transmission by issuing an error message and returning to the console prompt.

The entire command, including the checksum, can be sent to the console as a single burst of characters at the specified character rate of the console serial line. The console is able to receive at least 4 Kbytes of data in a single X command.

3.2.20 ! (Comment)

The comment character (an exclamation point) is used to document command sequences. It can appear anywhere on the command line. All characters following the comment character are ignored.

Format: !

Example:

```
>>>! The console ignores this line.
>>>
```

4

System Initialization and Acceptance Testing (Normal Operation)

This chapter describes the system initialization, testing, and bootstrap processes that occur at power-up. In addition, the acceptance test procedure to be performed when installing a system or whenever adding or replacing FRUs is described.

Note

The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95 and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, will appear on the console and/or printouts from time to time.

4.1 Basic Initialization Flow

On power-up, the firmware identifies the console device, optionally performs a language inquiry, and runs the diagnostics.

The firmware waits for power to stabilize by monitoring SCR<15>(POK). Once power is stable, the firmware verifies that the console battery backup RAM (BBU RAM) is valid (backup battery is charged) by checking SSCCR<31>(BLO). If it is invalid or zero (battery is discharged), the BBU RAM is initialized.

After the battery check, the firmware tries to determine the type of terminal attached to the console serial line. It uses this information to determine if multinational support is appropriate.

The console uses the saved console language if the contents of the BBU RAM are valid.

If the firmware detects that the contents of the BBU RAM are invalid, the firmware prompts you for the language to be used for displaying the following system messages (if the console terminal supports the multinational character set):

```
Loading system software.
Failure.
Restarting system software.
Performing normal system tests.
Tests completed.
Normal operation not possible.
Bootfile.
Memory configuration error.
No default boot device has been specified.
Available devices.
Device?
Retrying network bootstrap.
```

The position of the Break Enable/Disable switch has no effect on these conditions. The firmware will not prompt for a language if the console terminal, such as the VT100, does not support the multinational character set (MCS).

Following a successful diagnostic countdown (see Example 4–1), the console may prompt you for a default boot device.

Example 4–1 Successful Diagnostic Countdown

```
KA50-A VX.X, VMB 2.14
Performing normal system tests.
72..71..70..69..68..67..66..65..64..63..62..61..60..59..58..57..
56..55..54..53..52..51..50..49..48..47..46..45..44..43..42..41..
40..39..38..37..36..35..34..33..32..31..30..29..28..27..26..25..
24..23..22..21..20..19..18..17..16..15..14..13..12..11..10..09..
08..07..06..05..04..03..
Tests completed.
>>>
```

4.2 Power-On Self-Tests (POST)

Power-on self-tests provide core testing of the system kernel comprised of the CPU and memory. Certain registers are flushed, and data structures are set up to initialize and set the system to a known state for the operating system.

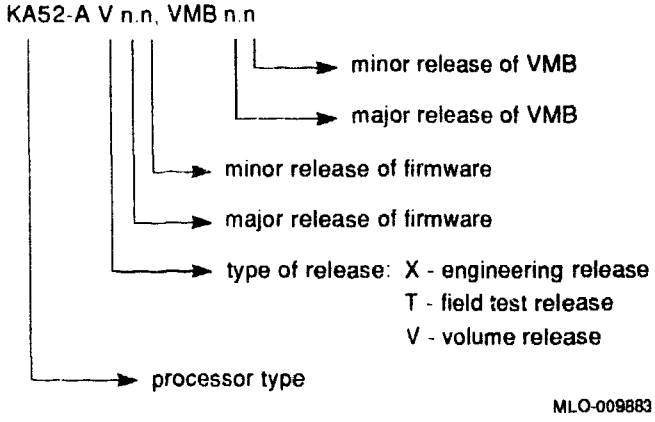
4.2.1 Power-Up Tests for Kernel

In a nonmanufacturing environment where the intended console device is the serial line unit (SLU), the console program performs the following actions at power-up:

1. Checks for POK.
2. Establishes SLU as console device.
3. Prints banner message.

   The banner message contains the processor name, the version of the firmware, and the version of VMB. The letter code in the firmware version indicates if the firmware is pre-field test, field test, or official release. The first digit indicates the major release number and the trailing digit indicates the minor release number (Figure 4–1).

Figure 4–1 Console Banner



4. Displays language inquiry menu on console if console supports multinational character set (MCS) *and any* of the following are true:
   - Battery is dead.
   - Contents of SSC RAM are invalid.
5. Calls the diagnostic executive (DE) with Test Code = 0.
   a. DE executes script A1 (Tests system module and memory).

While the diagnostics are running, the LEDs display a test code. A different countdown appears on the console terminal. Refer to Table 5–4 for a complete explanation of the power-up test display. Table 4–1 lists the LED codes and the associated actions performed at power-up. Example 4–2 shows a successful power-up to a list of bootable devices.

b. DE passes control back to the console program.

6. Issues end message and >>> prompt.

Table 4–1 LED Codes

LED Value	Actions
F	Initial state on power-up, no code has executed
E	Entered ROM space, some instructions have executed
D	Waiting for power to stabilize (POK)
C	SSC RAM, SSC registers, and ROM checksum tests
B	O-bit memory, interval timer, and virtual mode tests
A	FPA tests
9	Backup cache tests
8	NMC, NCA, memory, and I/O interaction tests
7	CQBIC, SYNC, and ASYNC tests
6	Console and QUART tests
5	SCSI tests
4	SGEC Ethernet subsystem tests
3	"Console I/O" mode
2	Control passed to VMB
1	Control passed to secondary bootstrap
0	"Program I/O" mode, control passed to operating system

Example 4–2 Successful Power-Up to List of Bootable Devices

```
KA50-A VX.X, VMB 2.14
Performing normal system tests.
72..71..70..69..68..67..66..65..64..63..62..61..60..59..58..57..
56..55..54..53..52..51..50..49..48..47..46..45..44..43..42..41..
40..39..38..37..36..35..34..33..32..31..30..29..28..27..26..25..
24..23..22..21..20..19..18..17..16..15..14..13..12..11..10..09..
08..07..06..05..04..03..
Tests completed.
Loading system software.
No default boot device has been specified.
Available devices.
-DIA0 (RF73)
-DIA1 (RF73)
-MIA5 (TF85)
-EZA0 (08-00-2B-06-10-42)
Device? [EZA0]:
```

4.2.2 Power-Up Tests for Mass Storage Devices

An RZ-series ISE may fail either during initial power-up or during normal operation. In both cases, the failure is indicated by the lighting of the red fault LED on the drive's front panel. The ISE also has a red fault LED, but it is not visible from the outside of the system enclosure.

If the drive is unable to execute the Power-On Self-Test (POST) successfully, the red fault LED remains lit and the ready LED does not come on, or both LEDs remain on.

POST is also used to handle two types of error conditions in the drive:

- *Controller errors* are caused by the hardware associated with the controller function of the drive module. A controller error is fatal to the operation of the drive, since the controller cannot establish a logical connection to the host. The red fault LED lights. If this occurs, replace the drive module.
- *Drive errors* are caused by the hardware associated with the drive control function of the drive module. These errors are not fatal to the drive, since the drive can establish a logical connection and report the error to the host. Both LEDs go out for about 1 second, then the red fault LED lights.

4.3 CPU ROM-Based Diagnostics

The KA50/51/55/56 ROM-based diagnostic facility is the primary diagnostic tool for troubleshooting and testing of the CPU, memory, and Ethernet. ROM based diagnostics have significant advantages:

- Load time is virtually nonexistent.
- The boot path is more reliable.
- Diagnosis is done in a more primitive state.

The ROM-based diagnostics can detect failures in field-replaceable units (FRUs) other than the CPU module. For example, they can isolate to two memory SIMMS. (Table 5–4 lists the FRUs indicated by ROM-based diagnostic error messages.)

The diagnostics run automatically on power-up. While the diagnostics are running, the LED displays a hexadecimal number; while booting the operating system, 2 through 0 display.

The ROM-based diagnostics are a collection of individual tests with parameters that you can specify. A data structure called a *script* points to the tests (see Section 4.3.2). There are several field and manufacturing scripts.

A program called the *diagnostic executive* determines which of the available scripts to invoke. The script sequence varies if the system is in the manufacturing environment. The diagnostic executive interprets the script to determine what tests to run, the correct order to run the tests, and the correct parameters to use for each test.

The diagnostic executive also controls tests so that errors can be detected and reported. It ensures that when the tests are run, the machine is left in a consistent and well-defined state.

4.3.1 Diagnostic Tests

Example 4–3 shows a list of the ROM-based tests and utilities. To get this listing, enter T 9E at the console prompt (T is the abbreviation of TEST). The column headings have the following meanings:

Note

Base addresses shown in this document may not be the same as the addresses you see when you run T 9E. Run T 9E to get a list of actual addresses. See Example 4–3.

- Test is the test number or utility code.
- Address is the base address of where the test or utility starts in ROM. If a test fails, entering T FE displays diagnostic state to the console. You can subtract the base address of the failing test from the last_exception_pc to find the index into the failing test's diagnostic listing.
- Name is a brief description of the test or utility.
- Parameters shows the parameters for each diagnostic test or utility. These parameters are encoded in ROM and are provided by the diagnostic executive. Tests accept up to 10 parameters. The asterisks (*) represent parameters that are used by the tests but that you cannot specify individually. These parameters are displayed in error messages, each one preceded by identifiers P1 through P10.

Example 4–3 Test 9E

```
>>>T 9E

Test
#   Address   Name                Parameters
-------------------------------------------------------------------------
    20051200  SCB
    20054D28  De_executive
30  200645A4  Memory_Init_Bitmap *** mark_Hard_SBEs ******
31  20064E9C  Memory_Setup_CSRs  **********
32  20065288  NMC_registers      **********
33  20065440  NMC_powerup        **
34  2005DB60  SSC_ROM            ***
35  200682F4  B_Cache_diag_mode  bypass_test_mask *********
37  200691EC  Cache_w_Memory     bypass_test_mask *********
40  200631F4  Memory_count_pages SIMM_set0 SIMM_set1 Soft_errs_allowed *****
41  200581F8  Board_Reset        *
42  2005BEF0  Chk_for_Interrupts **********
46  2006801C  P_Cache_diag_mode  bypass_test_mask *********
47  20064B84  Memory_Refresh     start_a end incr cont_on_err time_seconds *****
48  200622E4  Memory_Addr_shorts start_add end_add * cont_on_err pat2 pat3 ****
4A  200642C8  Memory_ECC_SBEs    start_add end_add add_incr cont_on_err ******
4B  20062B24  Memory_Byte_Errors start_add end_add add_incr cont_on_err ******
4C  20063C70  Memory_ECC_Logic   start_add end_add add_incr cont_on_err ******
4D  20062144  Memory_Address     start_add end_add add_incr cont_on_err ******
4E  200628A0  Memory_Byte        start_add end_add add_incr cont_on_err ******
4F  20063408  Memory_Data        start_add end_add add_incr cont_on_err ******
51  2005C408  FPA                **********
52  2005C8C4  SSC_Prog_timers    which_timer wait_time_us ***
53  2005CBA8  SSC_TOY_Clock      repeat_test_250ms_ea Tolerance ***
54  2005C008  Virtual_Mode       ********
55  2005CD74  Interval_Timer     *****
58  20061060  SHAC_RESET         port_number time_secs not_pres
59  200602AC  SGEC_LPBCK_ASSIST  time_secs **
5C  2006082C  SHAC               bypass_test_mask *******
5F  2005F52C  SGEC               loopback_type no_ram_tests ******
63  2005D99C  QDSS_any           input_csr selftest_r0 selftest_r1 ******
80  20065884  CQBIC_memory       bypass_test_mask *********
81  2005D5D0  Qbus_MSCP          IP_csr ******
82  2005D7AC  Qbus_DELQA         device_num_addr ****
83  20059570  QZA_Intlpbck1      controller_number ********
84  2005AC74  QZA_Intlpbck2      controller_number *********
85  2005877C  QZA_memory         incr test_pattern controller_number *******
86  20058C74  QZA_DMA            Controller_number main_mem_buf ********
90  2005C82C  CQBIC_registers    *
91  2005C7A8  CQBIC_powerup      **
99  20065644  Flush_Ena_Caches   dis_flush_VIC dis_flush_BC dis_flush_PC
9A  2005DCB4  INTERACTION        pass_count disable_device ****
9B  200654DC  Init_memory        ***
9C  2005DC80  List_CPU_registers *
9D  2005EB60  Utility            Modify_CPU_type *********
9E  2005CF40  List_diagnostics   script_number *
9F  20061610  Create_A0_Script   **********
```

(continued on next page)

Example 4–3 (Cont.) Test 9E

```
C1  200583F8  SSC_RAM_Data          *
C2  200585E8  SSC_RAM_Data_Addr     *
C5  2005F414  SSC_registers         *
C6  20058320  SSC_powerup           *********
D0  20067BC8  V_Cache_diag_mode     bypass_test_mask *********
D2  200660E8  O_Bit_diag_mode       bypass_test_mask *********
DA  20068FFC  PB_Flush_Cache        **********
DB  20066908  Speed                 print_speed *********
DC  200650C8  NO_Memory_present     *
DD  20067118  B_Cache_Data_debug    start_add end_add add_incr *******
DE  20066CB8  B_Cache_Tag_Debug     start_add end_add add_incr *******
DF  200664F0  O_BIT_DEBUG           start_add end_add add_incr seq_incr ******
E0  200694D8  SCSI                  environment reset_bus time_s *******
E1  200695D8  SCSI_Utility          environment util_nbr target_ID lun ******
E2  200696A4  SCSI_MAP              bypass_test addr_incr_data_tst ********
E4  20069A60  DZ                    environment *********
E8  20069BF0  SYNC                  environment *********
E9  20069CC4  SYNC_Utility          environment *********
EC  20069DA8  ASYNC                 environment *********

Scripts
#   Description

A0  User defined scripts
A1  Powerup tests, Functional Verify, continue on error, numeric countdown
A3  Functional Verify, stop on error, test # announcements
A4  Loop on A3 Functional Verify
A6  Memory tests, mark only multiple bit errors
A7  Memory tests
A8  Memory acceptance tests, mark single and multi-bit errors, call A7
A9  Memory tests, stop on error
B2  Extended tests plus BF, then loop
B5  Extended tests, then loop
BF  DZ, SYNC, ASYNC with loopbacks

Load & start system exerciser
100 Customer mode, 2 passes
101 CSSE mode, 2 passes
102 CSSE mode, continous until ^C
103 Manuf mode, continous until ^C
104 Manuf TINA mode, continous until ^C
105 Manuf mode, 2 passes
106 CSSE mode, select tests, continous until ^C
107 Manuf mode, select tests, continous until ^C

>>>
```

User Determined Parameters

Parameters that you can specify are written out, as shown in the following examples:

```
30  2005C33C  Memory_Init_Bitmap *** mark_Hard_SBEs ******
54  20055181  Virtual_Mode       *********
```

For example, the virtual mode test contains several parameters, but you cannot specify any that appear in the table as asterisks. To run this test individually, enter:

```
>>>T 54
```

The MEM_bitmap test, for example, accepts 10 parameters, but you can only specify mark_hard_SBEs because the rest are asterisks. To map out solid, single-bit ECC memory errors, type:

```
>>>T 30 0 0 0 1
```

Even though you cannot change the first three parameters, you need to enter zeros (0) as placeholders. The zeros are placeholders for parameters 1 through 3, which allows the program to parse the command line correctly. The diagnostic executive then provides the proper value for the test.

You enter 1 for parameter 4 to indicate that the test should map out solid, single-bit as well as multibit ECC memory errors. You then terminate the command line by pressing RETURN. You do not need to specify parameters 5 through 10; placeholders are needed only for parameters that precede the user-definable parameter.

For the most part tests and scripts can be run without any special setup. If a test or script is run interactively without an intervening power up, such as after a system crash or shutdown, enter the UNJAM and INIT commands before running the tests or script. This will ensure that the CPU is in a well known state. If the commands are not entered, misleading errors may occur.

Other considerations to be aware of when running individual tests or scripts interactively:

- When using the TEST or REPEAT TEST commands, you must specify a test number, test code or script number following the TEST command before pressing RETURN.
- The memory bitmap and Q–bus scatter-gather map are created in main memory and the memory tests are run with these data structures left intact. Therefore, the upper portion of memory should not be accessed to avoid corrupting these data structures. The location of the maps is displayed using the SHOW MEMORY/FULL command.

4.3.2 Scripts

Most of the tests shown by utility 9E are arranged into scripts. A *script* is a data structure that points to various tests and defines the order in which they are run. Scripts should be thought of as diagnostic tables—these tables do not contain the actual diagnostic tests themselves, instead scripts simply define what tests or scripts should be run, the order that the tests or scripts should be run, and any input parameters to be parsed by the Diagnostic Executive.

Different scripts can run the same set of tests, but these tests can be run in a different order and/or with different parameters and flags. A script also contains the following information:

- The parameters and flags that need to be passed to the test.
- The locations from which the tests can be run. For example, certain tests can be run only from the FEPROM. Other tests are program-independent code, and can be run from FEPROM or main memory to enhance execution speed.
- What is to be shown, if anything, on the console.
- What is to be shown, if anything, in the LED display.
- What action to take on errors (halt, repeat, continue).

The power-up script runs every time the system is powered on. You can also invoke the power-up script at any time by entering T 0.

Additional scripts are included in the FEPROMs for use in manufacturing and engineering environments. Customer Services personnel can run these scripts and tests individually, using the T command. When doing so, note that certain tests may be dependent upon a state set up from a previous test. For this reason, use the UNJAM and INITIALIZE commands before running an individual test. You do not need these commands on system power-up because the system power-up leaves the machine in a defined state.

Customer Services Engineers (CSE) with a detailed knowledge of the system hardware and firmware can also create their own scripts by using the 9F User Script Utility. Table 4–2 lists the scripts available to Customer Services.

Table 4–2 Scripts Available to Customer Services

Script[1]	Enter with TEST Command	Description
A0	A0	Runs user-defined script. Enter T 9F to create.
A1	A1, 0	Primary power-up script; builds memory bitmap; marks hard single-bit errors and multi-bit errors. Continues or. error.
A3	A3, A4	Runs power-up tests, halts on first error.
A4	A4	Loops on A3. Press Ctrl C to exit.
A6	A6	Memory test script; initializes memory bitmap and marks only multiple bit errors.
A7	A7, A8	Memory test portion invoked by script A8. Reruns the memory tests without rebuilding and reinitializing the bitmap. Run script A8 once before running script A7 separately to allow mapping out of both single-bit and double-bit main memory ECC errors.
A8	A8	Memory acceptance. Running script A8 with script A7 tests main memory more extensively. It enables hard single-bit and multibit main memory ECC errors to be marked bad in the bitmap. Invokes script A7 when it has completed its tests.
A9	A9	Memory tests. Halts and reports the first error. Does not reset the bitmap or busmap. It is a quick way to specify which test caused a failure when a hard error is present.
AD	AD	Console program. Runs memory tests, marks bitmap, resets busmap, and resets caches. Calls script AE.
AE	AE, AD	Console program. Resets memory CSRs and resets caches. Also called by the INIT command.
AF	AF	Console program. Resets busmap and resets caches.
B2[2]	B2	Runs extended tests, calls the BF script, then loops. Press Ctrl C to exit.
B5	B5	Runs extended tests, then loops. Press Ctrl C to exit.
BF[2]	BF	Runs tests requiring loopback connectors for QUART, SYNC, and, ASYNC options if present. Press Ctrl C to exit.

[1]Scripts AD, AE, and AF exist primarily for console program; error displays and progress messages are suppressed (not recommended for CSE use).

[2]B2 and BF require loopback connectors.

4.4 Basic Acceptance Test Procedure

Perform the acceptance testing procedure listed below, after installing a system, or whenever adding or replacing the following:

CPU module
MS44 memory SIMM
SCSI device
SYNC device
ASYNC device

1. Run two error-free passes of the power-up scripts by entering the following command:

```
>>>T B5
```

Script B5 will halt on an error so that the error message will not scroll off the screen.

Press Ctrl/C to terminate the scripts. Refer to Chapter 5 if failures occur.

To check the memory configuration and to ensure there are no bad pages, enter the following command line:

```
>>>SHOW MEM/FULL

16 MB RAM, SIMM Set (0A,0B,0C,0D) present
Memory Set 0: 00000000 to 00FFFFFF, 16MB, 32768 good pages, 0 bad pages

Total of 16MB, 32768 good pages, 0 bad pages, 104 reserved pages

Memory Bitmap
-00FF3000 to 00FF3FFF, 8 pages

Console Scratch Area
-00FF4000 to 00FF7FFF, 32 pages

Scan of Bad Pages

Q-bus Map
-01FF8000 to 01FFFFFF, 64 pages

Scan of Bad Pages

>>>
```

The Q22–bus map always spans the top 32 Kbytes of good memory. The memory bitmap always spans two pages (1 Kbyte) for each 4 Mbytes of memory configured. Each bit within the memory bit map represents a page of memory.

To identify registers and register bit fields, see the *KA50/51/55/56 CPU Technical Manual.*

Examine MEMCON 0–1 to verify the memory configuration. Each pair of MEMCONs maps one memory module as follows:

MEMCON0	Set 0; 0A, 0B, 0C, 0D
MEMCON1	Set 1; 1E, 1F, 1G, 1H

4.5 Machine State on Power-Up

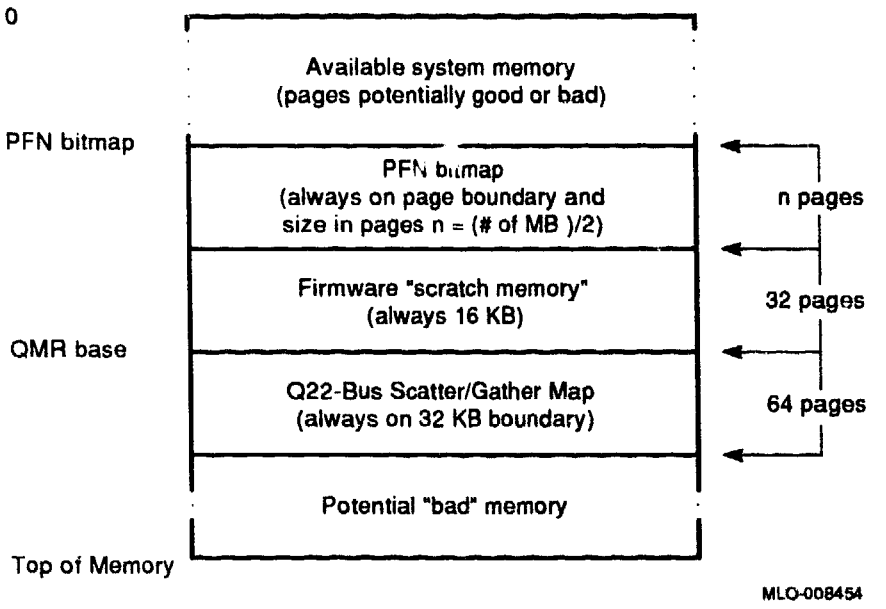
This section describes the state of the kernel after a power-up halt.

The descriptions in this section assume the system has just powered-up and the power-up diagnostics have successfully completed. The state of the machine is not defined if individual diagnostics are run or for any other halts other than a power-up halt (SAVPSL<13:8>(RESTART_CODE) = 3). Refer to Appendix D for a description of the normal state of CPU configurable bits following completion of power-up tests.

4.6 Main Memory Layout and State

Main memory is tested and initialized by the firmware on power-up. Figure 4–2 is a diagram of how main memory is partitioned after diagnostics.

Figure 4–2 Memory Layout After Power-Up Diagnostics



4.6.1 Reserved Main Memory

In order to build the scatter/gather map and the bitmap, the firmware attempts to find a physically contiguous page-aligned 1M byte block of memory at the highest possible address.

Of the 1M byte, the upper 32 KB is dedicated to the Q22–bus scatter/gather map, as shown in Figure 4–2. Of the lower portion, up to 32K bytes at the bottom of the block is allocated to the Page Frame Number (PFN) bitmap. The size of the PFN bitmap is dependent on the extent of physical memory. Each bit in the bitmap maps one page (512 bytes) of memory. The remainder of the block between the bitmap and scatter/gather map (minimally 16 KB) is allocated for the firmware.

4.6.1.1 PFN Bitmap

The PFN bitmap is a data structure that indicates which pages in memory are deemed usable by operating systems. The bitmap is built by the diagnostics as a side effect of the memory tests on power-up. The bitmap always starts on a page boundary. The bitmap requires 1 KB for every 4 MB of main memory, hence, a 8 MB system requires 2 KB, 16 MB requires 4 KB, 32 MB requires 8 KB, and a 64 MB requires 16 KB. There may be memory above the bitmap which has both good and bad pages.

Each bit in the PFN bitmap corresponds to a page in main memory. There is a one to one correspondence between a page frame number (origin 0) and a bit index in the bitmap. A one in the bitmap indicates that the page is "good" and can be used. A zero indicates that the page is "bad" and should not be used.

The PFN bitmap is protected by a checksum stored in the NVRAM. The checksum is a simple byte wide, two's complement checksum. The sum of all bytes in the bitmap and the bitmap checksum should result in zero.

4.6.1.2 Scatter/Gather Map

On power-up, the scatter/gather map is initialized by the firmware to map to the first 4M bytes of main memory. Main memory pages will not be mapped if there is a corresponding page in Q22–bus memory.

On a processor halt other than power-up, the contents of the scatter/gather map is undefined, and is dependent on operating system usage.

Operating systems should not move the location of the scatter/gather map, and should access the map only on aligned longwords through the local I/O space of 20088000 to 2008FFFC, inclusive. The Q22–bus map base register (QMBR), is set up by the firmware to point to this area, and should not be changed by software.

4.6.1.3 Firmware "Scratch Memory"

This section of memory is reserved for the firmware. However, it is only used after successful execution of the memory diagnostics and initialization of the PFN bitmap and scatter/gather map. This memory is primarily used for diagnostic purposes.

4.6.2 Contents of Main Memory

The contents of main memory are undefined after the diagnostics have run. Typically, nonzero test patterns will be left in memory.

The diagnostics will "scrub" all of main memory, so that no power-up induced errors remain in the memory system. On the KA50/51/55/56 memory subsystem, the state of the ECC bits and the data bits are undefined on initial power-up. This can result in single and multiple bit errors if the locations are read before written, because the ECC bits are not in agreement with their corresponding data bits. An aligned longword write to every location (done by diagnostics) eliminates all power-up induced errors.

4.6.3 Memory Controller Registers

The SHOW MEMORY command defines the mapping of addresses to specific SIMM sets as follows:

- MEMCON0 is used with SIMM bank 0 (the 0A, 0B, 0C, and 0D memory slots)
- MEMCON1 is used with SIMM bank 1 (the 1E, 1F, 1G, and 1H memory slots)

Additional information should be captured from the NMCDSR, MOAMR, MSER, and MEAR as needed.

4.6.4 On-Chip and Backup Caches

All three caches are tested.

4.6.5 Translation Buffer

The CPU translation buffer is tested by diagnostics on power-up, but not used by the firmware because it runs in physical mode. The translation buffer can be invalidated by using PR$_TBIA, IPR 57.

4.6.6 Halt-Protected Space

On the KA50/51/55/56, halt-protected space spans the first half of the 512K byte FEPROM from 20040000 to 2007FFFF. The second half of the FEPROM has data which is loaded into memory and run.

The firmware always runs in halt-protected space. When passing control to the bootstrap, the firmware exits the halt-protected space, so if halts are enabled, and the halt line is asserted, the processor will then halt before booting.

4.7 Operating System Bootstrap

Bootstrapping is the process by which an operating system loads and assumes control of the system. The KA50/51/55/56 supports bootstrap of the VAX/OpenVMS and VAXELN operating systems. Additionally, the KA50/51 /55 will boot MDM diagnostics and any user application image which conforms to the boot formats described herein.

On the KA50/51/55/56 a bootstrap occurs whenever a BOOT command is issued at the console or whenever the processor halts and the conditions specified in Table G–1 for automatic bootstrap are satisfied.

4.7.1 Preparing for the Bootstrap

Prior to dispatching to the primary bootstrap (VMB), the firmware initializes the system to a known state. The initialization sequence follows:

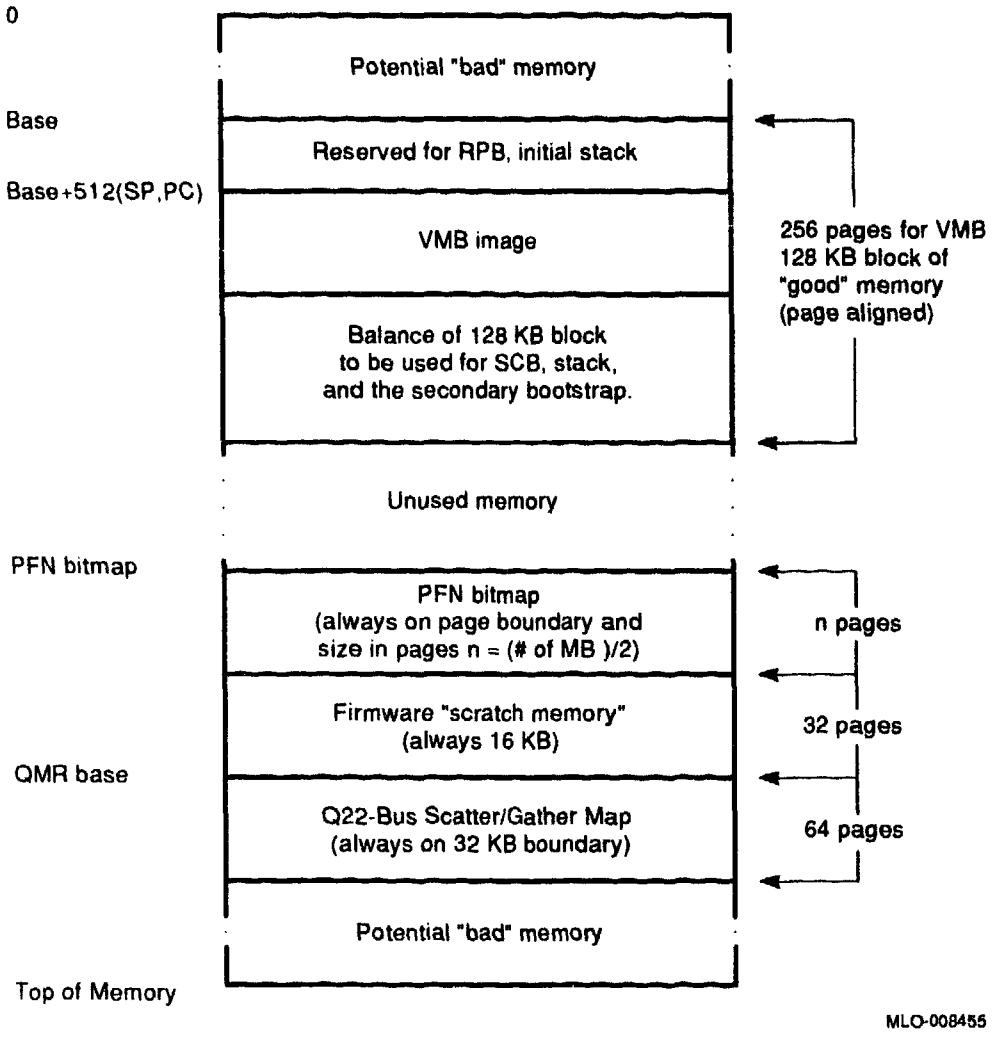
1. Check the console program mailbox "bootstrap in progress" bit (CPMBX<2>(BIP)). If it is set, bootstrap fails.
2. If this is an automatic bootstrap, display the message "Loading system software." on the console terminal.
3. Set CPMBX<2>(BIP).
4. Validate the Page Frame Number (PFN) bitmap. If PFN bitmap checksum is invalid, then:
   a. Perform an UNJAM.
   b. Perform an INIT.
   c. Retest memory and rebuild PFN bitmap.
5. Validate the boot device name. If none exists, supply a list of available devices and prompt user for a device. If no device is entered within 30 seconds, use EZA0.
6. Write a form of this BOOT request including the active boot flags and boot device on the console, for example "(BOOT/R5:0 DUA0)".
7. Initialize the Q22–bus scatter/gather map.
   a. Set IPCR<8>(AUX_HLT).
   b. Clear IPCR<5>(LMEAE).
   c. Perform an UNJAM.
   d. Perform an INIT.
   e. If an arbiter, map all vacant Q22–bus pages to the corresponding page in local memory and validate each entry if that page is "good".
   f. Set IPCR<5>(LMEAE).
8. Search for a 128K byte contiguous block of good memory as defined by the PFN bitmap. If 128K bytes cannot be found, the bootstrap fails.
9. Initialize the general purpose registers as follows:

R0	Address of descriptor of boot device name; 0 if none specified
R2	Length of PFN bitmap in bytes
R3	Address of PFN bitmap
R4	Time-of-day of bootstrap from PR$_TODR
R5	Boot flags
R10	Halt PC value
R11	Halt PSL value (without halt code and map enable)
AP	Halt code
SP	Base of 128-Kbyte good memory block + 512
PC	Base of 128-Kbyte good memory block + 512
R1, R6, R7, R8, R9, FP	0

10. Copy the VMB image from FEPROM to local memory beginning at the base of the 128 KB good memory block + 512.
11. Exit from the firmware to memory resident VMB.

On entry to VMB the processor is running at IPL 31 on the interrupt stack with memory management disabled. Also, local memory is partitioned as shown in Figure 4–3.

Figure 4–3 Memory Layout Prior to VMB Entry



4.7.2 Primary Bootstrap Procedures (VMB)

Virtual Memory Boot (VMB) is the primary bootstrap for booting VAX processors. On the KA50/51/55/56 module, VMB is resident in the firmware and is copied into main memory before control is transferred to it. VMB then loads the secondary bootstrap image and transfers control to it.

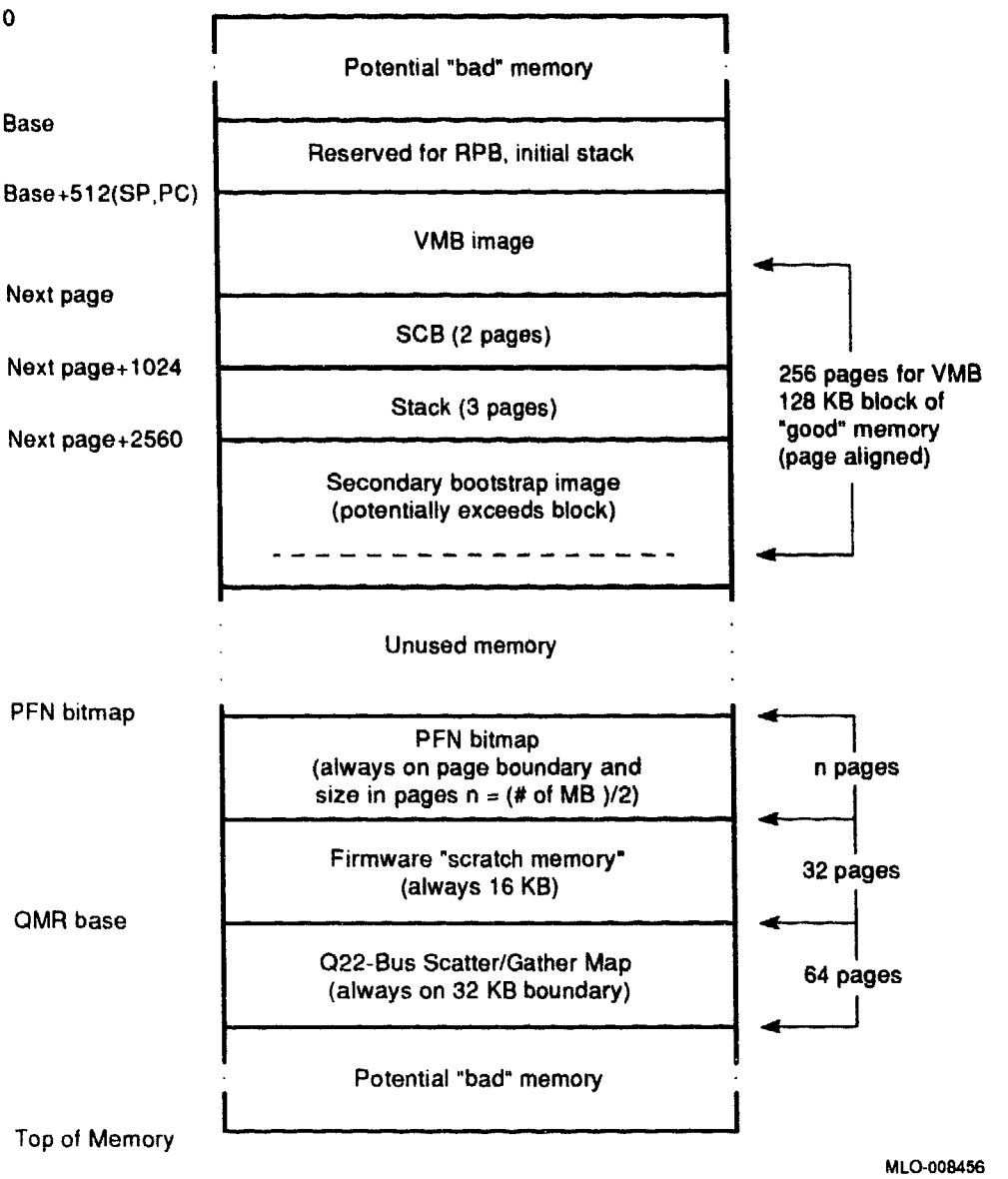
In certain cases, such as VAXELN, VMB actually loads the operating system directly. However, for the purpose of this discussion "secondary bootstrap" refers to any VMB loadable image.

VMB inherits a well defined environment and is responsible for further initialization. The following summarizes the operation of VMB.

1. Initialize a two-page SCB on the first-page boundary above VMB.
2. Allocate a three-page stack above the SCB.
3. Initialize the Restart Parameter Block (RPB).
4. Initialize the secondary bootstrap argument list.
5. If not a PROM boot, locate a minimum of three consecutive valid QMRs.
6. Write "2" to the diagnostic LEDs and display "2.." on the console to indicate that VMB is searching for the device.
7. Optionally, solicit from the console a "Bootfile: " name.
8. Write the name of the boot device from which VMB will attempt to boot on the console, for example, "-DIA0".
9. Copy the secondary bootstrap from the boot device into local memory above the stack. If this fails, the bootstrap fails.
10. Write "1" to the diagnostic LEDs and display "1.." on the console to indicate that VMB has found the secondary bootstrap image on the boot device and has loaded the image into local memory.
11. Clear CPMBX<2>(BIP) and CPMBX<3>(RIP).
12. Write "0" to the diagnostic LEDs and display "0.." on the console to indicate that VMB is now transferring control to the loaded image.
13. Transfer control to the loaded image with the following register usage.

R5	Transfer address in secondary bootstrap image
R10	Base address of secondary bootstrap memory
R11	Base address of RPB
AP	Base address of secondary boot parameter block
SP	Base address of secondary boot parameter block

Figure 4–4 Memory Layout at VMB Exit



In the event that an operating system has an extraordinarily large secondary bootstrap which overflows the 128 KB of "good" memory, VMB loads the remainder of the image in memory above the "good" block. However, if there are not enough contiguous "good" pages above the block to load the remainder of the image, the bootstrap fails.

4.7.3 Device Dependent Secondary Bootstrap Procedures

The following sections describe the various device dependent boot procedures.

4.7.3.1 Disk and Tape Bootstrap Procedure

The disk and tape bootstrap supports Files–11 lookup (supporting only the ODS level 2 file structure) or the boot block mechanism (used in PROM boot also). Of the standard DEC operating systems, OpenVMS and ELN use the Files–11 bootstrap procedure, and Ultrix-32 uses the boot block mechanism.

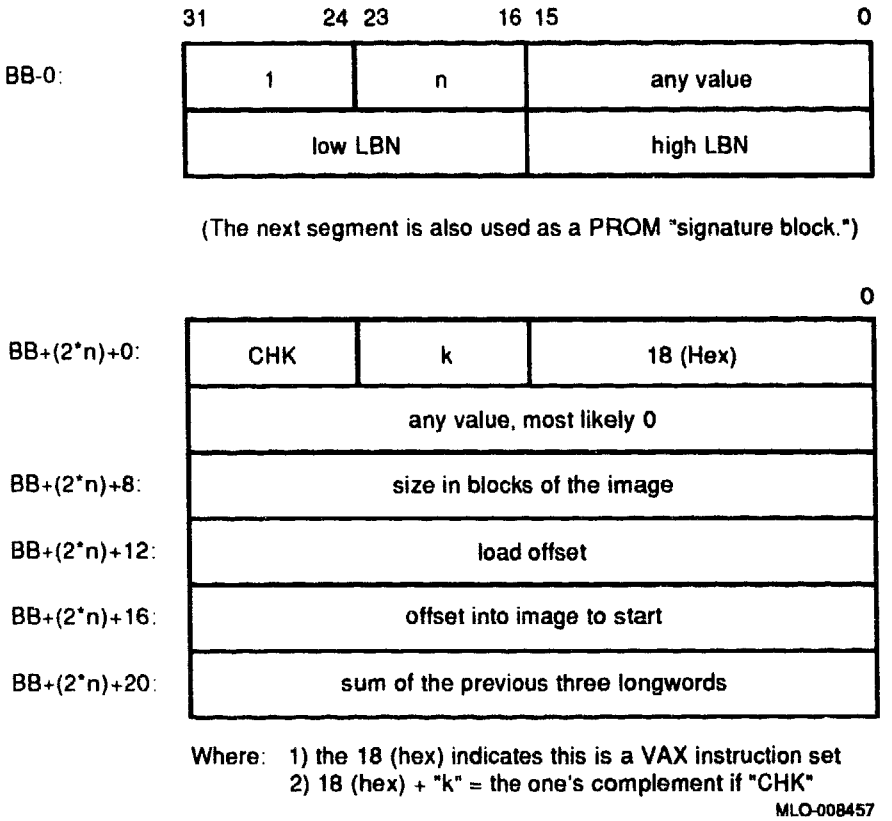
VMB first attempts a Files–11 lookup, unless the RPB$V_BBLOCK boot flag is set. If VMB determines that the designated boot disk is a Files–11 volume, it searches the volume for the designated boot program, usually [SYS0.SYSEXE]SYSBOOT.EXE. However, VMB can request a diagnostic image or prompt the user for an alternate file specification. If the boot image cannot be found, VMB fails.

If the volume is not a Files–11 volume or the RPB$V_BBLOCK boot flag was set, the boot block mechanism proceeds as follows:

1. Read logical block 0 of the selected boot device (this is the boot block).
2. Validate that the contents of the boot block conform to the boot block format (see below).
3. Use the boot block to find and read in the secondary bootstrap.
4. Transfer control to the secondary bootstrap image, just as for a Files–11 boot.

The format of the boot block must conform to that shown in Figure 4–5.

Figure 4–5 Boot Block Format



4.7.3.2 MOP Ethernet Functions and Network Bootstrap Procedure

Whenever a network bootstrap is selected on the KA50/51/55/56, the VMB code makes continuous attempts to boot from the network. VMB uses the DNA Maintenance Operations Protocol (MOP) as the transport protocol for network bootstraps and other network operations. Once a network boot has been invoked, VMB turns on the designated network link and repeats load attempt, until either a successful boot occurs, a fatal controller error occurs, or VMB is halted from the operator console.

The KA50/51/55/56 supports the load of a standard operating system, a diagnostic image, or a user-designated program via network bootstraps. The default image is the standard operating system, however, a user may select an alternate image by setting either the RPB$V_DIAG bit or the RPB$V_SOLICT bit in the boot flag longword R5. Note that the RPB$V_SOLICT bit has precedence over the RPB$V_DIAG bit. Hence, if both bits are set, then the solicited file is requested.

Note

VMB accepts a maximum 39 characters for a file specification for solicited boots. However, MOP V3 only supports a 15-character file name. If the network server is running the OpenVMS operating system, the following defaults apply to the file specification: the directory MOM$LOAD:, and the extension .SYS. Therefore, the file specification need only consist of the filename if the default directory and extension attributes are used.

The KA50/51/55/56 VMB uses the MOP program load sequence for bootstrapping the module and the MOP "dump/load" protocol type for load related message exchanges. The types of MOP message used in the exchange are listed in Table 4–3 and Table 4–4.

Table 4–3 Network Maintenance Operations Summary

Function	Role	Transmit		Receive
		MOP Ethernet and IEEE 802.3 Messages[1]		
Dump	Requester	—		—
	Server	—		—
Load	Requester	REQ_PROGRAM[2]	to solicit	VOLUNTEER
		REQ_MEM_LOAD	to solicit & ACK	MEM_LOAD
			or	MEM_LOAD_w_XFER
			or	PARAM_LOAD_w_XFER
	Server	—		—
Console	Requester	—		—
	Server	COUNTERS	in response to	REQ_COUNTERS
		SYSTEM_ID[3]	in response to	REQUEST_ID
				BOOT
Loopback	Requester	—		—
	Server	LOOPED_DATA[4]	in response to	LOOP_DATA
		IEEE 802.3 Messages[5]		
Exchange ID	Requester	—		—
	Server	XID_RSP	in response to	XID_CMD
Test	Requester	—		—

[1]All unsolicited messages are sent in Ethernet (MOP V3) and IEEE 802.2 (MOP V4), until the MOP version of the server is known. All solicited messages are sent in the format used for the request.

[2]The initial REQ_PROGRAM message is sent to the dumpload multicast address. If an assistance VOLUNTEER message is received, then the responder's address is used as the destination to repeat the REQ_PROGRAM message and for all subsequent REQ_MEM_LOAD messages.

[3]SYSTEM_ID messages are sent out every 8 to 12 minutes to the remote console multicast address and, on receipt of a REQUEST_ID message, they are sent to the initiator.

[4]LOOPED_DATA messages are sent out in response to LOOP_DATA messages. These messages are actually in Ethernet LOOP TEST format, not in MOP format, and when sent in Ethernet frames, omit the additional length field (padding is disabled).

[5]IEEE 802.2 support of XID and TEST is limited to Class 1 operations.

(continued on next page)

Table 4–3 (Cont.) Network Maintenance Operations Summary

Function	Role	Transmit		Receive
		IEEE 802.3 Messages[5]		
	Server	TEST_RSP	in response to	TEST_CMD

[5]IEEE 802.2 support of XID and TEST is limited to Class 1 operations.

Table 4–4 Supported MOP Messages

Message Type	Message Fields						
	DUMP/LOAD						
MEM_LOAD_w_ XFER	Code 00	Load # nn	Load addr aa-aa-aa-aa		Image data None		Xfer addr aa-aa-aa-aa
MEM_LOAD	Code 02	Load # nn	Load addr aa-aa-aa-aa		Image data dd-...		
REQ_PROGRAM	Code 08	Device 25 LQA 49 SGEC	Format 01 V3 04 V4	Program 02 Sys	SW ID[3] C-17 [1] C-128 [2] If C[1] >00 Len 00 No ID FF OS FE Maint	Procesr 00 Sys	Info (see SYSTEM_ ID)
REQ_MEM_ LOAD	Code 0A	Load # nn	Error ee				

[1]MOP V3.0 only.

[2]MOP x4.0 only.

[3]Software ID field is loaded from the string stored in the 40-byte field, RPB$T_FILE, of the RPB on a solicited boot.

(continued on next page)

Table 4–4 (Cont.) Supported MOP Messages

Message Type	Message Fields						
DUMP/LOAD							
PARM_LOAD_w_ XFER	Code 14	Load # nn	Prm typ 01 02 03 04 05 06 00 End	Prm len I-16 I-06 I-16 I-06 0A 08	Prm val Target name [1] Target addr [1] Host name [1] Host addr [1] Host time [1] Host time [2]		Xfer addr aa-aa-aa-aa
VOLUNTEER	Code 03						
REMOTE CONSOLE							
REQUEST_ID	Code 05	Rsrvd xx	Recpt # nn-nn				
SYSTEM_ID	Code 07	Rsrvd xx	Recpt # nn-nn or 00-00	Info type 01-00 Version 02-00 Functions 07-00 HW addr 64-00 Device 90-01 Datalink 91-01 Bufr size		Info len 03 02 06 01 01 02	Info value 04-00-00 00-59 ee-ee-ee-ee-ee-ee 25 or 49 01 06-04
REQ_COUNTERS	Code 09	Recpt # nn-nn					
COUNTERS	Code 0B	Recpt # nn-nn	Counter block				
BOOT [4]	Code 06	Verification vv-vv-vv-vv-vv-vv-vv-vv	Procesr 00 Sys	Control xx	Dev ID C-17	SW ID [3] (see REQ_PROGRAM)	Script ID [2] C-128

[1]MOP V3.0 only.

[2]MOP x4.0 only.

[3]Software ID field is loaded from the string stored in the 40-byte field, RPB$T_FILE, of the RPB on a solicited boot.

[4]A BOOT message is not verified, because in this context, a boot is already in progress. However, a received BOOT message will cause the boot backoff timer to be reset to it's minimum value.

(continued on next page)

Table 4–4 (Cont.) Supported MOP Messages

Message Type	Message Fields				
LOOPBACK					
LOOP_DATA	Skpcnt nn-nn	Skipped bytes bb-...	Function 00-02 Forward data	Forward addr ee-ee-ee-ee-ee-ee	Data dd-...
LOOPED_DATA	Skpcnt nn-nn	Skipped bytes bb-...	Function 00-01 Reply	Recpt # nn-nn	Data dd-...
IEEE 802.2					
XID_CMD/RSP	Form 81	Class 01	Rx window size (K) 00		
TEST_CMD/RSP	Optional data				

VMB, the requester, starts by sending a REQ_PROGRAM message to the MOP 'dump/load' multicast address. It then waits for a response in the form of a VOLUNTEER message from another node on the network, the MOP server. If a response is received, then the destination address is changed from the multicast address to the node address of the server and the same REQ_PROGRAM message is retransmitted to the server as an Acknowledge.

Next, VMB begins sending REQ_MEM_LOAD messages to the server. The server responds with either:

- MEM_LOAD message, while there is still more to load.
- MEM_LOAD_w_XFER, if it is the end of the image.
- PARAM_LOAD_w_XFER, if it is the end of the image and operating system parameters are required.

The "load number" field in the load messages is used to synchronize the load sequence. At the beginning of the exchange, both the requester and server initialize the load number. The requester only increments the load number if a load packet has been successfully received and loaded. This forms the Acknowledge to each exchange. The server will resend a packet with a specific load number, until it sees the load number incremented. The final Acknowledge is sent by the requester and has a load number equivalent to the load number of the appropriate LOAD_w_XFER message + 1.

Because the request for load assistance is a MOP "must transact" operation, the network bootstrap continues indefinitely until a volunteer is found. The REQ_PROGRAM message is sent out in bursts of eight at four second intervals, the first four in MOP Version four IEEE 802.3 format and the last four in MOP Version 3 Ethernet format. The backoff period between bursts doubles each cycle from an initial value of four seconds, to eight seconds,... up to a maximum of five minutes. However, to reduce the likelihood of many nodes posting requests in lock-step, a random "jitter" is applied to the backoff period. The actual backoff time is computed as (.75+(.5*RND(x)))*BACKOFF, where 0<=x<1.

4.7.3.3 Network "Listening"

While the CPU module is waiting for a load volunteer during bootstrap, it "listens" on the network for other maintenance messages directed to the node and periodically identifies itself at the end of each 8- to 12-minute interval before a bootstrap retry. In particular, this "listener" supplements the Maintenance Operation Protocol (MOP) functions of the VMB load requester typically found in bootstrap firmware and supports.

- A remote console server that generates COUNTERS messages in response to REQ_COUNTERS messages, unsolicited SYSTEM_ID messages every 8 to 12 minutes, and solicited SYSTEM_ID messages in response to REQUEST_ID messages, as well as recognition of BOOT messages.
- A loopback server that responds to Ethernet loopback messages by echoing the message to the requester.
- An IEEE 802.2 responder that replies to both XID and TEST messages.

During network bootstrap operation, the KA50/51/55/56 complies with the requirements defined in the "NI Node Architecture Specification" for a primitive node. The firmware listens only to MOP "Load/Dump", MOP "Remote Console", Ethernet "Loopback Assistance", and IEEE 802.3 XID/TEST messages (listed in Table 4–5) directed to the Ethernet physical address of the node. All other Ethernet protocols are filtered by the network device driver.

The MOP functions and message types, which are supported by the KA50/51 /55/56, are summarized in Tables 4–3 and 4–5.

Table 4–5 MOP Multicast Addresses and Protocol Specifiers

Function	Address	IEEE Prefix[1]	Protocol	Owner
Dump/Load	AB-00-00-01-00-00	08-00-2B	60-01	Digital
Remote console	AB-00-00-02-00-00	08-00-2B	60-02	Digital
Loopback assistance	CF-00-00-00-00-00[2]	08-00-2B	90-00	Digital

[1]MOP V4.0 only.
[2]Not used.

4.8 Operating System Restart

An operating system restart is the process of bringing up the operating system from a known initialization state following a processor halt. This procedure is often called *restart* or *warmstart*, and should not be confused with a processor restart which results in firmware entry.

On the KA50/51/55/56, a restart occurs if the conditions specified in Table G–1 are satisfied.

To restart a halted operating system, the firmware searches system memory for the Restart Parameter Block (RPB), a data structure constructed for this purpose by VMB. (Refer to Table C–2 in Appendix C for a detailed description of this data structure.) If a valid RPB is found, the firmware passes control to the operating system at an address specified in the RPB.

The firmware keeps a "restart in progress" (RIP) flag in CPMBX which it uses to avoid repeated attempts to restart a failing operating system. An additional "restart in progress" flag is maintained by the operating system in the RPB.

The firmware uses the following algorithm to restart the operating system:

1. Check CPMBX<3>(RIP). If it is set, restart fails.
2. Print the message "Restarting system software." on the console terminal.
3. Set CPMBX<3>(RIP).
4. Search for a valid RPB. If none is found, restart fails.
5. Check the operating system RPB$L_RSTRTFLG<0>(RIP) flag. If it is set, restart fails.
6. Write "0" on the diagnostic LEDs.

7. Dispatch to the restart address, RPB$L_RESTART, with:

SP	Physical address of the RPB plus 512
AP	Halt code
PSL	041F0000
PR$_MAPEN	0

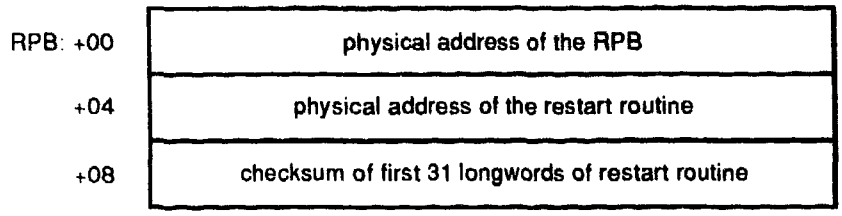
If the restart is successful, the operating system must clear CPMBX<3>(RIP).

If restart fails, the firmware prints "Restart failure." on the system console.

4.8.1 Locating the RPB

The RPB is a page-aligned control block which can be identified by the first three longwords. The format of the RPB "signature" is shown in Figure 4–6. (Refer to Table C–2 in Appendix C for a complete description of the RPB.)

Figure 4–6 Locating the Restart Parameter Block



The firmware uses the following algorithm to find a valid RPB:

1. Search for a page of memory that contains its address in the first longword. If none is found, the search for a valid RPB has failed.
2. Read the second longword in the page (the physical address of the restart routine). If it is not a valid physical address, or if it is zero, return to step 1. The check for zero is necessary to ensure that a page of zeros does not pass the test for a valid RPB.
3. Calculate the 32 bit twos-complement sum (ignoring overflows) of the first 31 longwords of the restart routine. If the sum does not match the third longword of the RPB, return to step 1.
4. A valid RPB has been found.

5

System Troubleshooting and Diagnostics

This chapter provides troubleshooting information for the two primary diagnostic methods: online, interpreting error logs to isolate the FRU; and offline, interpreting ROM-based diagnostic messages to isolate the FRU.

In addition, the chapter provides information on using MOP Ethernet functions to isolate errors, and interpreting UETP failures.

The chapter concludes with a section on running loopback tests to test the console port and embedded Ethernet ports.

Note

The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95, and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, may appear on the console and/or printouts from time to time.

5.1 Basic Troubleshooting Flow

Before troubleshooting any system problem, check the site maintenance log for the system's service history. Be sure to ask the system manager the following questions:

- Has the system been used before and did it work correctly?
- Have changes (changes to hardware, updates to firmware or software) been made to the system recently?
- What is the state of the system—is it on line or off line?

  If the system is off line and you are not able to bring it up, use the offline diagnostic tools, such as RBDs, MDM, and LEDs.

If the system is on line, use the online diagnostic tools, such as error logs, crash dumps, UETP, and other log files.

Four common problems occur when you make a change to the system:

- Incorrect cabling
- Module configuration errors (incorrect CSR addresses and interrupt vectors)
- Incorrect grant continuity
- Incorrect bus node ID plugs

In addition, check the following:

- If you have received error notification using VAXsimPLUS, check the mail messages and error logs as described in Section 5.2.
- If the operating system fails to boot (or appears to fail), check the console terminal screen for an error message. If the terminal displays an error message, see Section 5.3.
- Check the LEDs on the device you suspect is bad. If no errors are indicated by the device LEDs, run the ROM-based diagnostics described in this chapter.
- If the system boots successfully, but a device seems to fail or an intermittent failure occurs, check the error log ([SYSERR]ERRLOG.SYS) as described in Section 5.2.
- For fatal errors, check that the crash dump file exists for further analysis ([SYSEXE]SYSDUMP.DMP).
- Check other log files, such as OPERATOR.LOG, OPCOM.LOG, SETHOST.LOG, etc. Many of these can be found in the [SYSMGR] account. SETHOST.LOG is useful in comparing the console output with event logs and crash dumps in order to see what the system was doing at the time of the error.

  Use the following command to create SETHOST.LOG files, then log into the system account.

  ```
  $ SET HOST/LOG 0
  ```

  After logging out this file will reside in the [SYSMGR] account.

  If the system is failing in the boot or start-up phase, it may be useful to include the command SET VERIFY in the front of various start-up .COM files to obtain a trace of the start-up commands and procedures.

When troubleshooting, note the status of cables and connectors before you perform each step. Label cables before you disconnect them. This step saves you time and prevents you from introducing new problems.

Most communications modules use floating CSR addresses and interrupt vectors. If you remove a module from the system, you may have to change the addresses and vectors of other modules.

If you change the system configuration, run the CONFIGURE utility at the console I/O prompt (>>>) to determine the CSR addresses and interrupt vectors recommended by Digital.

5.2 Product Fault Management and Symptom-Directed Diagnosis

This section describes how errors are handled by the microcode and software, how the errors are logged, and how, through the Symptom-Directed Diagnosis (SDD) tool, VAXsimPLUS, errors are brought to the attention of the user. This section also provides the service theory used to interpret error logs to isolate the FRU. Interpreting error logs to isolate the FRU is the primary method of diagnosis.

5.2.1 General Exception and Interrupt Handling

This section describes the first step of error notification: the errors are first handled by the microcode and then are dispatched to the OpenVMS error handler.

The kernel uses the NVAX core chipset: NVAX CPU, NVAX Memory Controller (NMC), and NDAL to CDAL adapter (NCA).

Internal errors within the NVAX CPU result in machine check exceptions, through System Control Block (SCB) vector 004, or soft error interrupts at Interrupt Priority Level (IPL) 1A, SCB vector 054 hex.

External errors to the NVAX CPU, which are detected by the NMC or NDAL to CDAL adapter (NCA), usually result in these chips posting an error condition to the NVAX CPU. The NVAX CPU will then generate a machine check exception through SCB vector 004, hard error interrupt, IPL 1D, through SCB vector 060 (hex), or a soft error interrupt through SCB vector 054.

External errors to the NMC and NCA, which are detected by chips on the CDAL busses for transactions which originated by the NVAX CPU, are typically signaled back to the NCA adapter. The NCA adapter will post an error signal back to the NVAX CPU which generates a machine check or high level interrupt.

In the case of Direct Memory Access (DMA) transactions where the NCA or NMC detects the error, the errors are typically signaled back to the CDAL-Bus device, but not posted to the NVAX CPU. In these cases the CDAL-Bus device typically posts a device level interrupt to the NVAX CPU via the NCA. In almost all cases, error state is latched by the NMC and NCA. Although these errors will not result in a machine check exception or high level interrupt (i.e. results in device level IPL 14–17 versus error level IPL 1A, 1D), the OpenVMS machine check handler has a polling routine that will search for this state at one-second intervals. This will result in the host logging a polled error entry.

These conditions cover all of the cases that will eventually be handled by the OpenVMS error handler. The OpenVMS error handler will generate entries that correspond to the machine check exception, hard or soft error interrupt type, or polled error.

5.2.2 OpenVMS Error Handling

Upon detection of a machine check exception, hard error interrupt, soft error interrupt or polled error, the OpenVMS operating ststem will perform the following actions:

- Snapshot the state of the kernel.
- In most entry points, disable the caches.
- If it is a machine check and if the machine check is recoverable, determine if instruction retry is possible.

  Instruction retry is possible if one of the following conditions is true:

  - If PCSTS <10>PTE_ER = 0:

    Check that (ISTATE2 <07>VR = 1) or (PSL <27> FPD = 1)

    Otherwise crash the system or process depending on PSL <25:24> Current Mode.

  - If PCSTS <10>PTE_ER = 1:

    Check that (ISTATE2 <07>VR = 1) and (PSL <27>FPD = 0) and (PCSTS <09>PTE_ER_WR = 0)

    Otherwise crash the system.

  ISTATE2 is a longword in the machine check stack frame at offset (SP)+24; PSL is a longword in the machine check stack frame at offset (SP)+32; VR is the VAX Restart flag; and FPD is the First Part Done flag.
- Check to see if the threshold has been exceeded for various errors (typically the threshold is exceeded if 3 errors occur within a 10 minute interval).

- If the threshold has been exceeded for a particular type of cache error, mark a flag that will signify that this resource is to be disabled (the cache will be disabled in most, but not all, cases).
- Update the SYSTAT software register with results of error/fault handling.
- For memory uncorrectable Error Correction Code (ECC) errors:
  - If machine check, mark page bad and attempt to replace page.
  - Fill in MEMCON software register with memory configuration and error status for use in FRU isolation.
- For memory single-bit correctable ECC errors:
  - Fill in Corrected Read Data (CRD) entry FOOTPRINT with set, bank, and syndrome information for use in FRU isolation.
  - Update the CRD entry for time, address range, and count; fill the MEMCON software register with memory configuration information.
  - Scrub memory location for first occurrence of error within a particular footprint. If second or more occurrence within a footprint, mark page bad in hopes that page will be replaced later. Disable soft error logging for 10 minutes if threshold is exceeded.
  - Signify that CRD buffer be logged for the following events: system shutdown (operator shutdown or crash), hard single-cell address within footprint, multiple addresses within footprint, memory uncorrectable ECC error, or CRD buffer full.
- For ownership memory correctable ECC error, scrub location.
- Log error.
- Crash process or system, dependent upon PSL (Current Mode) with a fatal bugcheck for the following situations:
  - Retry is not possible.
  - Memory page could not be replaced for uncorrectable ECC memory error.
  - Uncorrectable tag store ECC errors present in writeback cache.
  - Uncorrectable data store ECC errors present in writeback cache for locations marked as OWNED.
  - Most INT60 errors.
  - Threshold is exceeded (except for cache errors).

  - A few other errors of the sort considered nonrecoverable are present.
- Disable cache(s) permanently if error threshold is exceeded.
- Flush and re-enable those caches which have been marked as good.
- Clear the error flags.
- Perform Return from Exception or Interrupt (REI) to recover and restart or continue the instruction stream for the following situations:
  - Most INT54 errors.
  - Those INT60 and INT54 errors which result in bad ECC written to a memory location. (These errors can provide clues that the problem is not memory related.)
  - Machine check conditions where instruction retry is possible.
  - Memory uncorrectable ECC error where page replacement is possible and instruction retry is possible.
  - Threshold exceeded (for cache errors only).
  - Return from Subroutine (RSB) and return from all polled errors.

Note

The results of the OpenVMS error handler may be preserved within the operating system session (for example, disabling a cache) but not across reboots.

Although the system can recover with cache disabled, the system performance will be degraded, since access time increases as available cache decreases.

5.2.3 OpenVMS Error Logging and Event Log Entry Format

The OpenVMS error handler for the kernel can generate six different entry types, as shown in Table 5–1. All error entry types, with the exception of correctable ECC memory errors, are logged immediately.

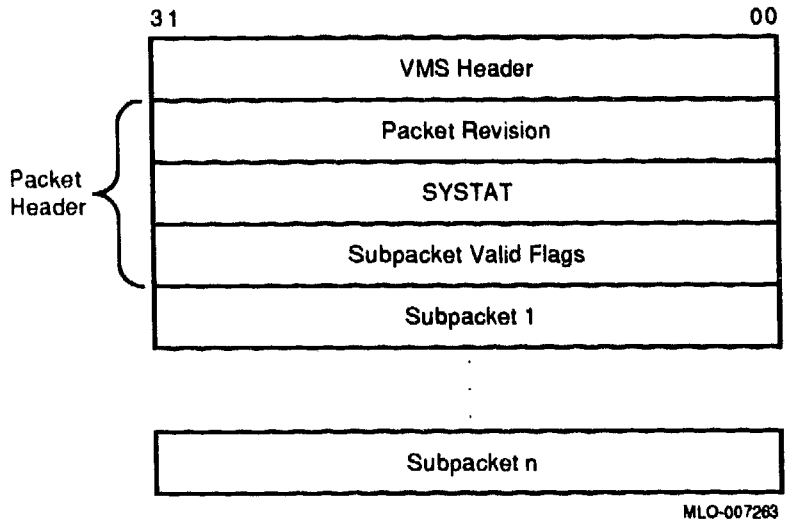
Table 5–1 OpenVMS Error Handler Entry Types

OpenVMS Entry Type	Code	Description
EMB$C_MC	(002.)	Machine Check Exception SCB Vector 4, IPL 1F
EMB$C_SE	(006.)	Soft Error Interrupt Correctable ECC Memory Error SCB Vector 54, IPL 1A
EMB$C_INT54	(026.)	Soft Error Interrupt SCB Vector 54, IPL 1A
EMB$C_INT60	(027.)	Hard Error Interrupt 60 SCB Vector 60, IPL 1D
EMB$C_POLLED	(044.)	Polled Errors No exception or interrupt generated by hardware.
EMB$C_BUGCHECK		Fatal bugcheck Bugcheck Types: MACHINECHK ASYNCWRTER BADMCKCOD INCONSTATE UNXINTEXC

Each entry consists of an OpenVMS operating system header, a packet header, and one or more subpackets (Figure 5–1). Entries can be of variable length based on the number of subpackets within the entry. The FLAGS software register in the packet header shows which subpackets are included within a given entry.

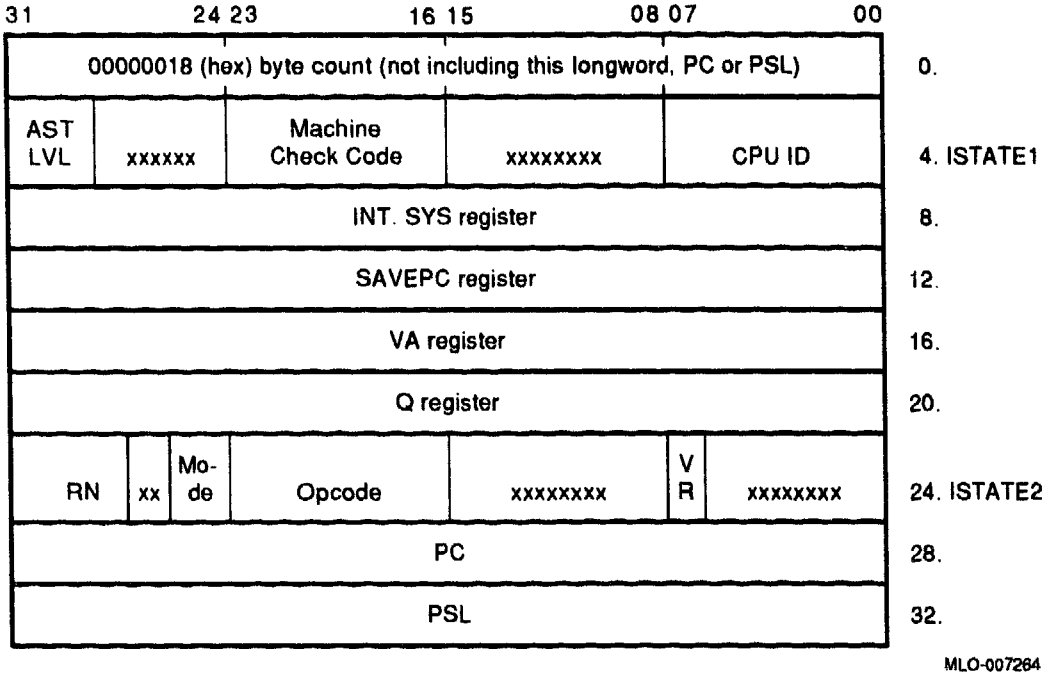
Refer to Section 5.2.4 for actual examples of the error and event logs described throughout this section.

Figure 5–1 Event Log Entry Format



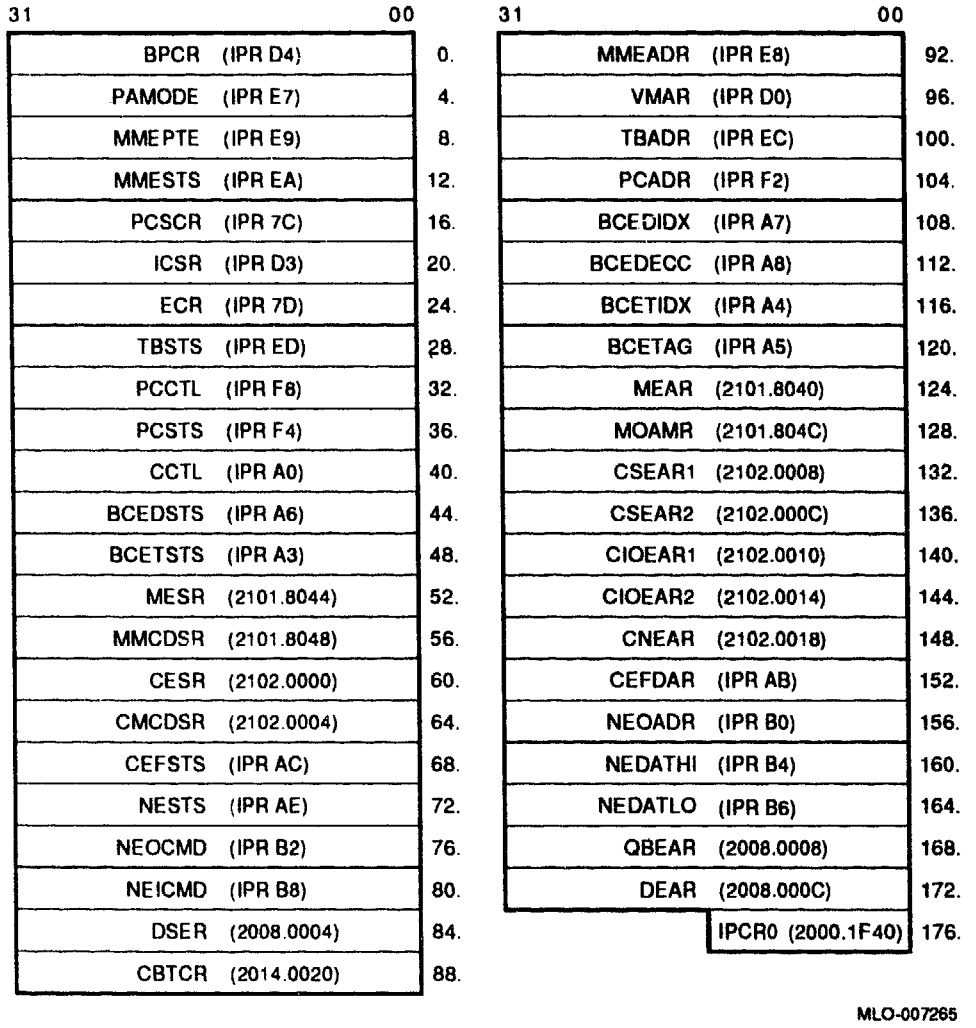
Machine check exception entries contain, at a minimum, a Machine Check Stack Frame subpacket (Figure 5–2).

Figure 5–2 Machine Check Stack Frame Subpacket



INT54, INT60, Polled, and some Machine Check entries contain a processor Register subpacket (Figure 5–3), which consists of some 40 plus hardware registers.

Figure 5–3 Processor Register Subpacket



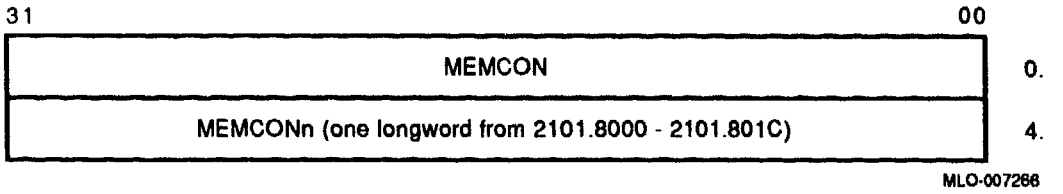
Note

The byte count, although part of the stack frame, is not included in the error log entry itself.

Bugcheck entries generated by the OpenVMS kernel error handler include the first 23 registers from the processor Register subpacket along with the Time of Day Register (TODR) and other software context states.

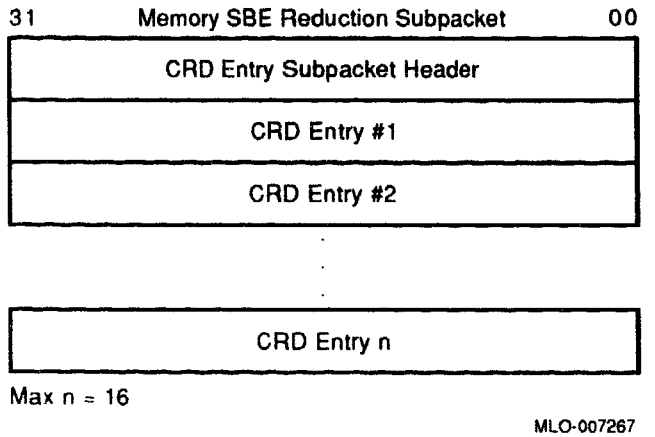
Uncorrectable ECC memory error entries include a Memory subpacket (Figure 5–4). The memory subpacket consists of MEMCON, which is a software register containing the memory configuration and error status used for FRU isolation, and MEMCONn, the hardware register that matched the error address in MEAR.

Figure 5–4 Memory Subpacket for ECC Memory Errors



Correctable Memory Error entries have a Memory (Single-Bit Error) SBE Reduction subpacket (Figure 5–5). This subpacket, unlike all others, is of variable length. It consists solely of software registers from state maintained by the error handler, as well as hardware state transformed into a more usable format.

Figure 5–5 Memory SBE Reduction Subpacket (Correctable Memory Errors)

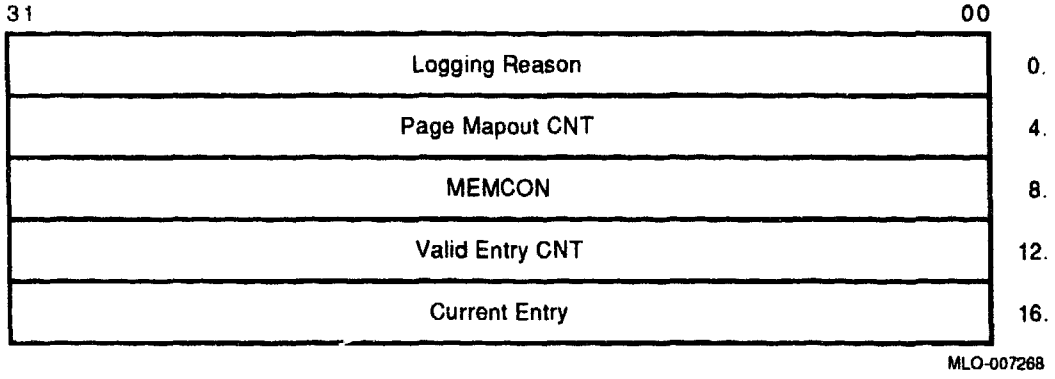


The OpenVMS error handler maintains a Correctable Read Data (CRD) buffer internally within memory that is flushed asynchronously for high-level events to the error log file. The CRD buffer and resultant error log entry are maintained and organized as follows.

- Each entry has a subpacket header (Figure 5–6) consisting of LOGGING REASON, PAGE MAPOUT CNT, MEMCON, VALID ENTRY CNT, and

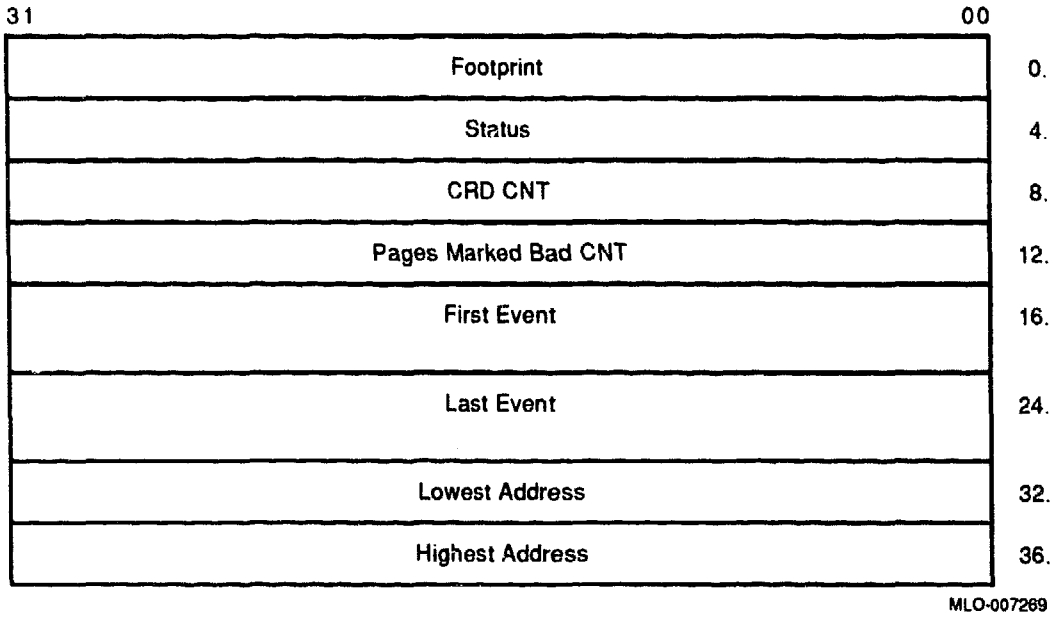
CURRENT ENTRY. MEMCON contains memory configuration information, but no error status as is done for the Memory subpacket.

Figure 5–6 CRD Entry Subpacket Header



- Following the subpacket header are 1 to 16 fixed-length Memory CRD Entries (Figure 5–7). The number of Memory CRD entries is shown in VALID ENTRY CNT. The entry which caused the report to be generated is in CURRENT ENTRY.

Figure 5–7 Correctable Read Data (CRD) Entry



Each Memory CRD Entry represents one unique DRAM within the memory subsystem. A unique set, bank, and syndrome are stored in footprint to construct a unique ID for the DRAM.

Rather than logging an error for each occurrence of a single symbol correctable ECC memory error, the OpenVMS error handler maintains the CRD buffer—it creates a Memory CRD Entry for new footprints and updates an existing Memory CRD Entry for errors that occur within the range specified by the ID in FOOTPRINT. This reduces the amount of data logged overall without losing important information—errors are logged per unique failure mode rather than on a per error basis.

Each Memory CRD entry consists of a FOOTPRINT, STATUS, CRD CNT, PAGE MAPOUT CNT, FIRST EVENT, LAST EVENT, LOWEST ADDRESS and HIGHEST ADDRESS.

FIRST EVENT, LAST EVENT, LOWEST ADDRESS and HIGHEST ADDRESS are updated to show the range of time and addresses of errors which have occurred for a DRAM. CRD CNT is simply the total count per footprint. PAGE MAPOUT CNT is the number of pages that have been marked bad for a particular DRAM.

STATUS contains a record of the failure mode status of a particular DRAM over time. This in turn determines whether or not the CRD buffer is logged. For the first occurrence of an error within a particular DRAM, the memory location will be scrubbed (corrected read data is read, then written back to the memory location) and CRD CNT will be set to 1. Since most memory single-bit errors are transient due to alpha particles, logging of the CRD buffer will not be done immediately for the first occurrence of an error within a DRAM. The CRD buffer will, however, be logged at the time of system shutdown (operator or crash induced), or when a more severe memory subsystem error occurs.

If the FOOTPRINT/DRAM experiences another error (CRD CNT > 1), the OpenVMS operating system will set HARD SINGLE ADDRESS or MULTIPLE ADDRESSES along with SCRUBBED in STATUS. Scrubbing is no longer performed; instead, pages are marked bad. In this case, the OpenVMS operating system will log the CRD buffer immediately. The CRD Buffer will also be logged immediately if PAGE MAPOUT THRESHOLD EXCEEDED is set in SYSTAT as a result of pages being marked bad. The threshold is reached if more than one page per Mbyte of system memory is marked bad.

Note

CURRENT ENTRY will be zero in the Memory SBE Reduction subpacket header if the CRD buffer was logged, not as a result of a HARD SINGLE ADDRESS or MULTIPLE ADDRESSES error in STATUS, but as a result of a memory uncorrectable ECC error shown as RELATED ERROR, or as a result of CRD BUFFER FULL or SYSTEM SHUTDOWN, all of which are shown under LOGGING REASON.

5.2.4 OpenVMS Event Record Translation

The kernel error log entries are translated from binary to ASCII using the ANALYZE/ERROR command. To invoke the error log utility, enter the DCL command ANALYZE/ERROR_LOG.

Format:

ANALYZE_ERROR_LOG [/qualifier(s)] [file-spec] [,...]

Example:

$ ANALYZE/ERROR_LOG/INCLUDE=(CPU,MEMORY)/SINCE=TODAY

The error log utility translates the entry into the traditional three-column format. The first column shows the register mnemonics, the second column depicts the data in hex, and the last column shows the actual English translations.

As in the above example, the OpenVMS error handler also provides support for the /INCLUDE qualifier, such that CPU and MEMORY error entries can be selectively translated.

Since most kernel errors are bounded to either the processor module/system board or memory modules, the individual error flags and fields are not covered by the service theory. Although these flags are generally not required to diagnose a system to the FRU (Field Replaceable Unit), this information can be useful for component isolation.

ERF bit to text translation highlights all error flags that are set, and other significant state—these are displayed in capital letters in the third column. Otherwise, nothing is shown in the translation column. The translation rules also have qualifiers such that if the setting of an error flag causes other registers to be latched, the other registers will be translated as well. For example, if a memory ECC error occurs, the syndrome and error address fields will be latched as well. If such a field is valid, the translation will be shown (e.g. MEMORY ERROR ADDRESS); otherwise, no translation is provided.

5.2.5 Interpreting CPU Faults Using ANALYZE/ERROR

If the following three conditions are satisfied, the most likely FRU is the CPU module. Example 5–1 shows an abbreviated error log with numbers to highlight the key registers.

❶ No memory subpacket is listed in the third column of the FLAGS register.

❷ CESR register bit <09>, CP2 IO Error, is equal to zero in the KA50/51/55 /56 Register Subpacket.

❸ DSER register bits <07>, Q22 Bus NXM, <05>, Q22 Bus Device Parity Error, or <02>, Q-22 Bus No Grant, are equal to zero in the KA50/51/55/56 Register Subpacket.

The FLAGS register is located in the packet header, which immediately follows the system identification header; the CESR and DSER registers are listed under the KA50/51/55/56 Register Subpacket.

CPU errors will increment an OpenVMS global counter, which can be viewed using the DCL command SHOW ERROR, as shown in Example 5–2.

To determine if any resources have been disabled, for example, if cache has been disabled for the duration of the OpenVMS session, examine the flags for the SYSTAT register in the packet header.

In Example 5–1, a translation buffer data parity error latched in the TBSTS register caused a machine check exception error.

Example 5–1 Error Log Entry Indicating CPU Error

```
V A X / V M S        SYSTEM ERROR REPORT        COMPILED 14-JAN-1992 18:55:52
                                                                 PAGE   1.
******************************** ENTRY      1. ********************************
ERROR SEQUENCE 11.                              LOGGED ON:        SID 13001401
DATE/TIME 27-SEP-1991 14:40:10.85                                 SYS_TYPE 03110A01
SYSTEM UPTIME: 0 DAYS 00:12:12
SCS NODE: OMEGA1                                                  VAX/OpenVMS V5.5-2

MACHINE CHECK KA50  CPU Microcode Rev # 1.  CONSOLE FW REV# 1.1
                    Standard Microcode Patch    Patch Rev # 10.

      REVISION        00000000
      SYSTAT          00000001
                                    ATTEMPTING RECOVERY
      FLAGS           00000003
                                    machine check stack frame
                                    KA50 subpacket

 STACK FRAME SUBPACKET

      ISTATE_1        80050000
                                    MACHINE CHECK FAULT CODE = 05(x)
                                    Current AST level  = 4(X)
                                    ASYNCHRONOUS HARDWARE ERROR
      .
      .
      .
      PSL             04140001
                                    c-bit
                                    executing on interrupt stack
                                    PSL previous mode = kernel
                                    PSL current mode = kernel
                                    first part done set

 KA50 REGISTER SUBPACKET
```

(continued on next page)

Example 5–1 (Cont.) Error Log Entry Indicating CPU Error

```
BPCR            ECC80024
.
.
.
TBSTS           800001D3
                                LOCK SET
                                TRANSLATION BUFFER DATA PARITY ERROR
                                em_latch invalid
                                s5_command  = 1D(X)
                                valid Ibox specifier ref. error stored
.
.
.
CESR            00000000 ●
.
.
.
DSER            00000000 ●
.
.
.
IPCR0           00000020
                                LOCAL MEMORY EXTERNAL ACCESS ENABLED
```

Note

Ownership (O-bit) memory correctable or fatal errors (MESR <04> or MESR <03> of the processor Register Subpacket set equal to 1) are processor module errors, NOT memory errors.

Example 5–2 SHOW ERROR Display Using the OpenVMS Operating System

```
$ SHOW ERROR
Device                          Error Count
CPU                                 1
MEMORY                              1
PAB0:                               1
PAA0:                               1
PTA0:                               1
RTA2:                               1
$
```

5.2.6 Interpreting Memory Faults Using ANALYZE/ERROR

If "memory subpacket" or "memory sbe reduction subpacket" is listed in the third column of the FLAGS register, there is a problem with one or more of the memory modules, CPU module, or backplane.

- The "memory subpacket" message indicates an uncorrectable ECC error. Refer to Section 5.2.6.1 for instructions in isolating uncorrectable ECC error problems.
- The "memory sbe reduction subpacket" message indicates correctable ECC errors. Refer to Section 5.2.6.2 for instructions in isolating correctable ECC error problems.

Note

The memory fault interpretation procedures work only if the memory modules have been properly installed and configured. For example, memory modules should start in backplane slot 4 (next to the processor module in slot 5) and proceed to slot 1 with no gaps.

Note

Although the OpenVMS error handler has built-in features to aid Services in memory repair, good judgment is needed by the Service Engineer. It is essential to understand that in many, if not most cases, correctable ECC errors are transient in nature. No amount of repair will fix them, as generally there is nothing to be fixed.

Memory modules can represent a great expense to the Corporation when they are sent back to Repair with no errors. If one disagrees with the strategy in this section or has questions or suggestions, please contact Corporate Support.

5.2.6.1 Uncorrectable ECC Errors

Refer to Example 5–3, which provides an abbreviated error log for uncorrectable ECC errors.

For uncorrectable ECC errors, a memory subpacket will be logged as indicated by "memory subpacket" listed in the third column of the FLAGS software register (❶). Also, the hardware register MESR <11> (❷) of the processor Register Subpacket will be set equal to 1, and MEAR will latch the error address (❺).

Examine the MEMCON software register (❸) under the memory subpacket. The MEMCON register provides memory configuration information.

The OpenVMS error handler will mark each page bad and attempt page replacement, indicated in SYSTAT (❶). The DCL command SHOW MEMORY (Example 5–4) will also indicate the result of OpenVMS page replacement.

Uncorrectable memory errors will increment the OpenVMS global counter, which can be viewed using the DCL command SHOW ERROR.

Note

If register MESR <11> was set equal to 1, but MESR <19:12> syndrome equals 07, no memory subpacket will be logged as a result of incorrect check bits written to memory because of an NDAL bus parity error detected by the NMC. In short, this indicates a problem with the CPU module, not memory. There should be a previous entry with MESR <22>, NDAL Data Parity Error set equal to 1.

Note

One type of uncorrectable ECC error, that due to a "disown write", will result in a CRD entry like those for correctable ECC errors. The FOOTPRINT longword for this entry contains the message "Uncorrectable ECC errors due to disown write". The failing module should be replaced for this error.

Example 5–3 Error Log Entry Indicating Uncorrectable ECC Error

```
V A X / V M S        SYSTEM ERROR REPORT        COMPILED  6-NOV-1991 10:16:49
                                                                  PAGE  25.
******************************** ENTRY      13. ********************************
ERROR SEQUENCE 2.                                LOGGED ON:        SID 13001401
DATE/TIME  4-OCT-1991 09:14:29.86                                  SYS_TYPE 03110A01
SYSTEM UPTIME: 0 DAYS 00:01:39
SCS NODE: OMEGA1                                                   VAX/OpenVMS V5.5-2

INT54 ERROR  KA50  CPU Microcode Rev # 1.  CONSOLE FW REV# 1.1
                   Standard Microcode Patch    Patch Rev # 10.

      REVISION        00000000
      SYSTAT          00000601
                                    ATTEMPTING RECOVERY
                                    PAGE MARKED BAD
                                    PAGE REPLACED      ❹
      FLAGS           00000006
                                    memory subpacket   ❶
                                    KA50 subpacket

KA50 REGISTER SUBPACKET

      BPCR            ECC80000
      .
      .
      .
      MESR            80006800
                                    UNCORRECTABLE MEMORY ECC ERROR   ❺
                                    ERROR SUMMARY
                                    MEMORY ERROR SYNDROME  = 06(X)
      .
      .
      .
      MEAR            02FFDC00
                                    main memory error address  = 0BFF7000   ❻
                                    ndal commander id  = 00(X)
      .
      .
      .
      IPCR0           00000020
                                    LOCAL MEMORY EXTERNAL ACCESS ENABLED

 MEMORY SUBPACKET

      MEMCON          000FFFF02  ❸
                                    MEMORY CONFIGURATION:
                                    MS44-AA SIM Memory Module 4 MB location 1E
                                    MS44-AA SIM Memory Module 4 MB location 1F
                                    MS44-AA SIM Memory Module 4 MB location 1G
                                    MS44-AA SIM Memory Module 4 MB location 1H
                                     total memory = 16MB
```

(continued on next page)

Example 5–3 (Cont.) Error Log Entry Indicating Uncorrectable ECC Error

```
MEMCON3          8B000003
                                  64 bit mode
                                  Base address valid
                                  RAM size = 1MB
                                  base address  = 0B(X)
```

Example 5–4 SHOW MEMORY Display Under the OpenVMS Operating System

```
$ SHOW MEMORY
            System Memory Resources on 21-FEB-1992 05:58:52.58

Physical Memory Usage (pages):     Total       Free      In Use    Modified
  Main Memory (128.00Mb)          262144     224527       28759        8858

  Bad Pages                        Total    Dynamic  I/O Errors      Static
                                       1          1           0           0

Slot Usage (slots):                Total       Free    Resident     Swapped
  Process Entry Slots                360        347          13           0
  Balance Set Slots                  324        313          11           0

Fixed-Size Pool Areas (packets):   Total       Free      In Use        Size
  Small Packet (SRP) List           3067       2724         343         128
  I/O Request Packet (IRP) List     2263       2070         193         176
  Large Packet (LRP) List             87         61          26        1856

Dynamic Memory Usage (bytes):      Total       Free      In Use     Largest
  Nonpaged Dynamic Memory        1037824     503920      533904      473184
  Paged Dynamic Memory           1468416     561584      906832      560624

Paging File Usage (pages):                     Free  Reservable       Total
  DISK$VMS054-0:[SYS0.SYSEXE]PAGEFILE.SYS    300000      266070      300000

Of the physical pages in use, 24120 pages are permanently allocated to OpenVMS.
$
```

Using the OpenVMS command ANALYZE/SYSTEM, you can associate a page that had been replaced (Bad Pages in SHOW MEMORY display) with the physical address in memory.

In Example 5–5, 5ffb8 (under the Page Frame Number (PFN) column) is identified as the single page that has been replaced. The command EVAL 5ffb8 * 200 converts the PFN to a physical page address. The result is 0bff7000, which is the MEAR address translated in Example 5–3. (Bits <8:0> of the addresses may differ since the page address from EVAL always shows bits <8:0> as 0.)

Example 5–5 Using ANALYZE/SYSTEM to Check the Physical Address in Memory for a Replaced Page

```
$ ANALYZE/SYSTEM

VAX/OpenVMS System analyzer

SDA> SHOW PFN /BAD

Bad page list
-------------
Count:                  1
Lolimit:               -1
High limit:     1073741824

  PFN     PTE ADDRESS    BAK     REFCNT   FLINK    BLINK      TYPE       STATE
  ----    -----------  --------  ------   -----    -----    ----------  ----------
0005FFB8    00000000   00000000      0   00000000 00000000  20 PROCESS  02 BADLIST

SDA> EVAL 5ffb8 * 200
Hex = 0BFF7000   Decimal = 201289728
SDA> EXIT
$
```

5.2.6.2 Correctable ECC Errors

Refer to Example 5–6, which provides an error log showing correctable ECC errors.

For correctable ECC errors, a Single-Bit Error (SBE) Memory Subpacket will be logged as indicated by "memory sbe reduction subpacket" listed in the third column of the FLAGS software register (❶).

The Memory SBE Reduction Subpacket header contains a CURRENT ENTRY register (❷) that displays the number of the Memory CRD Entry that caused the error notification. If CURRENT ENTRY > 0, examine which bits are set in the STATUS register (❸) for this entry—GENERATE REPORT should be set.

Note

If CURRENT ENTRY = 0, then the entry was logged for something other than a single-bit memory correctable error Footprint. You will need to examine all of the Memory CRD Entries and Footprints to try to determine the likely FRU.

Check for the following:

- SCRUBBED (❹)—If SCRUBBED is the only bit set in the STATUS register, memory modules should NOT generally be replaced.

The kernel performs memory scrubbing of DRAM memory cells that may flip due to transient alpha particles. Scrubbing simply reads the corrected data and writes it back to the memory location. Returning memory modules that only have SCRUBBED set in STATUS will cost the corporation money, since the repair centers will generally not find a problem.

Unlike uncorrectable ECC errors, the error handling code cannot indicate if the page has been replaced. To get some idea, use DCL command, SHOW MEMORY. If the page mapout threshold has not been reached ("PAGE MAPOUT THRESHOLD EXCEEDED" is not set in SYSTAT packet header register (❺)), the system should be restarted at a convenient time to allow the power-up self-test and ROM-based diagnostics to map out these pages. This can be done by entering TEST 0 at the console prompt, running an extended script TEST A9, or by powering down then powering up the system. In all cases, the diagnostic code will mark the page bad for hard single address errors, as well as any uncorrectable ECC error by default.

If there are many locations affected by hard single-cell errors, on the order of one or more pages per MB of system memory, the memory module should be replaced. The console command SHOW MEMORY will indicate the number of bad pages per module. For example, if the system contains 64 MB of main memory and there are 64 or more bad pages, the affected memory should be replaced.

Note

Under the OpenVMS operating system, the page mapout threshold is calculated automatically. If "PAGE MAPOUT THRESHOLD EXCEEDED" is set in SYSTAT (❺), the failing memory module should be replaced.

In cases of a new memory module used for repair or as part of system installation, one may elect to replace the module rather than having diagnostics map them out, even if the threshold has not been reached for hard single-address errors.

- MULTIPLE ADDRESSES (❻)—If the second occurrence of an error within a footprint is at a different address (LOWEST ADDRESS not equal to HIGHEST ADDRESS (❼), MULTIPLE ADDRESSES will be set in STATUS along with SCRUBBED. Scrubbing will not be attempted for this situation. In most cases, the failing memory module should be replaced regardless of the page mapout threshold.

If CRD BUFFER FULL is set in LOGGING REASON (❽) (located in the subpacket header) or PAGE MAPOUT THRESHOLD EXCEEDED is set in SYSTAT (❼), the failing memory module should be replaced regardless of any thresholds.

For all cases (except when SCRUBBED is the only flag set in STATUS) isolate the offending memory by examining the translation in FOOTPRINT called MEMORY ERROR STATUS (❾): The memory module is identified by its backplane position. In Example 5–6, SIMM memory modules in locations 0A and 0B are identified as failing.

The Memory SBE Reduction Subpacket header translates the MEMCON register (❿) for memory subsystem configuration information.

Unlike uncorrectable memory and CPU errors, the OpenVMS global counter, as shown by the DCL command SHOW ERROR, is not incremented for correctable ECC errors unless it results in an error log entry for reasons other than system shutdown.

Note

If footprints are being generated for more than one memory module, especially if they all have the same bit in error, the processor module, backplane, or other component may be the cause.

Note

One type of uncorrectable ECC error, that due to a "disown write", will result in a CRD entry like those for correctable ECC errors. The FOOTPRINT longword for this entry contains the message "Uncorrectable ECC errors due to disown write". The failing module should be replaced for this error.

Example 5–6 Error Log Entry Indicating Correctable ECC Error

```
V A X / V M S        SYSTEM ERROR REPORT        COMPILED 21-NOV-1991 16:55:58
                                                                  PAGE   1.
******************************* ENTRY      1. ******************************
ERROR SEQUENCE 2.                              LOGGED ON:        SID 13001401
DATE/TIME 27-SEP-1991 09:51:13.98                          SYS_TYPE 03110A01
SYSTEM UPTIME: 0 DAYS 00:05:06
SCS NODE: OMEGA1                                          VAX/OpenVMS V5.5-2

CORRECTABLE MEMORY ERROR  KA50  CPU Microcode Rev # 1.  CONSOLE FW REV# 1.1
                    Standard Microcode Patch    Patch Rev # 10.

      REVISION        00000000
      SYSTAT          00000040 ❺
      FLAGS           00000008

                                  ❶
                                      memory sbe reduction subpacket

 MEMORY SBE REDUCTION SUBPACKET

      LOGGING REASON  00000004 ❻
                                      shutdown
      PAGE MAPOUT CNT 00000000
      MEMCON          000FFD01 ❼
                                      MEMORY CONFIGURATION:
                                      MS44-AA SIM Memory Module (4MB) Loc 0A
                                      MS44-AA SIM Memory Module (4MB) Loc 0B
                                      MS44-AA SIM Memory Module (4MB) Loc 0C
                                      MS44-AA SIM Memory Module (4MB) Loc 0D
                                       _Total memory = 16MB
                                       _sets enabled = 000000001
                                ❾  MEMORY ERROR STATUS:
                                       SIMM MEMORY MODULES: LOCATIONS 0A & 0B
                                       Set  = 0(X)
                                       Bank = A
      VALID ENTRY CNT 00000001
                                      1.
      CURRENT ENTRY   00000000
                                      0. ❷

 MEMORY CRD ENTRY 1.

      FOOTPRINT       00000073
```

(continued on next page)

Example 5–6 (Cont.) Error Log Entry Indicating Correctable ECC Error

```
                                MEMORY ERROR STATUS:
                                _SIMM MEMORY MODULE:  LOCATION 0A
                                _set = 0
                                _bank = 0.
                                ECC SYNDROME = 73(X)
                                _CORRECTED DATA BIT = 0.
                                _
STATUS ❹  00000010
                                scrubbed ❶
CRD CNT          00000001
                                1.
PAGE MAPOUT CNT  00000000
                                0.
FIRST EVENT      16B0F640
                 009622CB
                                16-OCT-1992 11:03:36.10
LAST EVENT       16B0F640
                 009622CB
                                16-OCT-1992 11:03:36.10
LOWEST ADDRESS   0BFF4000
HIGHEST ADDRESS  0BFF4000  ❻ ❼
```

Note

Ownership (O-bit) memory correctable or fatal errors (MESR <04> or MESR <03> of the processor Register Subpacket set equal to 1) are processor module errors, NOT memory errors.

5.2.7 Interpreting System Bus Faults Using ANALYZE/ERROR

If hardware register CESR <09> (❶) and/or CQBIC hardware register DSER <07>, <05>, or <02> (❷) is set equal to 1, there may be a problem with the Q–bus or Q–bus option.

When CESR <09> is set equal to 1, examine the hardware register CIOEAR2 (❸) to determine the address of the offending option.

Example 5–7 provides an error log showing a faulty Q–bus option. The CIOEAR2 error register indicates the first UQSSP controller as the offending address.

Example 5–7 Error Log Entry Indicating Q-Bus Error

```
V A X / V M S        SYSTEM ERROR REPORT         COMPILED 20-NOV-1991 14:28:13
                                                                  PAGE   1.
******************************** ENTRY      75. ********************************
ERROR SEQUENCE 1852.                              LOGGED ON:        SID 13001401
DATE/TIME 20-NOV-1991 14:26:11.14                                 SYS_TYPE 00310A01
SYSTEM UPTIME: 12 DAYS 20:04:19
SCS NODE:                                                         VAX/OpenVMS V5.5-2

MACHINE CHECK  KA50  CPU Microcode Rev # 1.  CONSOLE FW REV# 1.1
                     Standard Microcode Patch     Patch Rev # 10.

      REVISION        00000000
      SYSTAT          00000001
                                    ATTEMPTING RECOVERY
      FLAGS           00000003
                                    machine check stack frame
                                    KA50 subpacket

 STACK FRAME SUBPACKET

      ISTATE_1        80060000
      .
      .
      .
      PSL             03C00000
                                    PSL previous mode = user
                                    PSL current mode = user
                                    first part done set

KA50  REGISTER SUBPACKET
```

(continued on next page)

Example 5–7 (Cont.) Error Log Entry Indicating Q-Bus Error

```
      BPCR            ECC80024
      .
      .
      .
      CESR            80000200 ❶
                                    CP2 IO ERROR
                                    ERROR SUMMARY
      .
      .
      .
      DSER            00000080 ❷
                                    Q-22 BUS NXM
      .
      .
      .
      CIOEAR2         00001468                    ❸
                                    cp2 IO error address  = 20001468
                                    NDAL commander id (cp2 transac)  = 0(X)
      .
      .
      .
      IPCR0           00000020
                                    LOCAL MEMORY EXTERNAL ACCESS ENABLED
ANAL/ERR/OUT=QBUS QBUS.ZPD
```

5.2.8 Interpreting DMA ⇔ Host Transaction Faults Using ANALYZE/ERROR

Some kernel errors may result in two or more entries being logged. If the SGEC Ethernet controller or other CDAL device (residing on the processor module) encounter host main memory uncorrectable ECC errors, main memory NXMs or CDAL parity errors or timeouts, more than one entry results. Usually there will be one Polled Error entry logged by the host, and one or more Device Attention and other assorted entries logged by the device drivers.

In these cases the processor module or one of the four memory modules are the most likely cause of the errors. Therefore, it is essential to analyze Polled Error entries, since a polled entry usually represents the source of the error versus other entries, which are simply aftereffects of the original error.

Example 5–8 provides an abbreviated error log for a polled error. Example 5–9 provides an example of a device attention entry.

Example 5–8 Error Log Entry Indicating Polled Error

```
V A X / V M S        SYSTEM ERROR REPORT         COMPILED 17-FEB-1992 05:32:21
                                                          PAGE   1.
******************************** ENTRY      2. ********************************
ERROR SEQUENCE 15.                              LOGGED ON:        SID 13001401
DATE/TIME 17-FEB-1992 05:22:00.90                                 SYS_TYPE 00310A01
SYSTEM UPTIME: 0 DAYS 00:27:48
SCS NODE:                                                         VAX/OpenVMS V5.5-2

POLLED ERROR  KA50  CPU Microcode Rev # 1.  CONSOLE FW REV# 1.1
                    Standard Microcode Patch    Patch Rev # 10.

      REVISION        00000000
      SYSTAT          00000001
                                    ATTEMPTING RECOVERY
      FLAGS           00000006
                                    memory subpacket
                    KA50 subpacket

KA50 REGISTER SUBPACKET

      BPCR            ECC80024
      .
      .
      .
      MESR            8001B800
                                    UNCORRECTABLE MEMORY ECC ERROR
                                    ERROR SUMMARY
                                    MEMORY ERROR SYNDROME  = 1B(X)
      .
      .
      .
      MEAR            50000410
                                    main memory error address  = 00001040
                                    ndal commander id  = 05(X)
      .
      .
      .
      IPCR0           00000020
                                    LOCAL MEMORY EXTERNAL ACCESS ENABLED

MEMORY SUBPACKET

      MEMCON          000FFFF02
                                    MEMORY CONFIGURATION:
                                    MS44-AA SIM Memory Module 4 MB location 1E
                                    MS44-AA SIM Memory Module 4 MB location 1F
                                    MS44-AA SIM Memory Module 4 MB location 1G
                                    MS44-AA SIM Memory Module 4 MB location 1H
                                     total memory = 16MB
```

(continued on next page)

Example 5–8 (Cont.) Error Log Entry Indicating Polled Error

```
    MEMCONO          80000003
                                    64 bit mode
                                    Base address valid
                                    RAM size = 1MB
                                    base address  = 00(X)

ANAL/ERR/OUT=TB1 TB1.ZPD
```

Example 5–9 Device Attention Entry

```
V A X / V M S        SYSTEM ERROR REPORT        COMPILED 17-FEB-1992 05:32:21
                                                          PAGE   1.
********************************* ENTRY      2. *********************************
ERROR SEQUENCE 15.                              LOGGED ON:        SID 13001401
DATE/TIME 17-FEB-1992 05:22:00.90                                 SYS_TYPE 00310A01
SYSTEM UPTIME: 0 DAYS 00:27:48
SCS NODE:                                                        VAX/OpenVMS V5.5-2

DEVICE ATTENTION  KA50  CPU Microcode Rev # 1.  CONSOLE FW REV# 1.1
                  Standard Microcode Patch    Patch Rev # 10.

DSSI SUB-SYSTEM, PAB0: - PORT WILL BE RE-STARTED

      PORT TIMEOUT, DRIVER RESETTING PORT

      CNF             03060022
                                    MAINTENANCE ID = 0022(X)
                                    FIRMWARE REVISION = 06(X)
                                    HARDWARE REVISION = 03(X)
      PMCSR           00000000
      PSR             80010000
                                    MAINTENANCE ERROR
                                    SHARED HOST MEMORY ERROR
      PFAR            40001044
                                    APPROX HOST ADDR 40001044(X)
      PESR            00010000
                                    CPDAL BUS ERROR
      PPR             00000000
                                    NODE #0.
                                    0. BYTE INTERNAL BUFFER
                                    16. NODES MAXIMUM

      UCB$B_ERTCNT          2C
                                    44. RETRIES REMAINING
      UCB$B_ERTMAX          32
                                    50. RETRIES ALLOWABLE
      UCB$L_CHAR      0C450000
                                    SHARABLE
                                    AVAILABLE
                                    ERROR LOGGING
                                    CAPABLE OF INPUT
                                    CAPABLE OF OUTPUT
      UCB$W_STS           0010
                                    ONLINE
      UCB$W_ERRCNT        0007
                                    7. ERRORS THIS UNIT
ANAL/ERR/ENTRY=(ST:2,END:3)/OUT=POLL_SHM
```

5.2.9 VAXsimPLUS and System-Initiated Call Logging (SICL) Support

Symptom-Directed Diagnostic (SDD) toolkit support for KA50/51/55/56 kernels is provided in version 2.0 of the toolkit. If version 2.0 is not available, you should install the previous version, as it provides support for many existing options.

MicroVAX 3100 systems use Symptom-Directed Diagnosis tools primarily for notification. The VAX System Integrity Monitor Plus (VAXsimPLUS) interactive reporting tool triggers notification for high-level events recorded in SYSTAT and LOGGING REASON.

The VAXsimPLUS monitor simply parses for a handful of SYSTAT flags and LOGGING reason codes. The VAXsimPLUS monitor display is updated and triggering occurs if the threshold has been reached. Some flags have a threshold of one; for example, SYSTAT <08> ERROR THRESHOLD EXCEEDED will trigger VAXsimPLUS upon the first occurrence, since at least three errors would have already occurred and been handled by the OpenVMS operating system.

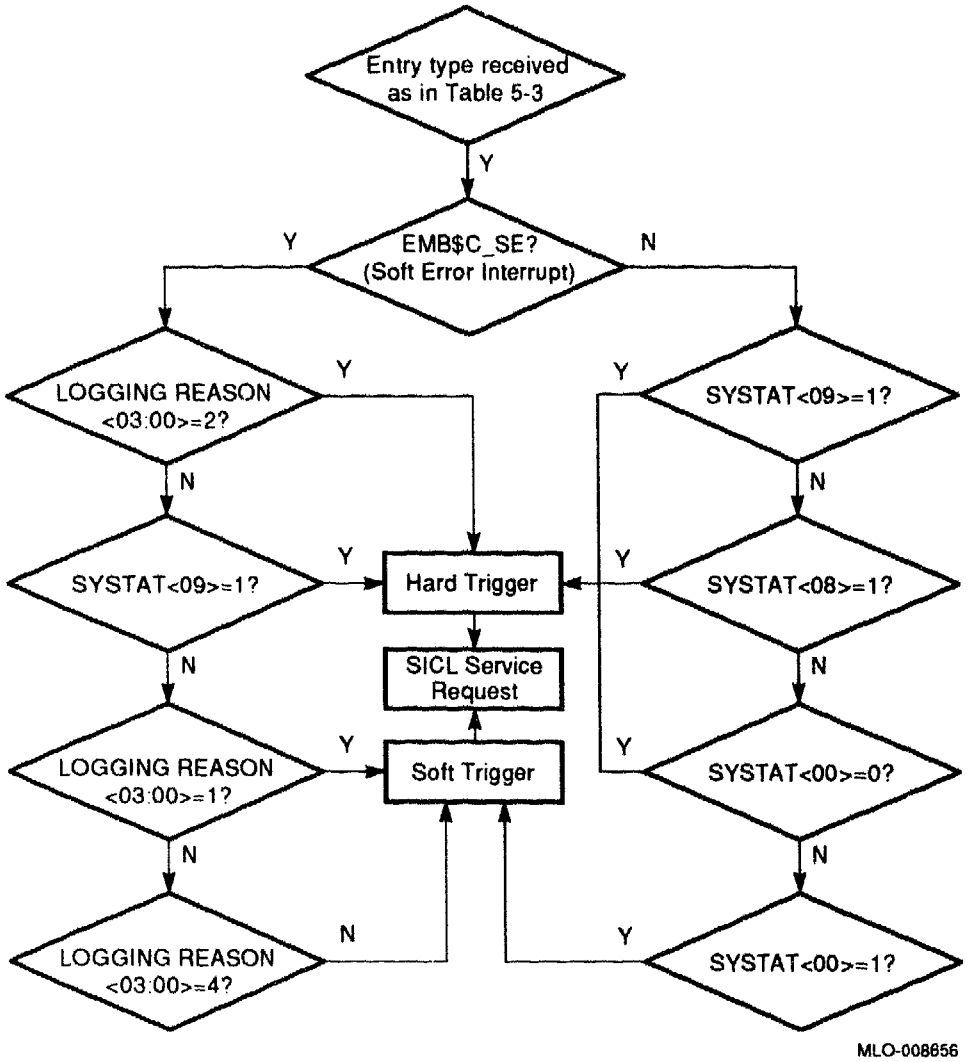
All lower level errors will ultimately set one of the conditions shown in Table 5–2. VAXsimPLUS will examine the conditions within a 24-hour period—thresholds are typically one or two flags or logging reason codes within that period.

Table 5–2 lists the conditions that will trigger VAXsimPLUS notification and updating. Figure 5–8 shows the flow for the VAXsimPLUS monitor trigger (for decision blocks with only one branch, the alternative is treated as an ignore condition). The entries ultimately are classified as either hard or soft. Errors that require corrective maintenance are classified as hard; while errors potentially requiring corrective maintenance are classified as soft.

Table 5–2 Conditions That Trigger VAXsimPLUS Notification and Updating

Condition	Description
SYSTAT <00> = 1	"Attempting recovery"
SYSTAT <00> = 0	"Full recovery or retry not possible"
SYSTAT <08> = 1	"Error threshold exceeded"
SYSTAT <09> = 1	"Page marked bad for uncorrectable ECC error in main memory"
SYSTAT <11> = 1	"Page mapout threshold for single bit ECC errors in main memory exceeded"
LOGGING REASON <3:0> = 1	"Memory CRD buffer full"
LOGGING REASON <3:0> = 2	"Generate report as a result of hard single address or multiple address DRAM memory fault"
LOGGING REASON <3:0> = 0, 3, 5–F	"Illegal LOGGING REASON"

Figure 5–8 Trigger Flow for the VAXsimPLUS Monitor



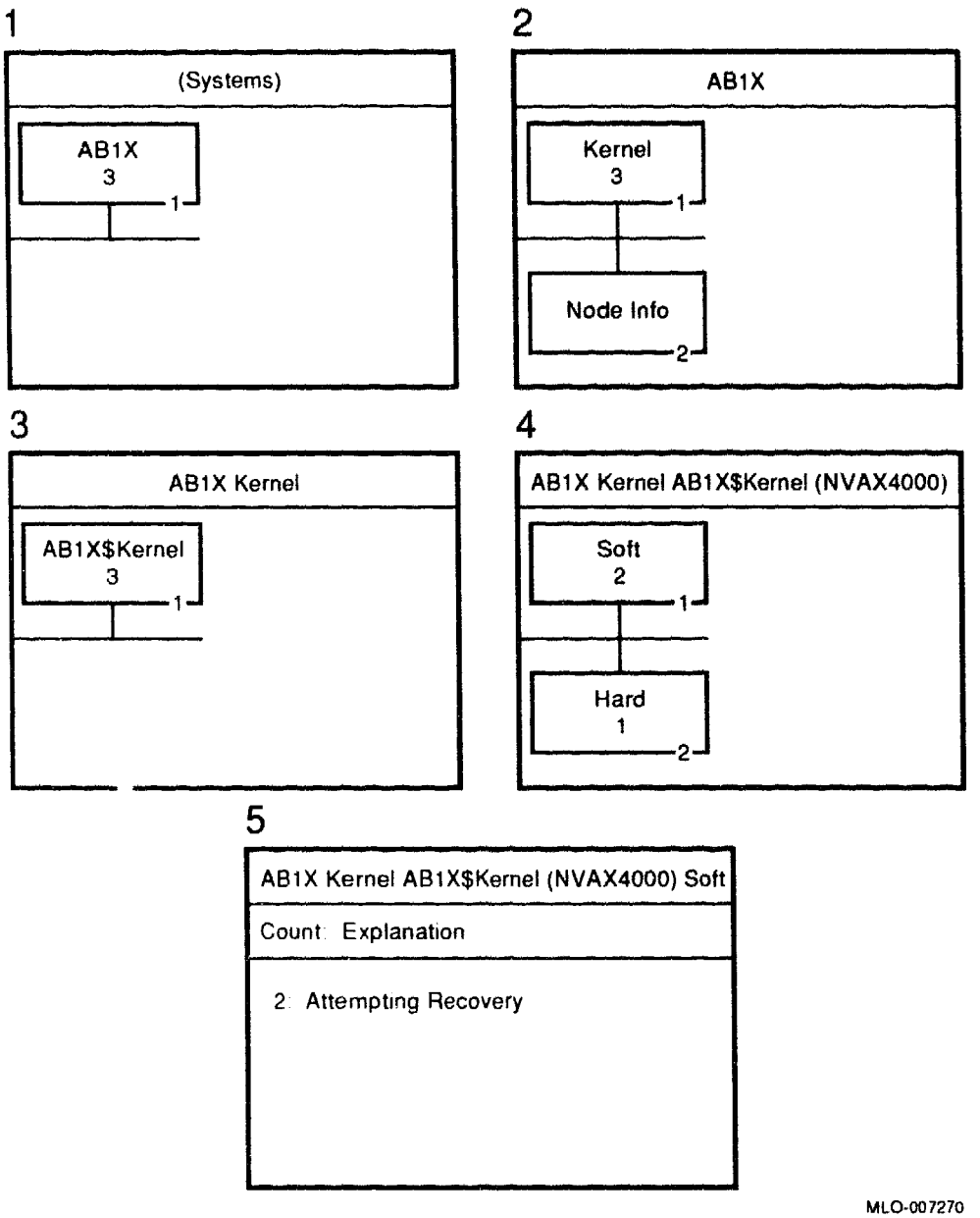
VAXsimPLUS triggering notifies the customer and Services using three message types: HARD, SOFT, and SICL Service Request. Each message contains the single STARS article theory number, as well as the SYSTAT or LOGGING REASON state. In addition, the SICL Service Request will have a Merged Error Log (MEL) datafile appended. Both hard and soft triggers will generate SICL Service Request messages.

Figure 5–9 shows the five VAXsimPLUS monitor screen displays. Table 5–3 provides a brief explanation of the five levels of screen displays.

Table 5–3 Five-Level VAXsimPLUS Monitor Screen Displays

Level	Explanation
1. System	The system level screen provides one box for each system being analyzed (in Figure 5–9 a single system is being analyzed). As with each screen level, the number of reported errors is displayed in the box. The boxes blink when the hard error thresholds are reached; the boxes are highlighted when the soft error thresholds are reached.
2. Subsystem	The subsystem level screen provides separate boxes for the kernel and node information. Other boxes that may be displayed are bus, disk, tape, etc.
3. Unit	The unit level screen provides a box for the kernel. If the subsystem has more than one unit or device with errors, those will be displayed as well.
4. Error Class	The error class level screen provides a box for both hard and soft errors.
5. Error Detail	Two error detail level screens (hard and soft) provide the number of reported errors along with a brief error description.

Figure 5–9 Five-Level VAXsimPLUS Monitor Display



Once notification occurs, the service engineer should examine the error log file (after using the ANALYZE/ERROR command) or read the appended Merged Error Log (MEL) file in the SICL service request message. (The MEL file is encrypted, refer to Section 5.2.9.1 for instructions in converting these files.)

Using the theory of interpretation provided in the previous sections, you can manually interpret the error logs.

Note

The interpretation theory provided in this manual is also a STARS article and can be accessed via the Decoder Kit. (Theory 30B01.xxx reproduces in full, Section 5.2 of this manual).

In summary, a service engineer should use VAXsimPLUS notification as follows:

1. Make sure all four message types are sent to the Field and System accounts.
2. Log into the Field or System account.
3. Read mail (look for the SICL service request message with its appended MEL file).
4. Convert the encrypted MEL file and use the theory provided in this manual to interpret the error log file.

5.2.9.1 Converting the SICL Service Request MEL File

Use the following procedure to convert the encrypted MEL file that is appended to the SICL service request message (MEL files can be converted on site or at a support center). Example 5–10 shows a sample SICL service request message and appended MEL file.

1. Extract the SICL mail message from mail.
2. Edit the extracted file to obtain the appended MEL file. The MEL file is the encrypted code that appears between the rows of asterisks and includes the words "SICL" and "end."
3. Convert the encrypted code to a binary file using the VAXsimPLUS decode command file as follows:

   $ MCR SDD$EXE:FMGR$SICL_DECODE [MEL filename] [binary filename]

4. Use the ANALYZE/ERROR command to produce an error log entry.

$ ANALYZE/ERROR [binary filename]

Example 5–10 SICL Service Request with Appended MEL File

```
From: AB1X::SDD$MANAGER      "VAXsimPLUS Message" 15-APR-1992 10:29:21.05
To: SYSTEM
CC:
Subj: SDD T2.0 Service Request - Analysis:[30B01.200]
********************************************************************************

                  VAXsimPLUS Notification Message

   VAXsimPLUS has detected that the following device needs attention:

       DEVICE:                 AB1X$KERNEL (NVAX4000)
       NODE:                   AB1X
       SYSTEM SERIAL NUMBER:   KA136H1520
       SYSTEM TYPE:            VAX 4000-600

                  VAXsimPLUS Diagnosis Information

Attn:          Field Service

Device:        AB1X$KERNEL (NVAX4000)
Count:         1.

Theory:        [30B01.200]

Evidence:  Urgent action required - AB1X$KERNEL Hard error(s):

          SYSTAT <9> = 1 - Page Marked Bad For Uncorrectable ECC Error In
                           Main Memory
********************************************************************************
%% SDD$PROFILE is defined to be NONE, no Customer Profile included in message %%
********************************************************************************
SICL
 134
M @( $_O  0=# 0        A$24U)3$\@(            %  @ G!::G+Y*5
M  @ 034N-2U-,2 7    &0\          @  !P   !@       :  0   "<<
M  !\F>] "    (  %/,%SP    #  RO R.P     \%31!03  !P  @ !
M (H   #0 0"  /S_S$X_\A  !            F 6  /CA"0     (P0
M"A(        %\   [P          0    '@U@. '        @%.^!0
M    (S+<]P ,12  P P                      , "          S ,13
FO@4          \         (     ?Y#P(%  " !_D/ @4  (
end
********************************************************************************
```

5.2.9.2 VAXsimPLUS Installation Tips

When installing VAXsimPLUS, the system will prompt you for information. You will need to know the serial number and system model number for the system on which you are installing VAXsimPLUS. The serial number is located on the front of the chassis at the bottom and to the left (the front door must be open). The system model number is attached to the outside of the door.

Also, if the system does not have dialout capability, you should answer no when asked if you want to enable SICL—if you enter yes, the system will attempt to send mail via DSNLink resulting in error messages. After VAXsimPLUS is installed you can activate SICL and customize the VAXsimPLUS mailing lists so that SICL messages are sent to an appropriate destination(s) on site. This way, SICL messages are received onsite without incurring error messages regarding remote link failures.

5.2.9.3 VAXsimPLUS Post-Installation Tips

Once VAXsimPLUS is installed, you can set up mailing lists to direct VAXsimPLUS messages to the appropriate destinations. If the system has no dialout capability, SICL messages should be directed to the System and/or Field account—this is good practice for systems with dialout and service center support as well.

In the example that follows, the four types of mailing lists are displayed and System and Field accounts are added to all four mailing lists using VAXSIM /FAULT_MANAGER commands.

Note

The commands can be abbreviated.

DSN%SICL appears under the SICL mailing list if you enabled SICL during installation.

```
$ VAXSIM/FAULT SHOW MAIL
-- FSE mailing list --

    FIELD

-- CUSTOMER mailing list --

    SYSTEM

-- MONITOR mailing list is empty --

-- SICL mailing list --

    DSN%SICL

$ VAXSIM/FAULT ADD SYSTEM ALL
$ VAXSIM/FAULT ADD FIELD ALL
$ VAXSIM/FAULT SHOW MAIL
-- FSE mailing list --

    FIELD
    SYSTEM

-- CUSTOMER mailing list --

    FIELD
    SYSTEM

-- MONITOR mailing list --

    FIELD
    SYSTEM

-- SICL mailing list --

    DSN%SICL
    FIELD
    SYSTEM
```

To activate SICL after installation, use the following command:

$ VAXSIM/FAULT SET SICL ON

VAXsimPLUS customer notification messages should display a phone number for the customer to call in the event the system needs service. Use the following commands to examine and set the phone number parameter:

```
$ VAXSIM/FAULT SHOW PARAMETER

(SET parameter)          (Parameter settings)

PHONE_NUMBER    Customer Service Phone Number is unknown
COPY            Automatic copying is OFF
SICL            System Initiated Call Logging is ON

SYSTEM_INFO  System info for AB1X
             Serial number   KA136H1520
             System type     VAX 4000-600
```

```
$ VAXSIM/FAULT SET PHONE 1-800-DIGITAL
```

Finally, the VAXSIMPLUS/MERGE command is useful in examining how a device is functioning in a cluster. The merge command collects the messages that are being sent to the other CPUs in the cluster.

5.2.10 Repair Data for Returning FRUs

When sending back an FRU for repair, include as much of the error log information as possible. If one or more error flags are set in a particular entry, record the mnemonic(s) of the register(s), the hex data, and error flag translation(s) on the repair tag. If an error address is valid, include the mnemonic, hex data, and translation on the repair tag as well. For memory and cache errors, include the syndrome and corrected-bit/bit-in-error information, along with the register mnemonic and hex data. Other registers which should be recorded for any entry type are SYSTAT, MEMCON and FOOTPRINT.

5.3 Power-On Self-Test (POST) and ROM-Based Diagnostic (RBD) Failures

If any of the tests fail, the test code displays on the console LED and, if specified in the firmware script, a diagnostic console printout displays in the format shown in Example 5–11.

Example 5–11 Sample Output with Errors

```
                ❶ ❷          ❸          ❹          ❺
? Test_Subtest_40_06  Loop_Subtest=00  Err_Type=FF   DE_Memory_count_pages.lis
  ❻          ❼
Vec=0000  Prev Errs=0004   P1=00000001   P2=00000002   P3=00000001  P4=00000000
P5=00000020  P6=00008000   P7=00000020   P8=00000000   P9=00000000 P10=00FCD44B
r0=00FF4008  r1=00000007   r2=00000000   r3=FFFFFFFF   r4=00000068  r5=00000000
r6=00000000  r7=00000002   r8=00FF4000   r9=20140758  r10=FFFFFFFE r11=FFFFFFFF
 dser=0000 cesr=00000000 intmsk=00 icsr=01 pcsts=FC00 pcadr=FFFFFFF8 pcctl=FC13
 cctl=00000021 bcetsts=0000 bcedsts=0000 cefsts=00000200 nests=00
 mmcdsr=01111000 mesr=00080000

>>>
```

Several lines are printed in the error display. The first line has eight column headings:

❶ *Test* identifies the diagnostic test, test ?40 in Example 5–11. Using Table 5–4, you can use the test number to point to possible problems in field replaceable units (FRUs).

❷ *Subtest log* is two hex digits identifying, usually within 10 instructions, where in the diagnostic the error occurred.

❸ *Loop_subtest_log* is an additional log generated out of the current test specified by the current test number and subtestlog. Usually these logs occur in common subroutines called from a diagnostic test.

❶ *Error_type* (diagnostic executive error) signals the diagnostic's state and any illegal behavior. This field indicates a condition that the diagnostic expects on detecting a failure. FE or EF in this field means that an unexpected exception or interrupt was detected. FF indicates an error as a result of normal testing, such as a miscompare. The possible codes are:

Error Code	Description
FF	Normal error exit from diagnostic
FE	Unanticipated interrupt
FD	Interrupt in cleanup routine
FC	Interrupt in interrupt handler
FB	Script requirements not met
FA	No such diagnostic
EF	Unanticipated exception in executive

❺ *ASCII messages* Shows the name of the listing file that contains the failed diagnostic.

❻ *Vec* identifies the SCB vector through which the unexpected exception or interrupt trapped, when the de_error field detects an unexpected exception or interrupt (FE or EF).

❼ *Prev_errs* is four hex digits showing the number of previous errors that have occurred (four in Example 5–11).

Lines 2 and 3 of the error printout are parameters 1 through 10. When the diagnostics are running normally, these parameters are the same parameters listed in Example 4–3.

When returning a module for repair, always record the the test number, subtest, and Err_type from line 1 of the printout. Also record the Vec from line 2. If possible, record additional information. If the error can be saved onto a printer, then enclose the full printout with the failing module.

Note

Do not confuse the countdown pattern of powerup tests with the test number. In the following the last countdown was 58; this number should not be reported! The test number was 31.

The countdown pattern is used to indicate progress in the power-up tests. The actual true test number associated with a countdown value can change from one release of the ROM code to another. For example:

```
KA50-A T1.2-156, VMB 2.14
Performing normal system tests.
72..71..70..69..68..67..66..65..64..63..62..61..60..59..58..

? Test_Subtest_31_06  Loop_Subtest=05  Err_Type=FF   DE_Memory_Setup_CSRs.lis
Vec=0000  Prev_Errs=0000   P1=C94AC94A   P2=01000000   P3=00000002  P4=00000000

Minimum recording for this error is:

 Test = 31
 Subtest = 6
 Loop_subtest = 5
 Err_type = FF
 Vec = 0.
```

Table 5-4 lists the hex LED display, the default action on errors, and the most likely unit that needs replacing reading from left to right. Example, 1,4 indicates 1 is most likely, then 4. The Default on Error column refers to the action taken by the diagnostic executive when the test fails in the script.

Memory tests are usually treated differently; when an error occurs, the memory tests usually try to continue and mark the bitmap. Test 40 reports failing pages in the bitmap.

When any memory test fails, always do a SHOW MEMORY to help identify the FRU. SHOW MEMORY will identify the FRU to a SET of SIMMs or to an individual SIMM if possible.

If a single set of SIMMs is present, and replacing a suspected bad SIMM or set does not fix the problem, assume that the system board is bad. Always check the seating of SIMMs before replacing. If nonvolatile data is lost after powerup or you always get a request to select a language at powerup, the battery may be bad.

Table 5–4 shows the various LED values and console terminal displays as they point to problems in field-replaceable units (FRUs).

Table 5–4 KA50/51/55/56 Console Displays as Pointers to FRUs

On Error Hex LED	Normal Console Display	Default Action on Error	Failing Test Number	Test Description	FRU[1]
Power-Up Tests (Script A1)					
F	None	Loop	None	Power up	1, 4
E	None	None	None	ROM code execution begun	1, 4
D	None	Loop	None	Wait for power	1, 4
B	72	Cont	9D	Utility	1, 4
B	71	Cont	42	Chk_for_interrupts	1, 3
9	70	Cont	35	B_Cache_diag_mode	1
B	69	Cont	33	NMC_powerup	1
B	68	Cont	32	NMC_registers	1
B	67	Cont	D0	V_Cache_diag_mode	1
B	66	Cont	D2	O_bit_Diag_mode	1
B	65	Cont	DF	O-bit_debug	1
B	64	Cont	46	P_cache_diag_mode	1
9	63	Cont	35	B_cache_diag_mode	1
9	62	Cont	DE	B_Cache_tag_debug	1
9	61	Cont	DD	B_Cache_data_debug	1
9	60	Cont	DA	PB_Flush_cache	1
8	59	Halt	DC	NO_Memory_present	2, 1
8	58	Cont	31	Memory_Setup_CSRs	2, 1
8	57	Halt	30	Memory_Init_Bitmap	2, 1
7	56	Cont	91	CQBIC_powerup	1, 3

[1]Field-replaceable unit key:

1 = KA50
2 = MS44
3 = Q22–bus option
4 = System power supply
5 = SCSI device or 2 devices with same target id
6 = ASYNC option board
7 = COMM option board (SYNC)
8 = SHAC option board

(continued on next page)

Table 5–4 (Cont.) KA50/51/55/56 Console Displays as Pointers to FRUs

On Error Hex LED	Normal Console Display	Default Action on Error	Failing Test Number	Test Description	FRU[1]
Power-Up Tests (Script A1)					
7	55	Cont	90	CQBIC_registers	1
C	54	Cont	C6	SSC_powerup	1
C	53	Cont	52	SSC_Prog_timers	1
C	52	Cont	52	SSC_Prog_timers	1
C	51	Cont	53	SSC_TOY_Clock	1
C	50	Cont	C1	SSC_RAM_Data	1
C	49	Cont	34	SSC_ROM	1
C	48	Cont	C5	SSC_registers	1
B	47	Cont	55	Interval_Timer	1
8	46	Cont	4F	Memory_Data	2, 1
8	45	Cont	4E	Memory_Byte	2, 1
8	44	Cont	4B	Memory_Byte_Errors	2, 1
8	43	Cont	4A	Memory_ECC_SBEs	2, 1
8	42	Cont	4C	Memory_ECC_Logic	2, 1
8	41	Cont	48	Memory_Addr_shorts	2, 1
8	40	Cont	48	Memory_addr_shorts	2, 1
8	39	Cont	48	Memory_addr_shorts	2, 1
8	38	Cont	48	Memory_addr_shorts	2, 1
8	37	Cont	48	Memory_addr_shorts	2, 1
8	36	Cont	48	Memory_addr_shorts	2, 1

[1]Field-replaceable unit key:

1 = KA50
2 = MS44
3 = Q22–bus option
4 = System power supply
5 = SCSI device or 2 devices with same target id
6 = ASYNC option board
7 = COMM option board (SYNC)
8 = SHAC option board

(continued on next page)

Table 5–4 (Cont.) KA50/51/55/56 Console Displays as Pointers to FRUs

On Error Hex LED	Normal Console Display	Default Action on Error	Failing Test Number	Test Description	FRU[1]
Power-Up Tests (Script A1)					
8	35	Cont	48	Memory_addr_shorts	2, 1
8	34	Cont	48	Memory_addr_shorts	2, 1
8	33	Cont	4D	Memory_address	2, 1
8	32	Cont	47	Memory_Refresh	2, 1
8	31	Halt	40	Memory_count_pages	2, 1
8	30	Cont	40	Memory_count_pages	2, 1
6	29	Cont	E4	DZ	1
B	28	Cont	54	Virtual_Mode	1
9	27	Cont	37	Cache_W_memory	1, 2
C	26	Cont	C2	SSC_RAM_Data_Addr	1
7	25	Cont	80	CQBIC_memory	1, 2
9	24	Cont	37	Cache_w_memory	1, 2
A	23	Cont	51	FPA	1
5	22	Cont	E2	SCSI_MAP	1
5	21	Cont	E0	SCSI	1, 5
4	20	Cont	5F	SGEC	1
5	19	Cont	5C	SHAC	8, 1
B	18	Cont	9A	INTERACTION	1
7	17	Cont	83	QZA_Intlpbck1	3
7	16	Cont	84	QZA_Intlpbck2	3

[1]Field-replaceable unit key:

1 = KA50
2 = MS44
3 = Q22-bus option
4 = System power supply
5 = SCSI device or 2 devices with same target id
6 = ASYNC option board
7 = COMM option board (SYNC)
8 = SHAC option board

(continued on next page)

Table 5–4 (Cont.) KA50/51/55/56 Console Displays as Pointers to FRUs

On Error Hex LED	Normal Console Display	Default Action on Error	Failing Test Number	Test Description	FRU[1]
Power-Up Tests (Script A1)					
7	15	Cont	85	QZA_memory	3
7	14	Cont	86	QZA_DMA	3
7	13	Cont	63	QDSS_any	3
7	12	Cont	63	QDSS_any	3
B	11	Cont	DB	Speed	1
7	10	Cont	EC	ASYNC	6, 1
7	09	Cont	E8	SYNC	7, 1
C	08	Cont	52	SSC_Prog_timers	1
C	07	Cont	52	SSC_Prog_timers	1
C	06	Cont	53	SSC_TOY_Clock	1
C	05	Cont	C1	SSC_RAM_Data	1
B	04	Cont	55	Interval_Timer	1
B	03	Cont	41	Board_Reset	1, 3

[1]Field-replaceable unit key:

1 = KA50
2 = MS44
3 = Q22–bus option
4 = System power supply
5 = SCSI device or 2 devices with same target id
6 = ASYNC option board
7 = COMM option board (SYNC)
8 = SHAC option board

5.3.1 FE Utility

In addition to the diagnostic console display and the LED code, the FE utility dumps the diagnostic state to the console (Example 5–12). This state indicates the major and minor test code of the test that failed, the 10 parameters associated with the test, and additional diagnostic state information.

Example 5–12 FE Utility Example

```
>>>T FE

Bitmap=00FF3000, Length=00001000, Checksum=807F, Busmap=00FF8000
Test_number=00, Subtest=00, Loop_Subtest=00, Error_type=00
Error_vector=0060, Severity=02, Last_exception_PC=20057C37
Total_error_count=0004, Led_display=08, Console_display=81, save_mchk_code=00
parameter_1=00000082  2=00000000  3=2000146A  4=00000000  5=20051400
parameter_6=00000001  7=00000000  8=00000020  9=00000000 10=00000000
previous_errors, Test  Subtest  Loop_Subtest  Error_Type
  Test_81_02_00_FE   Test_40_06_00_FF   Test_E8_03_00_FF   Test_E4_02_00_FF
Flags=FFFF FFFCFC 0408443E   BCache_Disable=06  KA50  128KB BC  14.0 ns
Return_stack=201406CC, Subtest_pc=2005D7FF, Timeout=000007D0

>>>
```

5.3.2 Overriding Halt Protection

The ROM diagnostics are run in halt-protected space during execution after power-up of the system. During this time they cannot normally be halted with the BREAK key or the HALT button. After power-up is complete, all diagnostics including the power-up script (A1) or (0) are run with halts enabled allowing a user to stop a script or test. The preferred method to stop scripts is to use CONTROL C first.

5.3.3 Isolating Memory Failures

This section describes procedures for isolating memory subsystem failures.

Memory tests numbers are DC, 31, 30, 4F, 4E, 4B, 4A, 4C, 48, 4D, 47 and 40. All of these tests are run during power-up.

Normally, if one or more of these tests fail during power-up at the end of power-up the diagnostic executive will execute the SHOW MEMORY command automatically to help identify the memory failure. In all cases of a memory failure, the primary means to isolate to the FRU is to use the SHOW MEMORY command.

Example 5–13 shows a memory failure due to a missing SIMM. In this case only one 16-MB set (4 SIMMs of 4 MB each) is present, and one of these is missing. Because of this, test DC fails and the power-up script is halted because no usable memory is present. At the end, SHOW MEMORY is automatically executed before the test is halted. In this example, SIMM set 1 (1E,1F,1G,1H) is present but SIMM 1F is either missing or not correctly installed in its socket.

Example 5–13 Failure Due to a Missing SIMM (One 16 Mbyte Set)

```
KA50-A V1.2, VMB 2.14
Performing normal system tests.
72..71..70..69..68..67..66..65..64..63..62..61..60..59..

? Test_Subtest_DC_88  Loop_Subtest=05  Err_Type=FF   DE_NO_Memory_present.lis
Vec=0000  Prev_Errs=0000   P1=C90AC90A   P2=00000000   P3=00000000  P4=00001006
P5=00000000  P6=7F7F7F33   P7=00000000   P8=00000000   P9=FFFF0000 P10=200636E4
r0=00000008  r1=21018000   r2=C90AC90A   r3=80000000   r4=01000000  r5=04000000
r6=00000002  r7=00000000   r8=00000000   r9=20140758  r10=FFFFFFFE r11=FFFFFFFF
 dser=0000 cesr=00000000 intmsk=00 icsr=01 pcsts=FA00 pcadr=FFFFFFF8 pcctl=FE13
 cctl=00000006 bcetsts=03E0 bcedsts=0F00 cefsts=0001EC20 nests=00
 mmcdsr=01FFFE40 mesr=00000000

Error:  SIMM Set 1 (1E,1F,1G,1H), SSR = C90A
 SIMM_1E = 16MB      SIMM_1F = 00MB ??   SIMM_1G = 16MB      SIMM_1H = 16MB

Total of 0MB, 0 good pages, 0 bad pages, 0 reserved pages
Normal operation not possible.
>>>
```

Note

The value listed by each SIMM is either 16 MB or 64 MB which indicates the full size of the set of SIMMs if all are present.

ACTION:

- If SIMM 1F is missing, install a SIMM.
- If SIMM 1F is present in socket, reseat the SIMM.
- If reseating SIMM 1F does not fix the problem, replace the SIMM with a new SIMM.
- At this point the system board is probably bad. If no new system board is available, try moving the SIMMs to the other set of sockets.

Example 5–14 shows a memory failure due to a missing SIMM. In this case two 16-MB sets (4 SIMMs of 4 MB each) are present with one SIMM missing in Set 1. Since one set of memory is fully usable, all testing is completed. At the end SHOW MEMORY is automatically executed as before. SIMM 1H is missing or not installed correctly. The system is usable but with only 16 MB of memory instead of 32 MB.

Example 5–14 Failure Due to a Missing SIMM (Two 16 Mbyte Sets)

```
KA50-A T1.2-156, VMB 2.14
Performing normal system tests.
72..71..70..69..68..67..66..65..64..63..62..61..60..59..58..

? Test_Subtest_31_06  Loop_Subtest=05  Err_Type=FF   DE_Memory_Setup_CSRs.lis
Vec=0000  Prev_Errs=0000   P1=C94AC94A   P2=01000000   P3=00000002  P4=00000000
P5=25800000  P6=FFFFFFFF   P7=00000000   P8=00000000   P9=0000C94A P10=C94AC14A
r0=00000008  r1=21018000   r2=C94AC94A   r3=81000000   r4=01000000  r5=04000000
r6=00000002  r7=21018048   r8=00000000   r9=20140758  r10=FFFFFFFE r11=FFFFFFFF
 dser=0000 cesr=00000000 intmsk=00 icsr=01 pcsts=FA00 pcadr=FFFFFFF8 pcctl=FE13
 cctl=00000006 bcetsts=0360 bcedsts=0F00 cefsts=00206E20 nests=00
 mmcdsr=01FFFE00 mesr=00000000

57..56..55..54..53..52..51..50..49..48..47..46..45..44..43..42..
41..40..39..38..37..36..35..34..33..32..31..30..29..28..27..26..
25..24..23..22..21..20..19..18..17..16..15..14..13..12..11..10..
09..08..07..06..05..04..03..

16 MB RAM, SIMM Set (0A,0B,0C,0D) present
Memory Set 0: 00000000 to 00FFFFFF, 16MB, 32768 good pages, 0 bad pages

Error:  SIMM Set 1 (1E,1F,1G,1H), SSR = C94A
 SIMM_1E = 16MB      SIMM_1F = 16MB      SIMM_1G = 16MB      SIMM_1H = 00MB ??

Total of 16MB, 32768 good pages, 0 bad pages, 104 reserved pages

Normal operation not possible.

>>>
```

ACTION:

- If SIMM 1H is missing, install a SIMM.
- If SIMM 1H is present in socket, reseat the SIMM.
- If reseating SIMM 1H does not fix the problem then replace the SIMM with a new SIMM.
- At this point the system board is probably bad.

Example 5–15 shows a memory failure due to a bad SIMM. In this case two 16-MB sets (4 SIMMs of 4 MB each) are present with one bad SIMM. SIMM 1H is marked as being bad.

Example 5–15 Failure Due to a Bad SIMM

```
KA50-A V1.2, VMB 2.14
Performing normal system tests.
72..71..70..69..68..67..66..65..64..63..62..61..60..59..58..57..
56..55..54..53..52..51..50..49..48..47..46..45..44..43..42..41..
40..39..38..37..36..35..34..33..32..31..30..

? Test_Subtest_40_06  Loop_Subtest=00  Err_Type=FF   DE_Memory_count_pages.lis

29..28..27..26..25..24..23..22..21..20..19..18..17..16..15..14..
13..12..11..10..09..08..07..06..05..04..03..

16 MB RAM, SIMM Set (0A,0B,0C,0D) present
Memory Set 0: 00000000 to 00FFFFFF, 16MB, 32768 good pages, 0 bad pages

Error:  SIMM Set 1 (1E,1F,1G,1H), SSR = C14A
 SIMM_1E = 16MB      SIMM_1F = 16MB      SIMM_1G = 16MB      SIMM_1H = 16MB ??
Memory Set 1: 01000000 to 01FFFFFF, 16MB, 0 good pages, 32768 bad pages

Total of 32MB, 32768 good pages, 32768 bad pages, 112 reserved pages

>>>
```

ACTION

- Reseat the SIMM 1H.
- If reseating SIMM 1H does not fix the problem then replace the SIMM with a new SIMM.
- At this point the system board is probably bad.

Example 5–16 indicates that a large SIMM is mixed in with a set of small SIMMs. If a full set of SIMMs is present and one or more is the incorrect size then the diagnostic code will configure the set as a small set and run the tests. In this example, SIMM 1G is the wrong size SIMM. Because the set is configured as a small set, it is usable as a 16-MB set.

Example 5–16 SIMM Wrong Size

```
Error:  SIMM Set 1 (1E,1F,1G,1H), SSR = C14A
 SIMM_1E = 16MB       SIMM_1F = 16MB       SIMM_1G = 64MB ??   SIMM_1H = 16MB
Memory Set 1: 01000000 to 01FFFFFF, 16MB, 32768  good pages, 0 bad pages
```

ACTION:

Replace SIMM 1G with one of the correct size.

The diagnostics cannot always determine which SIMM caused a failure. If this occurs and more than one set is present, usually the failing set can be identified by using the **SHOW MEMORY** command.

```
>>>SHOW MEMORY

16 MB RAM, SIMM Set (0A,0B,0C,0D) present
Memory Set 0: 00000000 to 00FFFFFF, 16MB, 32768 good pages, 0 bad pages

16 MB RAM, SIMM Set (1E,1F,1G,1H) present
Memory Set 1: 01000000 to 01FFFFFF, 16MB, 0 good pages, 32768 bad pages

Total of 32MB, 32768 good pages, 32768 bad pages, 112 reserved pages

>>>
```

ACTION:

Replace SIMM set 1 (1E,1F,1G,1H).

After installing a new set of SIMMs and successfully running power-up tests, run memory test script A8.

>>>T A8

Note

Script A9 is another memory test script. This script will stop on the first occurrence of any error. It will also stop on a soft error. If a failure occurs in A9 and if A9 then runs successfully 10 times and script A8 runs without error the problem is a soft error and does not require action.

Note

If a memory failure is marked in the bitmap, it will not be erased until either the system is powered up or the bitmap placing test is run with

parameter P4 set to 0 to rebuild the bitmap.

To force rebuilding the bitmap to all good memory, enter the following commands:

```
T 30 0 0 0 0   ; T 30 will not work by itself.
T 0            ; rerun powerup script
```

5.4 Using MOP Ethernet Functions to Isolate Failures

The console requester can receive LOOPED_DATA messages from the server by sending out a LOOP_DATA message using NCP to set this up. An example follows.

Identify the Ethernet adapter address for the system under test (system 1) and attempt to boot over the network.

```
***system 1 (system under test)***

>>>SHOW ETHERNET
Ethernet Adapter
-EZA0 (08-00-2B-28-18-2C)
>>>BOOT EZA0
(BOOT/R5:2 EZA0)

  2..
-EZA0
Retrying network bootstrap.
```

Unless the system is able to boot, the "Retrying network bootstrap" message will display every 8–12 minutes.

Identify the system's Ethernet circuit and circuit state, enter the SHOW KNOWN CIRCUITS command from the system conducting the test (system 2).

```
***system 2 (system conducting test)***

$ MCR NCP
NCP>SHOW KNOWN CIRCUITS

Known Circuit Volatile Summary as of 14-NOV-1991 16:01:53

   Circuit          State                 Loopback     Adjacent
                                            Name      Routing Node

  ISA-0             on                                25.1023 (LAR25)
```

```
NCP>SET CIRCUIT ISA-0 STATE OFF
NCP>SET CIRCUIT ISA-0 SERVICE ENABLED
NCP>SET CIRCUIT ISA-0 STATE ON
NCP>LOOP CIRCUIT ISA-0 PHYSICAL ADDRESS 08-00-2B-28-18-2C
WITH ZEROES
NCP>EXIT
$
```

If the loopback message was received successfully, the NCP prompt will reappear with no messages.

The following two examples show how to perform the Loopback Assist Function using another node on the network as an assistant (system 3) and the system under test as the destination. Both the assistant and the system under test are attempting to boot from the network. We will also need the physical address of the assistant node.

```
***system #3 (loopback assistant)***

>>>SHOW ETHERNET
Ethernet Adapter
-EZA0 (08-00-2B-1E-76-9E)
>>>b eza0
(BOOT/R5:2 EZA0)

  2..
-EZA0
Retrying network bootstrap.

***system 2***

NCP>LOOP CIRCUIT ISA-0 PHYSICAL ADDRESS 08-00-2b-28-18-2C ASSISTANT PHYSICAL
ADDRESS 08-00-2B-1E-76-9E WITH MIXED COUNT 20 LENGTH 200 HELP FULL
NCP>
```

Instead of using the physical address, you could use the assistant node's area address. When using the area address, system 3 is running the OpenVMS operating system.

```
***system 3***
$MCR NCP

NCP>SHOW NODE KLATCH

Node Volatile Summary as of 27-FEB-1992 21:04:11

Executor node = 25.900 (KLATCH)

State                     = on
Identification            = DECnet-VAX V5.4-1,  OpenVMS V5.4-2
Active links              = 2

NCP>SHOW KNOWN LINES CHARACTERISTICS

Known Line Volatile Characteristics as of 27-FEB-1992 11:20:50

Line = ISA-0
```

```
Receive buffers          = 6
Controller               = normal
Protocol                 = Ethernet
Service timer            = 4000
Hardware address         = 08-00-2B-1E-76-9E
Device buffer size       = 1498

NCP>SET CIRCUIT ISA-0 STATE OFF
NCP>SET CIRCUIT ISA-0 SERVICE ENABLED
NCP>SET CIRCUIT ISA-0 STATE ON
NCP>EXIT
$

***system 2***
$ MCR NCP
NCP>LOOP CIRCUIT ISA-0 PHYSICAL ADDRESS 08-00-2B-28-18-2C ASSISTANT MODE 25.900
WITH MIXED COUNT 20 LENGTH 200 HELP FULL
NCP>EXIT
$
```

Note

The kernel's Ethernet buffer is 1024 bytes deep for the LOOP functions and will not support the maximum 1500-byte transfer length.

In order to verify that the address is reaching this node, a remote node can examine the status of the periodic SYSTEM_IDs sent by the KA50/51/55/56 Ethernet server. The SYSTEM_ID is sent every 8–12 minutes using NCP as in the following example:

```
***system 2***

$ MCR NCP
NCP>SET MODULE CONFIGURATOR CIRCUIT ISA-0 SURVEILLANCE ENABLED
NCP>SHOW MODULE CONFIGURATOR KNOWN CIRCUITS STATUS TO ETHER.LIS
NCP>EXIT
$ TYPE ETHER.LIS

Circuit name             = ISA-0
Surveillance flag        = enabled
Elapsed time             = 00:09:37
Physical address         = 08-00-2B-28-18-2C
Time of last report      = 27-Feb 11:50:34
Maintenance version      = V4.0.0
Function list            = Loop, Multi-block loader, Boot, Data link counters
Hardware address         = 08-00-2B-28-18-2C
Device type              = ISA
```

Depending on your network, the file used to receive the output from the SHOW MODULE CONFIGURATOR command may contain many entries, most of which do not apply to the system you are testing. It is helpful to use an editor to search the file for the Ethernet hardware address of the system under test.

Existence of the hardware address verifies that you are able to receive the address from the system under test.

5.5 Interpreting User Environmental Test Package (UETP) OpenVMS Failures

When UETP encounters an error, it reacts like a user program. It either returns an error message and continues, or it reports a fatal error and terminates the image or phase. In either case, UETP assumes the hardware is operating properly and it does not attempt to diagnose the error.

If the cause of an error is not readily apparent, use the following methods to diagnose the error:

- *OpenVMS Error Log Utility*—Run the Error Log Utility to obtain a detailed report of hardware and system errors. Error log reports provide information about the state of the hardware device and I/O request at the time of each error. For information about running the Error Log Utility, refer to the *OpenVMS Error Log Utility Manual* and Section 5.2 of this manual.
- *Diagnostic facilities*—Use the diagnostic facilities to test exhaustively a device or medium to isolate the source of the error.

5.5.1 Interpreting UETP Output

You can monitor the progress of UETP tests at the terminal from which they were started. This terminal always displays status information, such as messages that announce the beginning and end of each phase and messages that signal an error.

The tests send other types of output to various log files, depending on how you started the tests. The log files contain output generated by the test procedures. Even if UETP completes successfully, with no errors displayed at the terminal, it is good practice to check these log files for errors. Furthermore, when errors are displayed at the terminal, check the log files for more information about their origin and nature.

5.5.1.1 UETP Log Files

UETP stores all information generated by all UETP tests and phases from its current run in one or more UETP.LOG files, and it stores the information from the previous run in one or more OLDUETP.LOG files. If a run of UETP involves multiple passes, there will be one UETP.LOG or one OLDUETP.LOG file for each pass.

At the beginning of a run, UETP deletes all OLDUETP.LOG files, and renames any UETP.LOG files to OLDUETP.LOG. Then UETP creates a new UETP.LOG file and stores the information from the current pass in the new file. Subsequent passes of UETP create higher versions of UETP.LOG. Thus, at the end of a run of UETP that involves multiple passes, there is one UETP.LOG file for each pass. In producing the files UETP.LOG and OLDUETP.LOG, UETP provides the output from the two most recent runs.

If the run involves multiple passes, UETP.LOG contains information from all the passes. However, only information from the latest run is stored in this file. Information from the previous run is stored in a file named OLDUETP.LOG. Using these two files, UETP provides the output from its tests and phases from the two most recent runs.

The cluster test creates a NETSERVER.LOG file in SYS$TEST for each pass on each system included in the run. If the test is unable to report errors (for example, if the connection to another node is lost), the NETSERVER.LOG file on that node contains the result of the test run on that node. UETP does not purge or delete NETSERVER.LOG files; therefore, you must delete them occasionally to recover disk space.

If a UETP run does not complete normally, SYS$TEST might contain other log files. Ordinarily these log files are concatenated and placed within UETP.LOG. You can use any log files that appear on the system disk for error checking, but you must delete these log files before you run any new tests. You may delete these log files yourself or rerun the entire UETP, which checks for old UETP.LOG files and deletes them.

5.5.1.2 Possible UETP Errors

This section is intended to help you identify problems you might encounter running UETP.

The following are the most common failures encountered while running UETP:

- Wrong quotas, privileges, or account
- UETINIT01 failure
- Ethernet device allocated or in use by another application
- Insufficient disk space
- Incorrect VAXcluster setup
- Problems during the load test
- DECnet–VAX error
- Lack of default access for the FAL object

- Errors logged but not displayed
- No PCB or swap slots
- Hangs
- Bug checks and machine checks

For more information refer to the *VAX 3520, 3540 OpenVMS Installation and Operations (ZKS166)* manual.

5.6 Using Loopback Tests to Isolate Failures

You can use external loopback tests to isolate problems with the console port, and Ethernet controller (SGEC chip).

5.6.1 Testing the Console Port

To test the console port at power-up, set the Power-Up Mode switch on the console module to the Loop Back Test Mode position (bottom) and install an H3103 loopback connector into the MMJ. The H3103 connects the console port transmit and receive lines. At power-up, the SLU_EXT_LOOPBACK test then runs a continuous loopback test.

While the test is running, the LED display on the console module should alternate between 6 and 3. A value of 6 latched in the display indicates a test failure. If the test fails, one of the following parts is faulty: the KA50/51/55/56 or the cabling.

To test out to the end of the console terminal cable:

1. Plug the MMJ end of the console terminal cable into the back BA42B.
2. Disconnect the other end of the cable from the terminal.
3. Place an H8572 adapter into the disconnected end of the cable.
4. Connect the H3103 to the H8572.
5. Cycle power and observe the LED.

5.6.2 Embedded Ethernet Loopback Testing

Note

Before running Ethernet loopback tests, check that the problem is not due to a missing terminator on a ThinWire T-connector.

Test 5F is the internal loopback test for SGEC (Ethernet controller).

```
>>>T 5F
```

For an external SGEC loopback, enter "1".

```
>>>T 5F 1
```

Before running test 5F on the ThinWire Ethernet port, connect an H8223 T-connector with two H8225 terminators.

Before running test 5F on the standard Ethernet port, you must have a 12–22196–02 loopback connector installed.

Note

Make sure the Ethernet Connector Switch is set for the correct Ethernet port.

T 59 polls other nodes on Ethernet to verify SGEC functionality. The Ethernet cable must be connected to a functioning Ethernet. A series of MOP messages are generated; look for response messages from other nodes.

```
>>>T 59

Reply received from node: AA-00-04-00-FC-64
Total responses: 1
Reply received from node: AA-00-04-00-47-16
Total responses: 2
Reply received from node: 08-00-2B-15-48-70
Total responses: 3
   .
   .
   .
Reply received from node: AA-00-04-00-17-14
Total responses: 25
>>>
```

Table 5–5 Loopback Connectors for Common Devices

Device	Module Loopback	Cable Loopback
CXA16/CXB16	H3103 + H8572[1]	–
CXY08	H3046 (50-pin)	H3197 (25-pin)
DIV32	H3072	–
DPV11	12–15336–10 or H325	H329 (12–27351–01)
DRQB3	–	17–01481–01 (from port 1 to port 2)
DRV1W	70–24767–01	–
DZQ11	12–15336–10 or H325	H329 (12–27351–01)
Ethernet[2]	–	–
IBQ01	IBQ01–TA	–
IEQ11	17–01988–01	–
KMV1A	H3255	H3251
KZQSA	12–30552–01	–
LPV11	12–15336–11	–

[1]Use the appropriate cable to connect transmit-to-receive lines. H3101 and H3103 are double-ended cable connectors.

[2]For ThinWire, use H8223–00 plus two H8225–00 terminators. For standard Ethernet, use 12–22196–02.

6

FEPROM Firmware Update

Note

The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95, and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, will appear on the console and/or printouts from time to time.

KA50/51/55/56 firmware is located on four chips, each 128 K by 8 bits of FLASH programmable EPROMs, for a total of 512 Kbytes of ROM. (A FLASH EPROM (FEPROM) is a programmable read-only memory that uses electrical (bulk) erasure rather than ultraviolet erasure.)

FEPROMs provide nonvolatile storage of the CPU power-up diagnostics, console interface, and operating system primary bootstrap (VMB). An advantage of this technology is that the entire image in the FEPROMs may be erased, reprogrammed, and verified in place without removing the CPU module or replacing components.

A slight disadvantage to the FEPROM technology is that the entire part must be erased before reprogramming. Hence, there is a small "window of vulnerability" when the CPU has inoperable firmware. Normally, this window is less than 30 seconds. Nonetheless, an update should be allowed to execute undisturbed.

Firmware updates are provided through a package called the Firmware Update Utility. A Firmware Update Utility contains a bootable image, which can be booted from tape or Ethernet, that performs the FEPROM update. Firmware update packages, like software, are distributed through Digital's SSB. Service engineers are notified of updates through a service blitz or Engineering Change Order (ECO)/Field Change Order (FCO) notification.

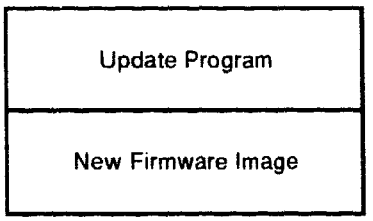
Note

The NVAX CPU chip has an area called the Patchable Control Store (PCS), which can be used to update the microcode for the CPU chip.

Updates to the PCS require a new version of the firmware.

A Firmware Update Utility image consists of two parts, the update program and the new firmware, as shown in Figure 6–1. The update program uniformly programs, erases, reprograms, and verifies the entire FEPROM.

Figure 6–1 Firmware Update Utility Layout



MLO-007271

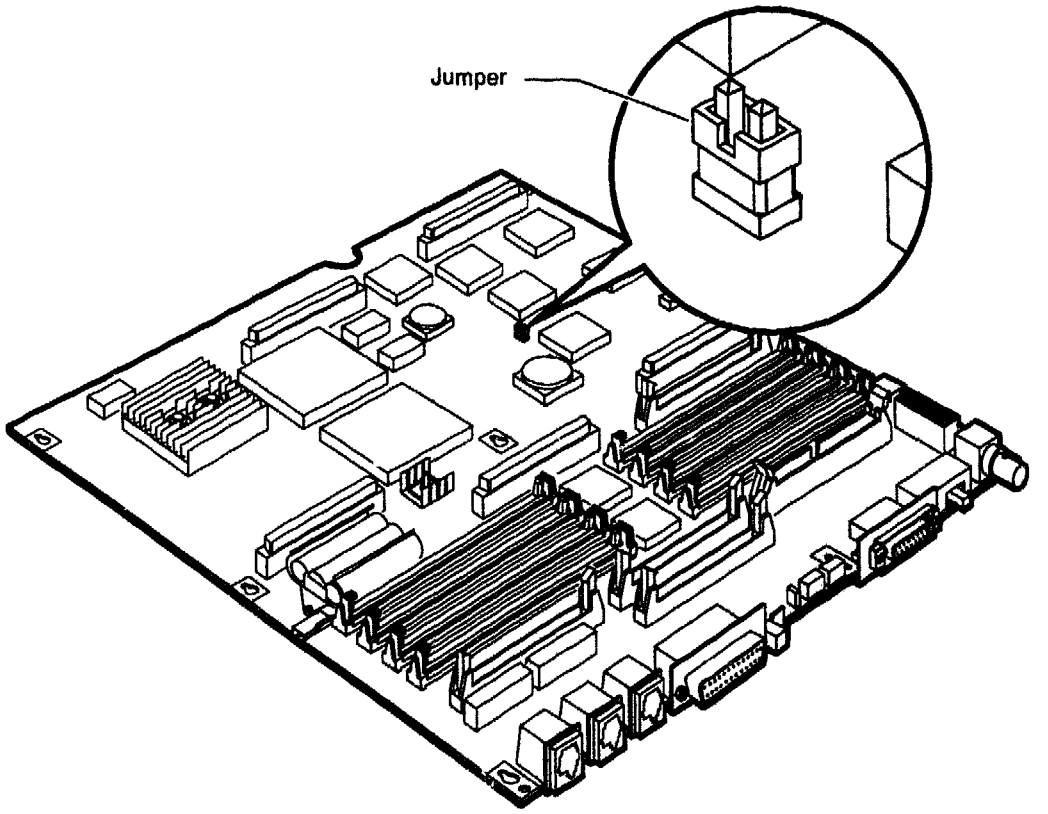
Once the update has completed successfully, normal operation of the system may continue. The operator may then either halt or reset the system and reboot the operating system.

6.1 Preparing the Processor for a FEPROM Update

Complete the following steps to prepare the processor for a FEPROM update:

1. The system manager should perform operating system shutdown.
2. Enter console mode by pressing the Halt button once to halt the system. If the Break Enable/Disable switch on the console module is set to enable (indicated by 1), you can halt the system by pressing the Break key on the console terminal.

Figure 6–2 W4 Jumper Setting for Updating Firmware



6.2 Updating Firmware via Ethernet

To update firmware via the Ethernet, the "client" system (the target system to be updated) and the "server" system (the system that serves boot requests) must be on the same Ethernet segment. The Maintenance Operation Protocol (MOP) is the transport used to copy the network image.

Use the following procedure to update firmware via the Ethernet:

1. Enable the server system's NCP circuit using the following OpenVMS commands:

```
$ MCR NCP
NCP>SET CIRCUIT <circuit> STATE OFF
NCP>SET CIRCUIT <circuit> SERVICE ENABLED
NCP>SET CIRCUIT <circuit> STATE ON
```

Where <circuit> is the system Ethernet circuit. Use the SHOW KNOWN CIRCUITS command to find the name of the circuit.

Note

The SET CIRCUIT STATE OFF command will bring down the system's network.

2. Copy the file containing the updated code to the MOM$LOAD area on the server (this procedure may require system privileges). Refer to the Firmware Update Utility Release Notes for the Ethernet bootable filename. Use the following command to copy the file:

   $ COPY <filename>.SYS MOM$LOAD:*.*

   Where <filename> is the Ethernet bootable filename provided in the release notes.

3. On the client system, enter the command BOOT/100 EZ at the console prompt (>>>).

   The system then prompts you for the name of the file.

Note

Do NOT type the ".SYS" suffix when entering the Ethernet bootfile name. The MOP load protocol only supports 15 character filenames.

4. After the FEPROM upgrade program is loaded, simply type "Y" at the prompt to start the FEPROM blast. Example 6–1 provides a console display of the FEPROM update program.

Caution

Once you enter the bootfile name, do not interrupt the FEPROM blasting program, as this can damage the CPU module. The program takes several minutes to complete.

Example 6–1 FEPROM Update via Ethernet

```
***** On Server System *****

$ MCR NCP
NCP>SET CIRCUIT ISA-0 STATE OFF
NCP>SET CIRCUIT ISA-0 SERVICE ENABLED
NCP>SET CIRCUIT ISA-0 STATE ON
NCP>EXIT
$
$ COPY KA50_V41_EE.SYS MOM$LOAD:*.*
$

***** On Client System *****

>>>b/100 eza0
(BOOT/R5:100 EZA0)

  2..

Bootfile: ka50_v12

-EZA0
  1..0..

 FEPROM update program

                        ---CAUTION---

 --- Executing this program will change your current FEPROM ---

Do you want to continue [Y/N] ? : y

Blasting in V1.2-41.   The program will take at most several minutes.

DO NOT ATTEMPT TO INTERRUPT PROGRAM EXECUTION
Doing so may result in loss of operable state !!!

+--------------------------------------+
10...9...8...7...6...5...4...3...2...1...0

FEPROM Programming successful

?06 HLT INST
        PC = 00008E24
>>>
```

Note

If the update does not work, check to be sure the "write enable" on-board jumper is installed (see Figure 6–2).

5. Recycle power or enter "T 0" at the console prompt (>>>).
6. If the customer requires, return the jumper on the module to the "write disable mode" setting.

6.3 Updating Firmware via Tape

To update firmware via tape, the system must have a TZ30, TF85, TK70, TK50 or TLZ04 tape drive.

If you need to make a bootable tape, copy the bootable image file to a tape as shown in the following example. Refer to the release notes for the name of the file.

```
$ INIT MKA500:"VOLUME_NAME"
$ MOUNT/BLOCK_SIZE = 512  MKA500:"VOLUME_NAME"
$ COPY/CONTIG <file_name> MkA500:<file_name>
$ DISMOUNT MKA500
$
```

Use the following procedure to update firmware via tape:

1. Be sure the on board jumper is in the correct ("write enable mode") position (Section 6.1).
2. At the console prompt (>>>), enter the BOOT/100 command for the tape device, for example: BOOT/100 MKA500.

   Use the SHOW DEVICE command if you are not sure of the device name for the tape drive.

   The system prompts you for the name of the file. Enter the bootfile name.
3. After the FEPROM upgrade program is loaded, simply type "Y" at the prompt to start the FEPROM blast. Example 6–2 provides a console display of the FEPROM update program.

   Caution

   Once you enter the bootfile name, do not interrupt the FEPROM blasting program, as this can damage the CPU module. The program takes several minutes to complete.

4. Press the Restart button on the SCP or enter "T 0" at the console prompt (>>>).
5. If the customer requires, return the jumper on the CPU module to the "write disable mode" setting.

Example 6–2 FEPROM Update via Tape

```
>>> BOOT/100 MKA500
(BOOT/R5:100 MKA500)

  2..
Bootfile: KA50_V41_BZ
-MKA500

  1..0..

 FEPROM update program

                         ---CAUTION---

 --- Executing this program will change your current FEPROM ---

Do you want to continue [Y/N] ? : y

Blasting in V1.2-41.   The program will take at most several minutes.

DO NOT ATTEMPT TO INTERRUPT PROGRAM EXECUTION
Doing so may result in loss of operable state !!!

+--------------------------------------+
10...9...8...7...6...5...4...3...2...1...0

FEPROM Programming successful

?06 HLT INST
        PC = 00008E24
>>>
```

6.4 FEPROM Update Error Messages

The following is a list of error messages generated by the FEPROM update program and actions to take if the errors occur.

MESSAGE:
?? ERROR update enable jumper is disconnected
unable to blast ROMs...
ACTION:
Reposition update enable jumper (Section 6.1).

MESSAGE:
?? ERROR, FEPROM programming failed
ACTION:
Turn off the system, then turn it on. If you see the banner message as expected, reenter console mode and try booting the update program again. If you do not see the usual banner message, replace the CPU module.

Patchable Control Store (PCS) Loading Error Messages

The following is a list of error messages that may appear if there is a problem with the PCS. The PCS is loaded as part of the power-up stream (before ROM-based diagnostics are executed).

MESSAGE:
CPU is not an NVAX
COMMENT:
CPU_TYPE as read in NVAX SID is not = 19 (decimal), as is should be for an NVAX processor.

MESSAGE:
Microcode patch/CPU rev mismatch
COMMENT:
Header in microcode patch does not match MICROCODE_REV as read in NVAX SID.

MESSAGE:
PCS Diagnostic failed
COMMENT:
Something is wrong with the PCS. Replace the NVAX chip (or CPU module).

A

Address Assignments

Note

The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95 and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, will appear on the console and/or printouts from time to time.

A.1 KA50/51/55/56 General Local Address Space Map

Address Assignments

A.1 KA50/51/55/56 General Local Address Space Map

VAX Memory Space

Address Range	Contents
0000 0000 - 1FFF FFFF	Local Memory Space (512MB)

VAX I/O Space

Address Range	Contents
2000 0000 - 2000 1FFF	Local Q22-Bus I/O Space (8KB)
2000 2000 - 2003 FFFF	Reserved Local I/O Space (248KB)
2008 0000 - 201F FFFF	Local Register I/O Space (1.5MB)
2020 0000 - 23FF FFFF	Reserved Local I/O Space (62.5MB)
2400 0000 - 27FF FFFF	Reserved Local I/O Space (64MB)
2008 0000 - 2BFF FFFF	Reserved Local I/O Space (64MB)
2C08 0000 - 2FFF FFFF	Reserved Local I/O Space (64MB)
3000 0000 - 303F FFFF	Local Q22-Bus Memory Space (4MB)
3040 0000 - 33FF FFFF	Reserved Local I/O Space (60MB)
3400 0000 - 37FF FFFF	Reserved Local I/O Space (64MB)
3800 0000 - 3BFF FFFF	Reserved Local I/O Space (64MB)
3C00 0000 - 3FFF FFFF	Reserved Local I/O Space (64MB)
E004 0000 - E007 FFFF	Local ROM Space

A.2 KA50/51/55/56 Detailed Local Address Space Map

Local Memory Space (up to 128MB)	0000 0000 - 7FF FFFF
Q22-bus Map - top 32KB of Main Memory	
VAX I/O Space	
Local Q22-bus I/O Space	2000 0000 - 2000 1FFF
Reserved Q22-bus I/O Space	2000 0000 - 2000 0007
Q22-bus Floating Address Space	2000 0008 - 2000 07FF
User Reserved Q22-bus I/O Space	2000 0800 - 2000 0FFF
Reserved Q22-bus I/O Space	2000 1000 - 2000 1F3F
Interprocessor Comm Reg	2000 1F40
Reserved Q22-bus I/O Space	2000 1F44 - 2000 1FFF
Local Register I/O Space	2000 2000 - 2003 FFFF
Reserved Local Register I/O Space	2000 4000 - 2000 422F
Reserved Local Register I/O Space	2000 42B0 - 2000 7FFF
Reserved Local Register I/O Space	2000 40B0 - 2000 422F
NICSR0 - Vector Add, IPL, Sync/Async	2000 8000
NICSR1 - Polling Demand Register	2000 8004
NICSR2 - Reserved	2000 8008
NICSR3 - Receiver List Address	2000 800C
NICSR4 - Transmitter List Address	2000 8010
NICSR5 - Status Register	2000 8014
NICSR6 - Command and Mode Register	2000 8018
NICSR7 - System Base Address	2000 801C
NICSR8 - Reserved	2000 8020*
NICSR9 - Watchdog Timers	2000 8024*
NICSR10- Reserved	2000 8028*
NICSR11- Rev Num & Missed Frame Count	2000 802C*
NICSR12- Reserved	2000 8030*
NICSR13- Breakpoint Address	2000 8034*
NICSR14- Reserved	2000 8038*
NICSR15- Diagnostic Mode & Status	2000 803C
Reserved Local Register I/O Space	2000 8040 - 2003 FFFF

A.2 KA50/51/55/56 Detailed Local Address Space Map

KA50/51/55/56 DETAILED LOCAL ADDRESS SPACE MAP (Cont.)

```
**********************************************************************
*
*  Q-22 Bus Local Register I/O Space          2008 0000 - 201F FFFF
*     DMA System Configuration Register       2008 0000
*     DMA System Error Register               2008 0004
*     DMA Master Error Address Register       2008 0008
*     DMA Slave Error Address Register        2008 000C
*     Q22-bus Map Base Register               2008 0010
*     Reserved Local Register I/O Space       2008 0014 - 2008 00FF
*
**********************************************************************

    Reserved Local Register I/O Space         2008 0194 - 2008 3FFF

    Boot and Diagnostic Reg (32 Copies)       2008 4000 - 2008 407C

    Reserved Local Register I/O Space         2008 4080 - 2008 7FFF

**********************************************************************
*
*   Q22-bus Map Registers                     2008 8000 - 2008 FFFF
*   Reserved Local Register I/O Space         2009 0000 - 2013 FFFF
*
**********************************************************************

SSC CSRs

    SSC Base Address Register                 2014 0000
    SSC Configuration Register                2014 0010
    CP Bus Timeout Control Register           2014 0020
    Diagnostic LED Register                   2014 0030
    Reserved Local Register I/O Space         2014 0034 - 2014 006B
```

KA50/51/55/56 DETAILED LOCAL ADDRESS SPACE MAP (Cont.)

VAX IPRs implemented by NCA	
Interval Clock Control Status Reg	2100 0060
Next Interval Count Register	2100 0064
Interval Count Register	2100 0068
NMC CSRs	
O-bit Data Registers	2101 0000 - 2101 7FFF
Main Memory Configuration Reg 0	2101 8000
Main Memory Configuration Reg 1	2101 8004
Main Memory Signature Register 0	2101 8020
Main Memory Signature Register 1	2101 8024
Main Memory Error Address Register	2101 8040
Main Memory Error Status Register	2101 8044
Main Memory Mode Control and Diagnostic Register	2101 8048
O-bit Address and Mode Register	2101 804C
NCA CSRs	
Error Status Register	2102 0000
Mode Control and Diagnostic Reg	2102 0004
CP1 Slave Error Address Register	2102 0006
CP2 Slave Error Address Register	2102 000C
CP1 IO Error Address Register	2102 0010
CP2 IO Error Address Register	2102 0014
NDAL Error Address Register	2102 0018

```
KA50/51/55/56 DETAILED LOCAL ADDRESS SPACE MAP (Cont.)

*****************************************************************
* OPTIONAL KZDDA SCSI CONTROLLER
*
*   SCSI  DMA address register                21C00000
*   SCSI  DMA direction register              21C00004
*   Interrupt mask register                   21C00008
*   Interrupt pending register                21C0000C
*   SCSI Controller (53C94) registers         22000080 - 220000B0
*      (13 byte regs (0:9,A,B,C) on LW boundary)
*                              scsicsr0       22000080
*                              scsicsr1       22000084
*                              scsicsr2       22000088
*                              scsicsr3       2200008C
*                              scsicsr4       22000090
*                              scsicsr5       22000094
*                              scsicsr6       22000098
*                              scsicsr7       2200009C
*                              scsicsr8       220000A0
*                              scsicsr9       220000A4
*                              scsicsra       220000A8
*                              scsicsrb       220000AC
*                              scsicsrc       220000B0
*   SCSI DMA Map registers                    23000000 - 23007FFF
*   (8,192 32 bit registers)
*****************************************************************

EDAL BUS DEVICES

*****************************************************************
* OPTIONAL SYNC COMMUNICATION DEVICE
*
*   Register sets of the SYNC ports       2400 0000 - 24FF FFFF
*   Option ROM Space          ???? ???? - ???? ????
*
*****************************************************************
    QUART (DC7085) Registers          2500 0000 - 2500 0007
    SCSI DMA Address Register        25C0 0000
    SCSI DMA Direction Register       25C0 0004
    Interrupt Mask Register         25C0 0008
    Interrupt Pending Register        25C0 000C
    SCSI Controller (53C94) Registers         2600 0080 - 2600 00BF
    SCSI DMA Map Registers         2700 0000 - 2700 7FFF
```

```
KA50/51/55/56 DETAILED LOCAL ADDRESS SPACE MAP (Cont.)
****************************************************************
* OPTIONAL ASYNC COMMUNICATION DEVICE
*
*   Register sets of the ASYNC ports       3E00 0000 - 3E00 000E
*   Option ROM Space          3E01 0000 - 3E02 FFFF
*
****************************************************************

Local FEPROM Space                                 E004 0000 - E007 FFFF
    VAX System Type Register (In ROM)              E004 0004
    Local FEPROM - (Halt Protected)                E004 0000 - E007 FFFF

************************************************************************
The following addresses allow those KA50/51/55/56 Internal Processor
Registers that are implemented in the SSC chip (External, Internal
Processor Registers) to be accessed via the local I/O page. These
addresses are documented for diagnostic purposes only and should
not be used by non-diagnostic programs.

    Time Of Year Register                          2014 006C

    Console Storage Receiver Status                2014 0070*
    Console Storage Receiver Data                  2014 0074*
    Console Storage Transmitter Status             2014 0078*
    Console Storage Transmitter Data               2014 007C*
    Console Receiver Control/Status                2014 0080
    Console Receiver Data Buffer                   2014 0084
    Console Transmitter Control/Status             2014 0088
    Console Transmitter Data Buffer                2014 008C
    Reserved Local Register I/O Space              2014 0090 - 2014 00DB

    I/O Bus Reset Register                         2014 00DC
    Reserved Local Register I/O Space              2014 00E0

    Reserved Local Register I/O Space              2014 00FC - 2014 00FF

  * These registers are not fully implemented, accesses yield
    UNPREDICTABLE results.

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

KA50/51/55/56 DETAILED LOCAL ADDRESS SPACE MAP (Cont.)

Local Register I/O Space (Cont.)	
Timer 0 Control Register	2014 0100
Timer 0 Interval Register	2014 0104
Timer 0 Next Interval Register	2014 0108
Timer 0 Interrupt Vector	2014 010C
Timer 1 Control Register	2014 0110
Timer 1 Interval Register	2014 0114
Timer 1 Next Interval Register	2014 0118
Timer 1 Interrupt Vector	2014 011C
Reserved Local Register I/O Space	2014 0120 - 2014 012F
BDR Address Decode Match Register	2014 0140
BDR Address Decode Mask Register	2014 0144
Reserved Local Register I/O Space	2014 0138 - 2014 03FF
Battery Backed-Up RAM	2014 0400 - 2014 07FF
Reserved Local Register I/O Space	2014 0800 - 201F FFFF
Reserved Local I/O Space	2020 0000 - 2FFF FFFF
Local Q22-bus Memory Space	3000 0000 - 303F FFFF
Reserved Local Register I/O Space	3040 0000 - 3FFF FFFF

A.3 External, Internal Processor Registers

Several of the Internal Processor Registers (IPR's) on the KA50/51/55/56 are implemented in the NCA or SSC chip rather than the CPU chip. These registers are referred to as External Internal Processor Registers and are listed below.

IPR #	Register Name	Abbrev.
27	Time of Year Register	TOY
28	Console Storage Receiver Status	CSRS*
29	Console Storage Receiver Data	CSRD*
30	Console Storage Transmitter Status	CSTS*
31	Console Storage Transmitter Data	CSDB*
32	Console Receiver Control/Status	RXCS
33	Console Receiver Data Buffer	RXDB
34	Console Transmitter Control/Status	TXCS
35	Console Transmitter Data Buffer	TXDB
55	I/O System Reset Register	IORESET

* These registers are not fully implemented, accesses yield UNPREDICTABLE results.

A.4 Global Q22–bus Address Space Map

```
Q22-bus Memory Space
--------------------

Q22-bus Memory Space  (Octal)                  0000 0000 - 1777 7777

Q22-bus I/O Space  (BBS7 Asserted)
----------------------------------

Q22-bus I/O Space (Octal)                      1776 0000 - 1777 7777
    Reserved Q22-bus I/O Space                 1776 0000 - 1776 0007
    Q22-bus Floating Address Space             1776 0010 - 1776 3777
    User Reserved Q22-bus I/O Space            1776 4000 - 1776 7777
    Reserved Q22-bus I/O Space                 1777 0000 - 1777 7477
    Interprocessor Comm Reg                    1777 7500
    Reserved Q22-bus I/O Space                 1777 7502 - 1777 7777
```

A.5 Processor Registers

Table A–1 Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Kernel Stack Pointer	KSP	0	0	RW	NVAX	1-1	
Executive Stack Pointer	ESP	1	1	RW	NVAX	1-1	
Supervisor Stack Pointer	SSP	2	2	RW	NVAX	1-1	
User Stack Pointer	USP	3	3	RW	NVAX	1-1	
Interrupt Stack Pointer	ISP	4	4	RW	NVAX	1-1	
Reserved		5–7	5			3	E1000014

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
P0 Base Register	P0BR	8	8	RW	NVAX	1-2	
P0 Length Register	P0LR	9	9	RW	NVAX	1-2	
P1 Base Register	P1BR	10	A	RW	NVAX	1-2	
P1 Length Register	P1LR	11	B	RW	NVAX	1-2	
System Base Register	SBR	12	C	RW	NVAX	1-2	
System Length Register	SLR	13	D	RW	NVAX	1-2	
CPU Identification	CPUID	14	E	RW	NVAX	2-1	
Reserved		15	F			3	E100003C
Process Control Block Base	PCBB	16	10	RW	NVAX	1-1	
System Control Block Base	SCBB	17	11	RW	NVAX	1-1	
Interrupt Priority Level[1]	IPL	18	12	RW	NVAX	1-1	
AST Level[1]	ASTLVL	19	13	RW	NVAX	1-1	
Software Interrupt Request Register	SIRR	20	14	W	NVAX	1-1	

[1] Initialized on reset

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Software Interrupt Summary Register[1]	SISR	21	15	RW	NVAX	1-1	
Reserved		22–23	16			3	E1000058
Interval Counter Control /Status	ICCS	24	18	RW	NCA	2-7	E1000060
Next Interval Count	NICR	25	19	RW	NCA	3-7	E1000064
Interval Count	ICR	26	1A	RW	NCA	3-7	E1000068
Time of Year Register	TODR	27	1B	RW	SSC	2-3	E100006C
Console Storage Receiver Status	CSRS	28	1C	RW	SSC	2-3	E1000070
Console Storage Receiver Data	CSRD	29	1D	R	SSC	2-3	E1000074
Console Storage Transmitter Status	CSTS	30	1E	RW	SSC	2-3	E1000078
Console Storage Transmitter Data	CSTD	31	1F	W	SSC	2-3	E100007C

[1]Initialized on reset

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Console Receiver Control /Status	RXCS	32	20	RW	SSC	2-3	E1000080
Console Receiver Data Buffer	RXDB	33	21	R	SSC	2-3	E1000084
Console Transmitter Control /Status	TXCS	34	22	RW	SSC	2-3	E1000088
Console Transmitter Data Buffer	TXDB	35	23	W	SSC	2-3	E100008C
Reserved		36	24			3	E1000090
Reserved		37	25			3	E1000094
Machine Check Error Register	MCESR	38	26	W	NVAX	2-1	
Reserved		39	27			3	E100009C
Reserved		40	28			3	E10000A0
Reserved		41	29			3	E10000A4
Console Saved PC	SAVPC	42	2A	R	NVAX	2-1	
Console Saved PSL	SAVPSL	43	2B	R	NVAX	2-1	
Reserved		44–54	2C			3	E10000B0
I/O System Reset Register	IORESET	55	37	W	SSC	2-3	E10000DC

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Memory Management Enable[1,2]	MAPEN	56	38	RW	NVAX	1-2	
Translation Buffer Invalidate All[2]	TBIA	57	39	W	NVAX	1-1	
Translation Buffer Invalidate Single[2]	TBIS	58	3A	W	NVAX	1-1	
Reserved		59	3B			3	E10000EC
Reserved		60	3C			3	E10000F0
System Identification	SID	62	3E	R	NVAX	1-1	
Translation Buffer Check	TBCHK	63	3F	W	NVAX	1-1	
IPL 14 Interrupt ACK[3]	IAK14	64	40	R	SSC	2-3	E1000100
IPL 15 Interrupt ACK[3]	IAK15	65	41	R	SSC	2-3	E1000104
IPL 16 Interrupt ACK[3]	IAK16	66	42	R	SSC	2-3	E1000108
IPL 17 Interrupt ACK[3]	IAK17	67	43	R	SSC	2-3	E100010C
Clear Write Buffer[3]	CWB	68	44	RW	SSC	2-3	E1000110

[1]Initialized on reset

[2]Change broadcast to vector unit if present

[3]Testability and diagnostic use only; not for software use in normal operation

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Reserved		69–99	45			3	E1000114
Reserved for VM		100	64			3	E1000190
Reserved for VM		101	65			3	E1000194
Reserved for VM		102	66			3	E1000198
Reserved		103–121	67			3	E100019C
Interrupt System Status Register	INTSYS	122	7A	RW	NVAX	2-1	
Performance Monitoring Facility Count	PMFCNT	123	7B	RW	NVAX	2-1	
Patchable Control Store Control Register	PCSCR	124	7C	WO	NVAX	2-1	
Ebox Control Register	ECR	125	7D	RW	NVAX	2-1	
Mbox TB Tag Fill[3]	MTBTAG	126	7E	W	NVAX	2-1	
Mbox TB PTE Fill[3]	MTBPTE	127	7F	W	NVAX	2-1	
Cbox Control Register	CCTL	160	A0	RW	NVAX	2-5	

[3]Testability and diagnostic use only; not for software use in normal operation

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Reserved		161	A1		NVAX	2-6	
Bcache Data ECC	BCDECC	162	A2	W	NVAX	2-5	
Bcache Error Tag Status	BCETSTS	163	A3	RW	NVAX	2-5	
Bcache Error Tag Index	BCETIDX	164	A4	R	NVAX	2-5	
Bcache Error Tag	BCETAG	165	A5	R	NVAX	2-5	
Bcache Error Data Status	BCEDSTS	166	A6	RW	NVAX	2-5	
Bcache Error Data Index	BCEDIDX	167	A7	R	NVAX	2-5	
Bcache Error ECC	BCEDECC	168	A8	R	NVAX	2-5	
Reserved		169	A9		NVAX	2-6	
Reserved		170	AA		NVAX	2-6	
Fill Error Address	CEFADR	171	AB	R	NVAX	2-5	
Fill Error Status	CEFSTS	172	AC	RW	NVAX	2-5	
Reserved		173	AD		NVAX	2-6	
NDAL Error Status	NESTS	174	AE	RW	NVAX	2-5	
Reserved		175	AF		NVAX	2-6	

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
NDAL Error Output Address	NEOADR	176	B0	R	NVAX	2-5	
Reserved		177	B1		NVAX	2-6	
NDAL Error Output Command	NEOCMD	178	B2	R	NVAX	2-5	
Reserved		179	B3		NVAX	2-6	
NDAL Error Data High	NEDATHI	180	B4	R	NVAX	2-5	
Reserved		181	B5		NVAX	2-6	
NDAL Error Data Low	NEDATLO	182	B6	R	NVAX	2-5	
Reserved		183	B7		NVAX	2-6	
NDAL Error Input Command	NEICMD	184	B8	R	NVAX	2-5	
Reserved		185–207	B9		NVAX	2-6	
VIC Memory Address Register	VMAR	208	D0	RW	NVAX	2-5	
VIC Tag Register	VTAG	209	D1	RW	NVAX	2-5	
VIC Data Register	VDATA	210	D2	RW	NVAX	2-5	

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Ibox Control and Status Register	ICSR	211	D3	RW	NVAX	2-5	
Ibox Branch Prediction Control Register[3]	BPCR	212	D4	RW	NVAX	2-5	
Reserved		213	D5		NVAX	2-6	
Ibox Backup PC[3]	BPC	214	D6	R	NVAX	2-5	
Ibox Backup PC with RLOG Unwind[3]	BPCUNW	215	D7	R	NVAX	2-5	
Reserved		216–223	D8		NVAX	2-6	
Mbox P0 Base Register[3]	MP0BR	224	E0	RW	NVAX	2-5	
Mbox P0 Length Register[3]	MP0LR	225	E1	RW	NVAX	2-5	
Mbox P1 Base Register[3]	MP1BR	226	E2	RW	NVAX	2-5	
Mbox P1 Length Register[3]	MP1LR	227	E3	RW	NVAX	2-5	
Mbox System Base Register[3]	MSBR	228	E4	RW	NVAX	2-5	

[3]Testability and diagnostic use only; not for software use in normal operation

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Mbox System Length Register[3]	MSLR	229	E5	RW	NVAX	2-5	
Mbox Memory Management Enable[3]	MMAPEN	230	E6	RW	NVAX	2-5	
Mbox Physical Address Mode	PAMODE	231	E7	RW	NVAX	2-5	
Mbox MME Address	MMEADR	232	E8	R	NVAX	2-5	
Mbox MME PTE Address	MMEPTE	233	E9	R	NVAX	2-5	
Mbox MME Status	MMESTS	234	EA	R	NVAX	2-5	
Reserved		235	EB		NVAX	2-6	
Mbox TB Parity Address	TBADR	236	EC	R	NVAX	2-5	
Mbox TB Parity Status	TBSTS	237	ED	RW	NVAX	2-5	
Reserved		238	EE		NVAX	2-6	
Reserved		239	EF		NVAX	2-6	
Reserved		240	F0		NVAX	2-6	
Reserved		241	F1		NVAX	2-6	

[3]Testability and diagnostic use only; not for software use in normal operation

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Mbox Pcache Parity Address	PCADR	242	F2	R	NVAX	2-5	
Reserved		243	F3		NVAX	2-6	
Mbox Pcache Status	PCSTS	241	F4	RW	NVAX	2-5	
Reserved		245	F5		NVAX	2-6	
Reserved		246	F6		NVAX	2-6	
Reserved		247	F7		NVAX	2-6	
Mbox Pcache Control	PCCTL	248	F8	RW	NVAX	2-5	
Reserved		249	F9		NVAX	2-6	
Reserved		250	FA		NVAX	2-6	
Reserved		251	FB		NVAX	2-6	
Reserved		252	FC		NVAX	2-6	
Reserved		253	FD		NVAX	2-6	
Reserved		254	FE		NVAX	2-6	
Reserved		255	FF		NVAX	2-6	
Unimplemented			100–00FFFFFF			3	

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
See Table A–2			01000000–FFFFFFFF			2	

Type:

R = Read-only register
RW = Read-write register
W = Write-only register

Impl(emented):

NVAX = Implemented in the NVAX CPU chip
System = Implemented in the system environment
Vector = Implemented in the optional vector unit or its NDAL interface

Cat(egory), *class-subclass*, where:
class is one of:

1 = Implemented as per DEC standard 032
2 = NVAX-specific implementation which is unique or different from the DEC standard 032 implementation
3 = Not implemented internally; converted to I/O space read or write and passed to system environment

subclass is one of:

1 = Processed as appropriate by Ebox microcode
2 = Converted to Mbox IPR number and processed via internal IPR command
3 = Processed by internal IPR command, then converted to I/O space read or write and passed to system environment
4 = If virtual machine option is implemented, processed as in 1, otherwise as in 3
5 = Processed by internal IPR command
6 = May be block decoded; reference causes UNDEFINED behavior
7 = Full interval timer may be implemented in the system environment. Subset ICCS is implemented in NVAX CPU chip
8 = Converted to MFVP MSYNC

A.6 IPR Address Space Decoding

Table A–2 IPR Address Space Decoding

IPR Group	Mnemonic[2]	IPR Address Range (hex)	Contents
Normal		00000000..000000FF[1]	256 individual IPRs.
Bcache Tag	BCTAG	01000000..011FFFE0[1]	64k Bcache tag IPRs, each separated by 20(hex) from the previous one.
Bcache Deallocate	BCFLUSH	01400000..015FFFE0[1]	64k Bcache tag deallocate IPRs, each separated by 20(hex) from the previous one.
Pcache Tag	PCTAG	01800000..01801FE0[1]	256 Pcache tag IPRs, 128 for each Pcache set, each separated by 20(hex) from the previous one.
Pcache Data Parity	PCDAP	01C00000..01C01FF8[1]	1024 Pcache data parity IPRs, 512 for each Pcache set, each separated by 8(hex) from the previous one.

[1]Unused fields in the IPR addresses for these groups should be zero. Neither hardware nor microcode detects and faults on an address in which these bits are nonzero. Although noncontiguous address ranges are shown for these groups, the entire IPR address space maps into one of these groups. If these fields are nonzero, the operation of the CPU is UNDEFINED.

[2]The mnemonic is for the first IPR in the block.

Processor registers in all groups except the normal group are processed entirely by the NVAX CPU chip and will never appear on the NDAL. This is also true for a number of the IPRs in the normal group. IPRs in the normal group that are not processed by the NVAX CPU chip are converted into I/O space references and passed to the system environment via a read or write command on the NDAL.

Each of the 256 possible IPRs in the normal group are of longword length, so a 1-KB block of I/O space is required to convert each possible IPR to a unique I/O space longword. This block starts at address E1000000 (hex). Conversion of an IPR address to an I/O space address in this block is done by shifting the IPR address left into bits <9:2>, filling bits <1:0> with zeros, and merging in the base address of the block. This can be expressed by the equation:

$$IO\ ADDRESS = E1000000 + (IPR\ NUMBER * 4)$$

B

ROM Partitioning

Note

The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95, and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, will appear on the console and/or printouts from time to time.

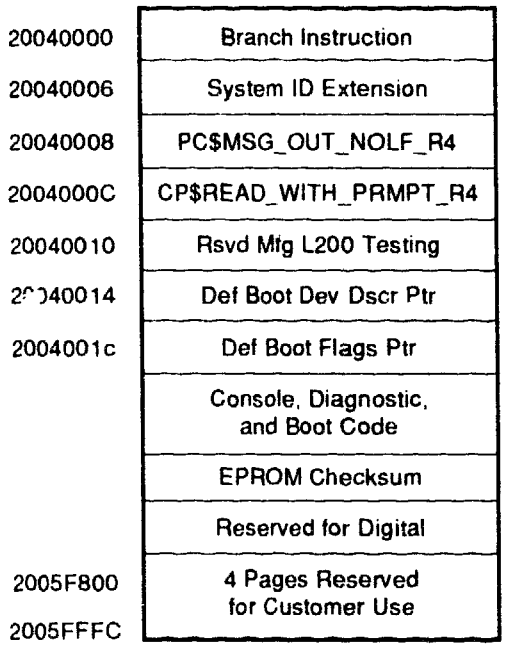
This section describes ROM partitioning and subroutine entry points that are public and are guaranteed to be compatible over future versions of the firmware. An entry point is the address at which any subroutine or subprogram will start execution.

B.1 Firmware EPROM Layout

The KA50/51/55/56 has 512 Kbytes of FEPROM. Unlike previous Q22–bus based processors, there is no duplicate decoding of the FEPROM into halt-protected and halt-unprotected spaces. The entire FEPROM is halt-protected. See Figure B–1 for the KA50/51/55/56 FEPROM layout.

ROM Partitioning
B.1 Firmware EPROM Layout

Figure B–1 KA50/51/55/56 FEPROM Layout



MLO-007698

The first instruction executed on halts is a branch around the System ID Extension (SIE) and the callback entry points. This allows these public data structures to reside in fixed locations in the FEPROM.

The callback area entry points provide a simple interface to the currently defined console for VMB and secondary bootstraps. This is documented further in the next section.

The fixed area checksum is the sum of longwords from 20040000 to the checksum, inclusive. This checksum is distinct from the checksum that the rest of the console uses.

The console, diagnostic and boot code constitute the bulk of the firmware. This code is field upgradable. The console checksum is from 20044000 to the checksum, inclusive.

The memory between the console checksum and the user area at the end of the FEPROM is reserved for Digital for future expansion of the firmware. The contents of this area is set to FF.

The last 4096 bytes of FEPROM are reserved for customer use and are not included in the console checksum. During a PROM bootstrap with PRB0 as the selected boot device, this block is tested for a PROM "signature block".

B.1.1 System Identification Registers

The firmware and operating system software reference two registers to determine the processor on which they are running. The first, the System Identification register (SID), is a NVAX internal processor register. The second, the System Identification Extension register (SIE), is a firmware register located in the FEPROM.

B.1.1.1 PR$_SID (IPR 62)

The SID longword can be read from IPR 62 using the MFPR instruction. This longword value is processor specific, however, the layout of this register is shown in Figure B–2. A description of each field is provided in Table B–1.

Figure B–2 SID : System Identification Register

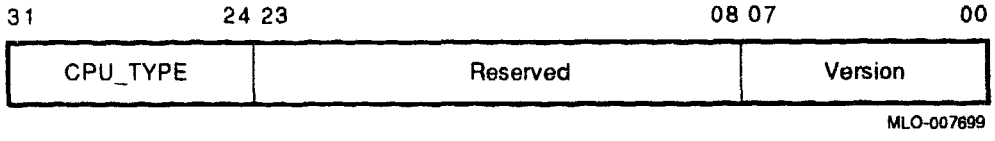


Table B–1 System Identification Register

Field	Name	RW	Description
31:24	CPU_TYPE	ro	CPU type is the processor specific identification code. *0A : CVAX* *0B : RIGEL* **13 : NVAX** *14 : SOC*
24:8	Reserved	ro	Reserved for future use.
7:0	VERSION	ro	Version of the microcode.

B.1.1.2 SIE (20040004)

The System Identification Extension register is an extension of the SID and is used to further differentiate between hardware configurations. The SID identifies which CPU and microcode are executing, and the SIE identifies which module and firmware revision are present. Note, the fields in this register are dependent on SID<31:24>(CPU_TYPE).

ROM Partitioning
B.1 Firmware EPROM Layout

By convention, all MicroVAX 3100 systems implement a longword at physical location 20040004 in the firmware FEPROM for the SIE. The layout of the SIE is shown in Figure B–3. A description of each field is provided in Table B–2.

Figure B–3 SIE : System Identification Extension (20040004)

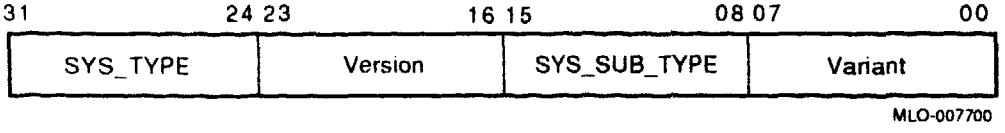


Table B–2 System Identification Extension

Field	Name	RW	Description
31:24	SYS_TYPE	ro	This field identifies the type of system for a specific processor. *03 : Bounded system.*
23:16	VERSION	ro	This field indentifies the resident version of the firmware encoded as two hexadecimal digits. For example, if the banner displays V5.0, then this field is 50 (hex).
15:8	SYS_SUB_ TYPE	ro	This field indentifies the particular system subtype. *08 : KA50/KA55* *09 : KA51/KA56* *0A : KA52* *0B : KA53*
7:0	VARIANT	ro	This field indentifies the particular system variant.

B.1.2 Call-Back Entry Points

The firmware provides several entry points that facilitate I/O to the designated console device. Users of these entry points do not need to be aware of the console device type, be it a video terminal or workstation.

The primary intent of these routines is to provide a simple console device to VMB and secondary bootstraps, before operating systems load their own terminal drivers.

These are JSB (subroutine as opposed to procedure) entry points located in fixed locations in the firmware. These locations branch to code that in turn calls the appropriate routines.

All of the entry points are designed to run at IPL 31 on the interrupt stack in physcial mode. Virtual mode is not supported. Due to internal firmware architectural restrictions, users are encouraged to only call into the halt-protected entry points. These entry points are listed in Table B–3.

Table B–3 Call-Back Entry Points

CP$GET_CHAR_R4	20040008
CP$MSG_OUT_NOLF_R4	2004000C
CP$READ_WTH_ PRMPT_R4	20040010

B.1.2.1 CP$GETCHAR_R4

This routine returns the next character entered by the operator in R0. A timeout interval can be specified. If the timeout interval is zero, no timeout is generated. If a timeout is specified and if timeout occurs, a value of 18 (CAN) is returned instead of normal input.

Registers R0,R1,R2,R3 and R4 are modified by this routine, all others are preserved.

```
;-------------------------------------------------------------
; Usage with timeout:

movl    #timeout_in_tenths_of_second,r0 ; Specify timeout.
jsb     @#CP$GET_CHAR_R4                ; Call routine.
cmpb    r0,#^x18                        ; Check for timeout.
beql    timeout_handler                 ; Branch if timeout.
; Input is in R0.

;-------------------------------------------------------------
; Usage without timeout:

clrl    r0                              ; Specify no timeout.
jsb     @#CP$GET_CHAR_R4                ; Call routine.
; Input is in R0.

;-------------------------------------------------------------
```

B.1.2.2 CP$MSG_OUT_NOLF_R4

This routine outputs a message to the console. The message is specified either by a message code or a string descriptor. The routine distinguishes between message codes and descriptors by requiring that any descriptor be located outside of the first page of memory. Hence, message codes are restricted to values between 0 and 511.

Registers R0,R1,R2,R3 and R4 are modified by this routine, all others are preserved.

```
;-------------------------------------------------------------
; Usage with message code:

movzbl  #console_message_code,r0        ; Specify message code.
jsb     @#CP$MSG_OUT_NOLF_R4            ; Call routine.

;-------------------------------------------------------------
; Usage with a message descriptor (position dependent).

movaq   5$,r0                           ; Specify address of desc.
jsb     @#CP$MSG_OUT_NOLF_R4            ; Call routine.
.
.
5$:     .ascid  /This is a message/     ; Message with descriptor.

;-------------------------------------------------------------
; Usage with a message descriptor (position independent).

pushab  5$                              ; Generate message desc.
pushl   #10$-5$                         ; on stack.
movl    sp,r0                           ; Pass desc. addr. in R0.
jsb     @#CP$MSG_OUT_NOLF_R4            ; Call routine.
clrq    (sp)+                           ; Purge desc. from stack.
.
.
5$:     .ascii  /This is a message/     ; Message.
10$:                                    ;

;-------------------------------------------------------------
```

B.1.2.3 CP$READ_WTH_PRMPT_R4

This routine outputs a prompt message and then inputs a character string from the console. When the input is accepted, DELETE, CONTROL-U and CONTROL-R functions are supported.

As with CP$MSG_OUT_NOLF_R4, either a message code or the address of a string descriptor is passed in R0 to specify the prompt string. A value of zero results in no prompt. A time-out value in 10-millisecond ticks may be passed in R1. If R1 is zero, the prompt will not timeout.

A descriptor of the input string is returned in R0 and R1. R0 contains the length of the string and R1 contains the address. This routine inputs the string into the console program string buffer and therefore the caller need not provide an input buffer. Successive calls however destroy the previous contents of the input buffer.

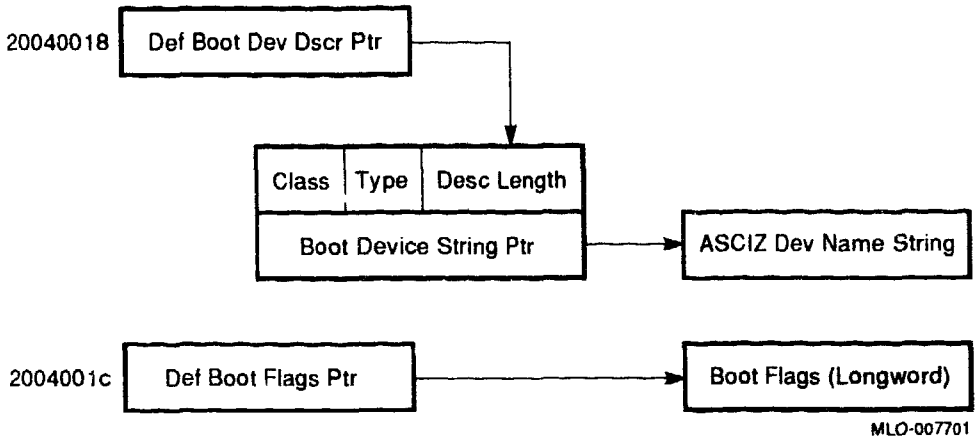
Registers R0 and R1 are modified by this routine, all others are preserved.

```
;-------------------------------------------------------------
; Usage with a message descriptor (position independent).

pushab  5$                              ; Generate prompt desc.
pushl   #10$-5$                         ; on stack.
movl    sp,r0                           ; Pass desc. addr. in R0.
clrl    r1                              ; Specify no time-out.
jsb     @#CP$READ_WTH_PRMPT_R4          ; Call routine.
clrq    (sp)+                           ; Purge prompt desc.
.                                       ; Input desc in R0 and R1.
.

5$:     .ascii  /Prompt> /              ; Prompt string.
10$:

;-------------------------------------------------------------
```

B.1.3 Boot Information Pointers

Two longwords located in FEPROM are used as pointers to the default boot device descriptor and the default boot flags (Figure B–4), because the actual location of this data may change in successive versions of the firmware. Any software that uses these pointers should reference them at the addresses in halt-protected space.

Figure B–4 Boot Information Pointers



The following macro defines the boot device descriptor format.

```
;-----------------------------------------------------------
; Default Boot Device Descriptor
;
boot_device_descriptor::
        base = .
        . = base + dsc$w_length
        .word   nvr$s_boot_device

        . = base + dsc$b_dtype
        .byte   dsc$k_dtype_z

        . = base + dsc$b_class
        .byte   dsc$k_class_z

        . = base + dsc$a_pointer
        .long   nvr_base + nvr$b_boot_device

        . = base + dsc$s_dscdef1
;-----------------------------------------------------------
```

C

Data Structures and Memory Layout

This appendix contains definitions of the key global data structures used by the CPU firmware.

Note

The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95 and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, will appear on the console and/or printouts from time to time.

C.1 Halt Dispatch State Machine

The CPU halt dispatcher determines what actions the firmware will take on halt entry based on the machine state. The dispatcher is implemented as a state machine, which uses a single bitmap control word and the transition (see Table C–1) to process all halts. The transition table is sequentially searched for matches with the current state and control word. If there is a match, a transition occurs to the next state.

The control word comprises the following information:

- **Halt Type,** used for resolving external halts. Valid only if Halt Code is 00.

  000 : power-up state
  001 : halt in progress
  010 : negation of Q22–bus DCOK
  011 : console BREAK condition detected
  100 : Q22–bus BHALT
  101 : SGEC BOOT_L asserted (trigger boot)

- **Halt Code,** compressed form of SAVPSL<13:8>(RESTART_CODE).

  00 : RESTART_CODE = 2, external halt
  01 : RESTART_CODE = 3, power-up/reset
  10 : RESTART_CODE = 6, halt instruction
  11 : RESTART_CODE = any other, error halts

- **Mailbox Action,** passed by an operating system in CPMBX<1:0>(HALT_ACTION).

  00 : restart, boot, halt
  01 : restart, halt
  10 : boot, halt
  11 : halt

- **User Action,** specified with the SET HALT console command.

  000 : default
  001 : restart, halt
  010 : boot, halt
  011 : halt
  100 : restart, boot, halt

- **HEN,** Break (halt) Enable/Disable switch, BDR<07>
- **ERR,** error status
- **TIP,** trace in progress
- **DIP,** diagnostics in progress
- **BIP,** bootstrap in progress CPMBX<2>
- **RIP,** restart in progress CPMBX<3>

A transition to a "next state" occurs if a match is found between the control word and a "current state" entry in the table. The firmware does a linear search through the table for a match. Therefore, the order of the entries in the transition table is important. The control longword is reassembled before each transition from the current machine state. The state machine transitions are shown in Table C--1.

Table C–1 Firmware State Transition Table

Current State	Next State	Halt Type	Halt Code	Mailbx Action	User Action	HEN-ERR-TIP-DIP-BIP-RIP
		Perform conditional initialization [1]				
ENTRY	->RESET INIT	xxx	01	xx	xxx	x - x - x - x - x - x
ENTRY	->BREAK INIT	011	00	xx	xxx	x - x - x - x - x - x
ENTRY	->TRACE INIT	xxx	10	xx	xxx	x - 0 - 1 - x - x - x
ENTRY	->OTHER INIT	xxx	xx	xx	xxx	x - x - x - x - x - x
		Perform common initialization [2]				
RESET INIT	->INIT	xxx	xx	xx	xxx	x - x - x - x - x - x
BREAK INIT	->INIT	xxx	xx	xx	xxx	x - x - x - x - x - x
TRACE INIT	->INIT	xxx	xx	xx	xxx	x - x - x - x - x - x
OTHER INIT	->INIT	xxx	xx	xx	xxx	x - x - x - x - x - x
		Check for external halts [3]				
INIT	->BOOTSTRAP	101	00	xx	xxx	x - x - x - x - x - x
INIT	->HALT	xxx	00	xx	xxx	x - x - x - x - x - x
		Check for pending (NEXT) trace [4]				
INIT	->TRACE	xxx	10	xx	xxx	x - x - 1 - x - x - x

[1] Perform a unique initialization routine on entry. In particular, power-ups, BREAKs, and TRACEs require special initialization. Any other halt entry performs a default initialization.

[2] After performing conditional initialization, complete common initialization.

[3] Halt on all external halts, except:

if DCOK (unlikely) and halts are disabled, bootstrap
if SGEC remote trigger, bootstrap

[4] Unconditionally enter the TRACE state, if the TIP flag is set and the halt was due to a HALT instruction. From the TRACE state the firmware exits, if TIP is set and ERR is clear; otherwise it halts.

(continued on next page)

Table C–1 (Cont.) Firmware State Transition Table

Current State	Next State	Halt Type	Halt Code	Mailbx Action	User Action	HEN-ERR-TIP-DIP-BIP-RIP
		Check for pending (NEXT) trace [4]				
TRACE	->EXIT	xxx	10	xx	xxx	x - 0 - 1 - x - x - x
TRACE	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
		Check for bootstrap conditions [5]				
INIT	->BOOTSTRAP	xxx	01	xx	xxx	0 - 0 - 0 - 0 - 0 - 0
INIT	->BOOTSTRAP	xxx	01	xx	010	1 - 0 - 0 - 0 - 0 - 0
INIT	->BOOTSTRAP	xxx	01	xx	100	1 - 0 - 0 - 0 - 0 - 0
INIT	->BOOTSTRAP	xxx	1x	10	xxx	x - 0 - 0 - 0 - 0 - 0
INIT	->BOOTSTRAP	xxx	1x	00	010	x - 0 - 0 - 0 - 0 - 0
INIT	->BOOTSTRAP	xxx	1x	00	100	x - 0 - 0 - 0 - 0 - 1
INIT	->BOOTSTRAP	xxx	1x	00	100	x - 1 - 0 - 0 - 0 - x
INIT	->BOOTSTRAP	xxx	1x	00	000	0 - 0 - 0 - 0 - 0 - 1
RESTART	->BOOTSTRAP	xxx	1x	00	000	0 - 1 - 0 - 0 - 0 - x
		Check for restart conditions [6]				
INIT	->RESTART	xxx	1x	01	xxx	x - 0 - 0 - 0 - 0 - 0
INIT	->RESTART	xxx	1x	00	001	x - 0 - 0 - 0 - 0 - 0
INIT	->RESTART	xxx	1x	00	100	x - 0 - 0 - 0 - 0 - 0

[4] Unconditionally enter the TRACE state, if the TIP flag is set and the halt was due to a HALT instruction. From the TRACE state the firmware exits, if TIP is set and ERR is clear; otherwise it halts.

[5] Bootstrap,

if power-up and halts are disabled.
if power-up and halts are enabled and user action is 2 or 4.
if not power-up and mailbox is 2.
if not power-up and mailbox is 0 and user action is 2.
if not power-up and restart failed and mailbox is 0 and user action is 0 or 4.

[6] Restart the operating system if not power-up and

if mailbox is 1.
if mailbox is 0 and user action is 1 or 4.
if mailbox is 0 and user action is 0 and halts are disabled.

(continued on next page)

Table C–1 (Cont.) Firmware State Transition Table

Current State	Next State	Halt Type	Halt Code	Mailbx Action	User Action	HEN-ERR-TIP-DIP-BIP-RIP
		Check for restart conditions [6]				
INIT	->RESTART	xxx	1x	00	000	0 - 0 - 0 - 0 - 0 - 0
		Perform common exit processing, if no errors [7]				
BOOTSTRAP	->EXIT	xxx	xx	xx	xxx	x - 0 - x - x - x - x
RESTART	->EXIT	xxx	xx	xx	xxx	x - 0 - x - x - x - x
HALT	->EXIT	xxx	xx	xx	xxx	x - 0 - x - x - x - x
		Exception transitions, just halt [8]				
INIT	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
BOOT	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
REST	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
HALT	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
TRACE	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
EXIT	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x

[6] Restart the operating system if not power-up and

if mailbox is 1.
if mailbox is 0 and user action is 1 or 4.
if mailbox is 0 and user action is 0 and halts are disabled.

[7] Exit after halts, bootstrap or restart. The exit state transitions to program I/O mode.

[8] Guard block that catches all exception conditions. In all cases, just halt.

C.2 Restart Parameter Block

VMB typically utilizes the low portion of memory unless there are bad pages in the first 128K bytes. The first page in its block is used for the Restart Parameter Block (RPB), through which it communicates to the operating system. Usually, this is page 0.

C.2 Restart Parameter Block

VMB will initialize the Restart Parameter Block (RPB) as shown in Table C–2.

Table C–2 Restart Parameter Block Fields

(R11)+	Field Name	Description
00:	RPB$L_BASE	Physical address of base of RPB.
04:	RPB$L_RESTART	Cleared.
08:	RPB$L_CHKSUM	-1
0C:	RPB$L_RSTRTFLG	Cleared.
10:	RPB$L_HALTPC	R10 on entry to VMB (HALT PC).
10:	RPB$L_HALTPSL	PR$_SAVPSL on entry to VMB (HALT PSL).
18:	RPB$L_HALTCODE	AP on entry to VMB (HALT CODE).
1C:	RPB$L_BOOTR0	R0 on entry to VMB.

Note

The field RPB$W_R0UBVEC, which overlaps the high-order word of RPB$L_BOOTR0, is set by the boot device drivers to the SCB offset (in the second page of the SCB) of the interrupt vector for the boot device.

20:	RPB$L_BOOTR1	VMB version number. The high-order word of the version is the major ID and the low-order word is the minor ID.
24:	RPB$L_BOOTR2	R2 on entry to VMB.
28:	RPB$L_BOOTR3	R3 on entry to VMB.
2C:	RPB$L_BOOTR4	R4 on entry to VMB.

Note

The 48-bit booting node address is stored in RPB$L_BOOTR3 and RPB$L_BOOTR4 for compatibility with ELN VX.X. (This field is only initialized this way when performing a network boot.)

30:	RPB$L_BOOTR5	R5 on entry to VMB.
34:	RPB$L_IOVEC	Physical address of boot driver's I/O vector of transfer addresses.

(continued on next page)

Table C–2 (Cont.) Restart Parameter Block Fields

(R11)+	Field Name	Description
38:	RPB$L_IOVECSZ	Size of BOOT QIO routine.
3C:	RPB$L_FILLBN	LBN of secondary bootstrap image.
40:	RPB$L_FILSIZ	Size of secondary bootstrap image in blocks.
44:	RPB$Q_PFNMAP	The PFN bitmap is an array of bits, where each bit has the value "1" if the corresponding page of memory is valid, or has the value "0" if the corresponding page of memory contains a memory error. Through use of the PFNMAP, the operating system can avoid memory errors by avoiding known bad pages altogether. The memory bitmap is always page-aligned, and describes all the pages of memory from physical page #0 to the high end of memory, but excluding the PFN bitmap itself and the Q–bus map registers. If the high byte of the bitmap spans some pages available to the operating system and some pages of the PFN bitmap itself, the pages corresponding to the bitmap itself will be marked as bad pages. The first longword of the PFNMAP descriptor contains the number of bytes in the PFNMAP; the second longword contains the physical address of the bitmap.
4C:	RPB$L_PFNCNT	Count of "good" pages of physical memory, but not including the pages allocated to the Q22–bus scatter/gather map, the console scratch area, and the PFN bitmap at the top of memory.
50:	RPB$L_SVASPT	0.
54:	RPB$L_CSRPHY	Physical address of CSR for boot device.
58:	RPB$L_CSRVIR	0.
5C:	RPB$L_ADPPHY	Physical address of ADP (really the address of QMRs - ^x800 to look like a UBA adapter).
60:	RPB$L_ADPVIR	0.
64:	RPB$W_UNIT	Unit number of boot device.
66:	RPB$B_DEVTYP	Device type code of boot device.
67:	RPB$B_SLAVE	Slave number of boot device.

(continued on next page)

Table C–2 (Cont.) Restart Parameter Block Fields

(R11)+	Field Name	Description
68:	RPB$T_FILE	Name of secondary bootstrap image (defaults to [SYS0.SYSEXE]SYSBOOT.EXE). This field (up to 40 bytes) is overwritten with the input string on a "solicit" boot.

Note

1. For VAX/OpenVMS, the RPB$T_FILE must contain the root directory string "SYSn." on a non-network bootstrap. This string is parsed by SYSBOOT (SYSBOOT does not use the high nibble of BOOTR5).
2. The RPB$T_FILE is overwritten to contain the boot node name for compatibility with ELN VX.X (this field is only initialized this way when performing a network boot).

90:	RPB$B_CONFREG	Array (16 bytes) of adapter types (NDT$_UB0 - UNIBUS).
A0:	RPB$B_HDRPGCNT	Count of header pages.
A1:	RPB$W_BOOTNDT	Boot adapter nexus device type. Used by SYSBOOT and INIADP (OF SYSLOA) to configure the adapter of the boot device (changed from a byte to a word field in Version 12 of VMB).
B0:	RPB$L_SCBB	Physical address of SCB.
BC:	RPB$L_MEMDSC	Count of pages in physical memory including both good and bad pages. The high 8 bits of this longword contain the TR #, which is always 0 for KA52.
C0:	RPB$L_MEMDSC+4	PFN of the first page of memory. This field is always 0 for KA50/51/55/56, even if page #0 is a bad page.

Note

No other memory descriptors are used.

104:	RPB$L_BADPGS	Count of "bad" pages of physical memory.

(continued on next page)

Table C–2 (Cont.) Restart Parameter Block Fields

(R11)+	Field Name	Description
108:	RPB$B_CTRLLTR	Boot device controller number biased by 1. In VAX/OpenVMS, this field is used by INIT (in SYS) to construct the boot device's controller letter. A 0 implies this field has not been initialized, else if initialized, A=1, B=2, etc. (this field was added in Version 13 of VMB).
nn:		The rest of the RPB is zeroed.

C.3 VMB Argument List

The VMB code will also initialize an argument list as shown in Table C–3 (the address of the argument list is passed in the AP).

Table C–3 VMB Argument List

(AP)+	Field Name	Description
04:	VMB$L_FILECACHE	Quadword filename.
0C:	VMB$L_LO_PFN	PFN of first page of physical memory (always 0, regardless of where 128 Kbytes of "good" memory starts).
10:	VMB$L_HI_PFN	PFN of last page of physical memory.
14:	VMB$Q_PFNMAP	Descriptor of PFN bitmap. First longword contains count of bytes in bitmap. Second longword contains physical address of bitmap. (Same rules as for RPB$Q_PFNMAP listed above.)
1C:	VMB$Q_UCODE	Quadword.
24:	VMB$B_SYSTEMID	48-bit (actually a quadword is allocated) booting node address which is initialized when performing a network boot. This field is copied from the Target System Address parameter of the parameters message. (The DECnet HIORD value is added if the field was two bytes.)
2C:	VMB$L_FLAGS	Set as needed.
30:	VMB$L_CI_HIPFN	Cluster interface high PFN.

(continued on next page)

Table C–3 (Cont.) VMB Argument List

(AP)+	Field Name	Description
34:	VMB$Q_NODENAME	Boot node name which is initialized when performing a network boot. This field is copied from the Target System Name parameter of the parameters message.
3C:	VMB$Q_HOSTADDR	Host node address (this value is only initialized when booting over the network). This field is copied from the Host System Address parameter of the parameters message.
44:	VMB$Q_HOSTNAME	Host node name (this value is only initialized when performing a network boot). This field is copied from the Host System Name parameter of the parameters message.
4C:	VMB$Q_TOD	Time of day (this value is only initialized when performing a network boot). The time of day is copied from the first eight bytes of the Host System Time parameter of the parameters message. (The time differential values are NOT copied.)
54:	VMB$L_XPARAM	Pointer to data retrieved from request of the parameter file.
58:		The rest of the argument list is zeroed.

D

Configurable Machine State

The KA50/51/55/56 CPU modules have many control registers that need to be configured for proper operation of the module. The following list shows the normal state of all configurable bits in the CPU module as they are left after the successful completion of power-up ROM diagnostics.

Note

The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95 and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, will appear on the console and/or printouts from time to time.

```
Configuration Register Bit Settings(* = reset state)

NCA:
-----
NCA_CSR1:       Mode Control and Diagnostic Status Register  (2102 0004)
                15:14:  CP2 MT Timer Prescaler
                        11 = 144000 cycles*  - needed for CQBIC 10ms No Grant
                                timeout

                13:12:  CP1 MT Timer Prescaler
                        00 = 144 cycles - minimum for passive releases, no
                                cycle should take longer than this.

                11:10:  NDAL Timeout Prescaler
                        00 = 3200 cycles* - this is longer than both NCA and
                                NMC transactions timeouts, preserves timeout
                                order.

                9:      CQBIC mode
                        0 =  CQBIC not present* - this is to avoid the
                                QBUS_TRANS deadlock.
```

```
                8:      IO2 ID enable
                        1 = enabled
                7:      Force wrong CP2 bus parity - off - diagnostic use only
                6:      Force wrong CP1 bus parity - off - diagnostic use only
                5:      Force wrong NDAL master parity - off - diagnostic use
                          only
                4:      Force wrong NDAL slave parity - off - diagnostic use
                          only
                3:      Enable prefetch
                        1 = enable CP bus prefetch on DMA reads
                2:      Force write buffer hit - off - diagnostic use only
                1:      Force CP2 bus owner - diagnostic use only
                        0 = disabled
                0:      Force CP1 bus owner - diagnostic use only
                        0 = disabled

ICCS:           Interval Clock Control and Status Register  (2100 0060)

                NOTE: OpenVMS sets ICCS, NICR to proper values.

                6:      Interrupt enable
                        0 = disabled*
                5:      Single step - off
                4:      Transfer
                        0 = disabled*
                0:      Run - increment every 1usec - off

NICR:           Next Interval Count Register  (2100 0064)
                31:0    Initial count value for ICR (FFFFD8F0* (10ms))

NMC:
-----
NMC_CSR0-7:     Memory Configuration Registers  (2101 8000 thru 2101 801C)

                NOTE: Diagnostics set these registers based on available memory

                31:     Base Address Valid
                        0 = not valid*
                        1 = valid
                28:24:  Base Address (0 on reset)
                        1MB RAM - all address bits used
                        4MB RAM - only <28:26> used
                2:1     RAM size
                        01 = 1MB RAM
                        10 = 4MB RAM
                        11 = non-existent bank
                0:      Mode
                        1 = 64-bit mode

NMC_CSR18:      Mode Control and Diagnostic Status Register  (2101 8048)
                31:     Fast Diagnostic Mode (FDM)
                        0 = disabled* - diagnostic use only

                30:     FDM Second pass
                        0 = disabled* - diagnostic use only
```

```
                29:     Diagnostic Checkbit mode
                        0 = disabled* - diagnostic use only

                28:     QBus on IO1
                        0 = QBus on IO2*

                27:     Enable soft error log (NDAL & memory related)
                        0 = disabled* - OpenVMS enables this

                26:     Flush BCache
                        0 = don't flush*

                24:17:  Memory diagnostic check bits (0*) - may not be read
                                                            as 0

                8:7:    NDAL Timeout Scaler
                        00 = 2600 cycles* - maximum to preserve timeout order

                6:      Disable memory error
                        0 = memory errors deteted and corrected*

                5:      Refresh interval timer select
                        0 = 328 cycles*

                4:2:    Force wrong parity on NDAL transactions - off
                              - diagnostic use only

                1:      Disable memory refresh
                        0 = memory refreshed*

                0:      Force refresh
                        0 = normal refresh*

NMC_CSR19:      6-bit Address and Mode Register  (2101 804C)
                16:     Ignore 6-bit mode
                        0 = 6-bits checked*
                15:     Disable 6-bit error
                        0 = 6-bit errors detected*
                14:6:   6-bit segment address (0*) - not used in normal
                                  operation
                5:3:    6-bit mask (0*) - not used in normal operation
                2:0:    6-bit operation mode
                        X00 = reconstruction mode* - not used in normal
                                  operation

NMC_OSCR:       6-bit Data Registers  (2101 0000 thru 2101 7FFF)
                23:12:  6-bit field 1 (0 at reset)
                11:0:   6-bit field 0 (0 at reset)

NVAX:
-----
CPUID:          CPU ID Register  (IPR E)
                7:0:    CPU identifcation = 0 (for single processor config.)

SID:            System Identification Register  (IPR 3E)
                NOTE: this register may only be written by microcode
```

Configurable Machine State

```
                31:24:  CPU type - 13hex (NVAX code)
                13:8:   Patch revision
                7:0:    Microcode revision

ICSR:           IBox Control and Status Register (IPR D3)
                0:      VIC enable
                        1 = enabled

ECR:            EBox Control Register  (IPR 7D)
                13:     FBox test enable
                        0 = disable* - diagnostic use only

                7:      Interval time mode
                        1 = full CPU implemented interval timer

                5:      S3 stall timeout
                        0 = counts cycles w/ timeout_enable asserted (~3 sec)*

                3:      FBox stage 4 bypass
                        1 = enabled - improves FBox latency

                2:      S3 external time base timeout
                        0 = disabled* - use internal time base

                1:      FBox enable
                        1 = enabled

                0:      Vector present
                        0 = no* - no vector option available at this time

MMAPEN:         Memory Map Enable Register  (IPR E6)
                0:      Memory map enable
                        0 = disabled* - OpenVMS enables this

PAMODE:         Physical Address Mode Register  (IPR E7)
                0:      Physical address mode
                        0 = 30-bit physical address space*

PCCTL:          PCache Control Register  (IPR F8)
                8:      PCache Electrical disable
                        0 = PCache enabled*

                7:5     MBox performance monitor mode (0*) - diagnostic use
                        only

                4:      PCache error enable
                        1 = enables PCache error detection

                3:      Bank select during force hit mode
                        0 = left bank selected if force hit mode enabled*
                                - diagnostic use only

                2:      Force hit
                        0 = disabled* - diagnostic use only

                1:      I_enable
                        1 = enable PCache for IREAD, INVAL, I_CF commands
```

```
                0:      D_enable
                        1 = enable PCache for INVAL, D-stream read/write/fill

CCTL:           CBox Control Register  (IPR A0)
                30:     Software ETM
                        0 = disabled* - diagnostic use only

                16:     Force NDAL parity error - off - diagnostic use only

                15:11:  Performance monitoring bits (0*) - diagnostic use only

                10:     Disable CBox write packer
                        0 = write packer enabled* - improves write latency

                9:      Read timeout time base
                        0 = external time base

                8:      Software ECC
                        0 = use correct ECC*

                7:      Disable BCache errors
                        0 = BCache errors detected*

                6:      Force Hit
                        0 = disabled* - diagnostic use only

                5:4:    BCache size
                        00 = 128 KB* (KA50/52/55)
                        10 = 512 KB  (KA51/53/54/56)

                3:2:    Data store speed
                        00 = 2 cycle read, 3 cycle write* (KA51/53/54/56)
                        01 = 3 cycle read, 4 cycle write  (KA50/52)
                        10 = 4 cycle read, 5 cycle write  (KA55)
                1:      Tag store speed
                        0 = 3 cycle read, 3 cycle write* (KA51/53/54/56)
                        1 = 4 cycle read, 4 cycle write  (KA50/52/55)

                0:      Enable BCache
                        1 = enabled

CQBIC:
------
SCR:            System Configuration Register  (2008 0000)
                14:     Halt enable
                        1 = BHALT to CQBIC HALTIN pin to cause halts

                12:     Page prefetch disable
                        1 = map prefetch disabled - historical latency reasons

                7:      Restart enable
                        0 = QBus restart causes ARB power-up reset*

                3:1:    ICR offset address select bits
                        0 = (AUX mode not supported)*
```

Configurable Machine State

```
ICR:            Interprocessor Communication Register  (2000 1F40)
                8:      AUX Halt
                        0 = no halt - AUX mode not supported

                6:      ICR interrupt enable
                        0 = interprocessor interrupts disabled - only
                                uniprocessor config. allowed

                5:      Local memory external access enable
                        0 = external access disabled* - OpenVMS configures map

QBMBR:          Q-Bus Map Base Address Register (2008 0010)
                28:15:  address where 8K QBus mapping register are located
                        (undefined at reset)

SHAC:
-----
NOTE:  all SHAC registers are set up by OpenVMS driver

PQBBR:          Port Queue Block Base Register  (2000 4248)
                20:0:   upper bits of physical address of base of Port Queue
                        block.  Contains HW version, FW version, shared host
                        memory version and CI port maintenance ID at power-up.

PPR:            Port Parameter Register  (2000 4258)
                31:29:  Cluster size.  For SHAC value = 0.
                28:16:  Internal buffer length = 0* (For SHAC value = 1010 hex)
                7:0:    Port number.  Same as SHAC's DSSI ID.

PMCSR:          Port Maintenance Control and Status Register  (2000 425C)
                2:      Interrupt enable
                        0 = disabled*
                1:      Maintenance timer disable
                        0 = enabled*

SGEC:
-----
NOTE:  all SGEC registers are set up by OpenVMS driver

NICSR0:         Vector Address, IPL, Synch/Asynch Register  (2000 8000)
                31:30:  Interrupt priority
                        00 = 14*
                29:     Synch/Asynch bus master operating mode
                        0 = asynchronous*
                15:0:   Interrupt vector = 0003hex*
```

```
NICSR6:     Command and Mode Register  (2000 8018)
            30:     Interrupt enable
                    0 = disabled*
            28:25:  Burst limit mode
                    maximum number of longwords transferred in a single
                    DMA burst.  1*,2,4,8 when NICSR<19>is clear; 1*,4
                    when set.
            20:     Boot message enable mode
                    0 = disabled*
            19:     Single cycle enable mode
                    0 = disabled*
            11:     Start/Stop transmission command
                    0 = SGEC transmission process in stopped state*
            10:     Start/Stop reception command
                    0 = SGEC reception process in stopped state*
            9:8:    Operating mode
                    00 = normal mode*
            7:      Disable data chaining mode
                    0 = frames too long for current receive buffer
                        will be transferred to the next buffer(s)
                        in receive list*
            6:      Force collision mode (internal loopback mode only)
                    0 = no collision*
            3:      Pass bad frames mode
                    0 = bad frames discarded*
            2:1:    Address filtering mode
                    00 = normal mode*

NICSR7:     System Base Register  (2000 801C)
            29:0:   System base address - physical starting address of
                    the VAX system page table (unpredictable after reset)

NICSR9:     Watchdog Timers Register  (2000 8024)
            31:16:  Receive watchdog timeout
                    0 = never timeout*
                    default = 1250 = 2 ms
                    range = 72 µs (45) to 100 ms
            15:0:   Transmit watchdog timeout
                    0 = never timeout*
                    default = 1250 = 2 ms
                    range = 72 µs (45) to 100 ms

SSC:
----
SSCBAR:     SSC Base Address Register  (2014 0000)
            29:0    Base Address (reset value = 20140000)

SSCCR:      SSC Configuration Register  (2014 0010)
            27:     Interrupt vector disable
                    0 = interrupt vector enabled*
```

Configurable Machine State

```
                25:24:  IPL Level
                        00 = 14*

                23:     ROM access time
                        0 = 350 ns*

                22:20:  ROM size
                        110 = 512KB

                18:16:  Halt protected space
                        110 = 20040000 - 200BFFFF (historical)

                15:     n/a

                14:12:  n/a

                6:      Programmable address strobe 1 ready enable (for BDR)
                        1 = ready asserted after address strobe

                5:4:    Programmable address strobe 1 enable (for BDR)
                        11 = read enabled, write enabled

                2:      Programmable address strobe 0 ready enable
                        0 = no ready after address strobe*  Used for FEPROM
  programming

                1:0:    Programmable address strobe 0 enable
                        00 = read disabled, write disabled* Used for FEPROM
  programming
SSCBT:          SSC Bus Time Out Register  (2014 0020)
                23:0:   Bus timeout interval = 4000hex (16.384 ms)
                        range = 1 to FFFFFF (1 µs to 16.77 sec)

ADS0MAT:        Programmable Address Strobe 0 Match Register  (2014 0130)
                29:2:   Match address
                        0 = disabled*

ADS0MAS:        Programmable Address Strobe 0 Mask Register  (2014 0134)
                29:2:   Mask address bits

ADS1MAT:        Programmable Address Strobe 1 Match Register  (2014 0140)
                29:2:   Match address = 20084000 (for BDR)

ADS1MAS:        Programmable Address Strobe 1 Mask Register  (2014 0144)
                29:2:   Mask address bits = 7C (for BDR)

T1CR:           Programmable Timer 0 Control Register  (2014 0100)
                6:      Interrupt enable
                        0 = disabled*
                2:      STP
                        0 = run after overflow*
                0:      RUN
                        0 = counter not running*  (historical)
```

```
T1CR:       Programmable Timer 1 Control Register  (2014 0110)
            6:      Interrupt enable
                    0 = disabled*
            2:      STP
                    0 = run after overflow*
            0:      RUN
                    1 = counter incrementing every microsecond (historical)

TNIR:       Programmable Timer Next Interval Registers  (2014 0108,
                                                         2014 0118)
            31:0:   Timer next interval count (use 2's complement)
                    range = 0* to 1.2 hours

T0IV:       Programmable Timer 0 Interrupt Vector Register  (2014 010C)
            9:2:    Timer interrupt vector = 78hex

T1IV:       Programmable Timer 1 Interrupt Vector Registers (2014 011C)
            9:2:    Timer interrupt vector = 7Chex

TOY:        Time of Year Register  (2014 006C)
            31:0:   Number of 10 ms intervals since written

DLEDR:      Diagnostic LED Register  (2014 0030)
            3:0:    Display bits
                    0 = LEDs on*  (historical)
```

E

NVRAM Partitioning

This appendix describes how the CPU firmware partitions the SSC 1 KB battery-backed-up (BBU) RAM.

Note

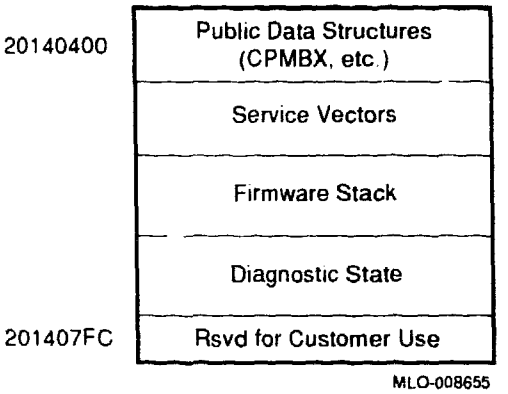
The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95 and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, will appear on the console and/or printouts from time to time.

E.1 SSC RAM Layout

The KA50/51/55/56 firmware uses the 1K byte of NVRAM on the SSC (see Figure E–1), for storage of firmware specific data structures and other information that must be preserved across power cycles. This NVRAM resides in the SSC chip starting at address 20140400. The NVRAM should not be used by the operating systems except as documented below. This NVRAM is not reflected in the bitmap built by the firmware.

NVRAM Partitioning
E.1 SSC RAM Layout

Figure E–1 KA50/51/55/56 SSC NVRAM Layout



E.1.1 Public Data Structures

Public data structures consist of three bytes, NVR0, NVR1, and NVR2. Their functions are described in Table E–1, Table E–2, and Table E–3.

E.1.1.1 Console Program MailBoX (CPMBX)

The Console Program MailBoX (CPMBX) comprised of NVR0, is a software data structure located at the beginning of NVRAM (20140400). The CPMBX is used to pass information between the CPU firmware and diagnostics, VMB, or an operating system.

Figure E–2 NVR0 (20140400) : Console Program MailBoX (CPMBX)

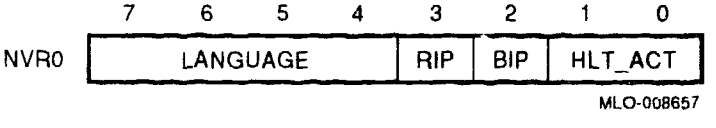


Table E–1 Bit Functions for NVR0

Field	Name	Description
7:4	LANGUAGE	This field specifies the current selected language for displaying halt and error messages on terminals which support MCS.
3	RIP	If set, a restart attempt is in progress. This flag must be cleared by the operating system, if the restart succeeds.

(continued on next page)

Table E–1 (Cont.) Bit Functions for NVR0

Field	Name	Description
2	BIP	If set, a bootstrap attempt is in progress. This flag must be cleared by the operating system if the bootstrap succeeds.
1:0	HLT_ACT	Processor halt action - this field in conjunction with the conditions specified for system halts is used to control the automatic restart/bootstrap procedure. HLT_ACT is normally written by the operating system. 0 : Restart; if that fails, reboot; if that fails, halt. 1 : Restart; if that fails, halt. 2 : Reboot; if that fails, halt. 3 : Halt.

E.1.1.2 Terminal Status

Figure E–3 NVR1 (20140401)

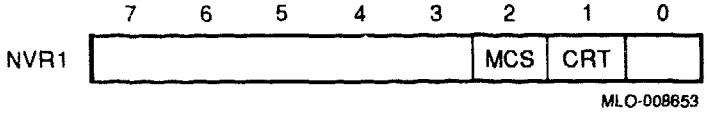


Table E–2 Bit Functions for NVR1

Field	Name	Description
2	MCS	If set, indicates that the attached terminal supports Multinational Character Set. If clear, MCS is not supported.
1	CRT	If set, indicates that the attached terminal is a CRT. If clear, indicates that the terminal is hardcopy.

E.1.1.3 Keyboard Status

Figure E–4 NVR2 (20140402)

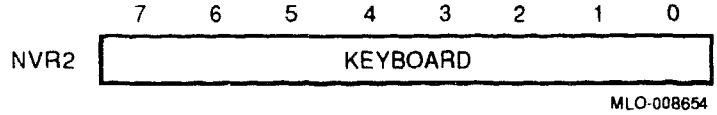


Table E–3 Bit Functions for NVR2

Field	Name	Description
7:0	KEYBOARD	This field indicates the national keyboard variant in use.

E.1.2 Service Vectors

Service vectors point to the routines for the reading or writing of characters by the console.

E.1.3 Firmware Stack

This section contains the stack that is used by all of the firmware, with the exception of VMB, which has its own built-in stack.

E.1.4 Diagnostic State

This area is used by the firmware resident diagnostics. It serves as the primary communications mechanism between the diagnostics and the console program.

E.1.5 USER Area

The KA50/51/55/56 console reserves the last longword (address 201407FC) of the NVRAM for customer use. This location is not tested by the console firmware. Its value is undefined.

F

MOP Counters

The following counters are kept for the Ethernet boot channel. All counters are unsigned integers. V4 counters rollover on overflow. All V3 counters "latch" at their maximum value to indicate overflow. Unless otherwise stated, all counters include both normal and multicast traffic. Furthermore, they include information for all protocol types. Frames received and bytes received counters do not include frames received with errors. Table F–1 displays the byte lengths and ordering of all the counters in both MOP Versions 3.0 and 4.0.

Table F–1 MOP Counter Block

	V3		V4		
Name	**Off**	**Len**	**Off**	**Len**	**Description**
TIME_SINCE_CREATION	00	2	00	16	**Time since last zeroed.** The time which has elapsed, since the counters were last zeroed. Provides a frame of reference for the other counters by indicating the amount of time they cover. For MOP V3, this time is the number of seconds. MOP V4 uses the UTC Binary Relative Time format.

(continued on next page)

Table F–1 (Cont.) MOP Counter Block

Name	V3 Off	V3 Len	V4 Off	V4 Len	Description
Rx_BYTES	02	4	10	8	**Bytes received.** The total number of user data bytes successfully received. This does not include Ethernet data link headers. This number is the number of bytes in the Ethernet data field, which includes any padding or length fields when they are enabled. These are bytes from frames that passed hardware filtering. Wnen the number of frames received is used to calculate protocol overhead, the overhead plus bytes received provides a measurement of the amount of Ethernet bandwidth (over time) consumed by frames addressed to the local system.
Tx_BYTES	06	4	18	8	**Bytes sent.** The total number of user data bytes successfully transmitted. This does not include Ethernet data link headers or data link generated retransmissions. This number is thc number of bytes in the Ethernet data field, which includes any padding or length fields when they are enabled. When the number of frames sent is used to calculate protocol overhead, the overhead plus bytes sent provides a measurement of the amount of Ethernet bandwidth (over time) consumed by frames sent by the local system.

(continued on next page)

Table F–1 (Cont.) MOP Counter Block

Name	V3 Off	V3 Len	V4 Off	V4 Len	Description
Rx_FRAMES	0A	4	20	8	**Frames received.** The total number of frames successfully received. These are frames that passed hardware filtering. Provides a gross measurement of incoming Ethernet usage by the local system. Provides information used to determine the ratio of the error counters to successful transmits.
Tx_FRAMES	0E	4	28	8	**Frames sent.** The total number of frames successfully transmitted. This does not include data link generated retransmissions. Provides a gross measurement of outgoing Ethernet usage by the local system. Provides information used to determine the ratio of the error counters to successful transmits.
Rx_MCAST_BYTES	12	4	30	8	**Multicast bytes received.** The total number of multicast data bytes successfully received. This does not include Ethernet data link headers. This number is the number of bytes in the Ethernet data field. In conjunction with total bytes received, provides a measurement of the percentage of this system's receive bandwidth (over time) that was consumed by multicast frames addressed to the local system.
Rx_MCAST_FRAMES	16	4	38	8	**Multicast frames received.** The total number of multicast frames successfully received. In conjunction with total frames received, provides a gross percentage of the Ethernet usage for multicast frames addressed to this system.

(continued on next page)

Table F–1 (Cont.) MOP Counter Block

Name	V3 Off	V3 Len	V4 Off	V4 Len	Description
Tx_INIT_DEFFERED	1A	4	40	8	**Frames sent[1], initially deferred.** The total number of times that a frame transmission was deferred on its first transmission attempt. In conjunction with total frames sent, measures Ethernet contention with no collisions.
Tx_ONE_COLLISION	1E	4	48	8	**Frames sent [1], single collision.** The total number of times that a frame was successfully transmitted on the second attempt after a normal collision on the first attempt. In conjunction with total frames sent, measures Ethernet contention at a level where there are collisions but the backoff algorithm still operates efficiently.
Tx_MULTI_COLLISION	22	4	50	8	**Frames sent[1], multiple collisions.** The total number of times that a frame was successfully transmitted on the third or later attempt after normal collisions on previous attempts. In conjunction with total frames sent, measures Ethernet contention at a level where there are collisions and the backoff algorithm no longer operates efficiently. NO SINGLE FRAME IS COUNTED IN MORE THAN ONE OF THE ABOVE THREE COUNTERS.

[1]Only one of these three counters will be incremented for a given frame.

(continued on next page)

Table F–1 (Cont.) MOP Counter Block

Name	V3 Off	V3 Len	V4 Off	V4 Len	Description
TxFAIL_COUNT	26	2	-	-	**Send failure count[2].** The total number of times a transmit attempt failed. Each time the counter is incremented, a type of failure is recorded. When Read-counter function reads the counter, the list of failures is also read. When the counter is set to zero, the list of failures is cleared. In conjunction with total frames sent, provides a measure of significant transmit problems. TxFAIL_BITMAP contains the possible reasons.
TxFAIL_BITMAP	2C	2	-	-	**Send failure reason bitmap[2].** This bitmap lists the types of transmit failures that occurred as summarized below: 0 - Excessive collisions 1 - Carrier detect failed 2 - Short circuit 3 - Open circuit 4 - Frame too long 5 - Remote failure to defer
TxFAIL_EXCESS_COLLS	-	-	58	8	**Send failure—Excessive collisions.** Exceeded the maximum number of retransmissions due to collisions. Indicates an overload condition on the Ethernet.

[2]V3 send/receive failures are collapsed into one counter with bitmap indicating which failures occurred.

(continued on next page)

Table F–1 (Cont.) MOP Counter Block

Name	V3 Off	V3 Len	V4 Off	V4 Len	Description
TxFAIL_CARIER_CHECK	-	-	60	8	**Send failure—Carrier check failed.** The data link did not sense the receive signal that is required to accompany the transmission of a frame. Indicates a failure in either the transmitting or receiving hardware. Could be caused by either transceiver, transceiver cable, or a babbling controller that has been cut off.
TxFAIL_SHRT_CIRCUIT	-	-	68	8	**Send failure—Short circuit[3].** There is a short somewhere in the local area network coaxial cable or the transceiver or controller /transceiver cable has failed. This indicates a problem either in local hardware or global network. The two can be distinguished by checking to see if other systems are reporting the same problem.
TxFAIL_OPEN_CIRCUIT	-	-	70	8	**Send failure—Open circuit[3].** There is a break somewhere in the local area network coaxial cable. This indicates a problem either in local hardware or global network. The two can be distinguished by checking to see if other systems are reporting the same problem.
TxFAIL_LONG_FRAME	-	-	78	8	**Send failure—Frame too long[3].** The controller or transceiver cut off transmission at the maximum size. This indicates a problem with the local system. Either it tried to send a frame that was too long or the hardware cutoff transmission too soon.

[3]Always zero.

(continued on next page)

Table F–1 (Cont.) MOP Counter Block

Name	V3 Off	V3 Len	V4 Off	V4 Len	Description
TxFAIL_REMOTE_DEFER	-	-	80	8	**Send failure—Remote failure to defer[3].** A remote system began transmitting after the allowed window for collisions. This indicates either a problem with some other system's carrier sense or a weak transmitter.
RxFAIL_COUNT	2A	2	-	-	**Receive failure count[2].** The total number of frames received with some data error. Includes only data frames that passed either physical or multicast address comparison. This counter includes failure reasons in the same way as the send failure counter. In conjunction with total frames received, provides a measure of data related receive problems. RxFAIL_BITMAP contains the possible reasons.
RxFAIL_BITMAP	2C	2	-	-	**Receive failure reason bitmap[2].** This bitmap lists the types of receive failures that occurred as summarized below: 0 - Block check failure 1 - Framing error 2 - Frame too long
RxFAIL_BLOCK_CHECK	-	-	88	8	**Receive failure—Block check error.** A frame failed the CRC check. This indicates several possible failures, such as EMI, late collisions, or improperly set hardware parameters.

[2]V3 send/receive failures are collapsed into one counter with bitmap indicating which failures occurred.

[3]Always zero.

(continued on next page)

Table F–1 (Cont.) MOP Counter Block

Name	V3 Off	V3 Len	V4 Off	V4 Len	Description
RxFAIL_FRAMING_ERR	-	-	90	8	**Receive failure—Framing error.** The frame did not contain an integral number of 8 bit bytes. This indicates several possible failures, such as EMI, late collisions, or improperly set hardware parameters.
RxFAIL_LONG_FRAME	-	-	98	8	**Receive failure—Frame too long**[3]. The frame was discarded because it was outside the Ethernet maximum length and could not be received. This indicates that a remote system is sending invalid length frames.
UNKNOWN_DESTINATION	2E	2	A0	8	**Unrecognized frame destination.** The number of times a frame was discarded because there was no portal with the protocol type or multicast address enabled. This includes frames received for the physical address, the broadcast address, or a multicast address.
DATA_OVERRUN	30	2	A8	8	**Data overrun.** The total number of times the hardware lost an incoming frame because it was unable to keep up with the data rate. In conjunction with total frames received, provides a measure of hardware resource failures. The problem reflected in this counter is also captured as an event.

[3]Always zero.

(continued on next page)

Table F–1 (Cont.) MOP Counter Block

Name	V3 Off	V3 Len	V4 Off	V4 Len	Description
NO_SYSTEM_BUFFER	32	2	B0	8	**System buffer unavailable[3].** The total number of times no system buffer was available for an incoming frame. In conjunction with total frames received, provides a measure of system buffer related receive problems. The problem reflected in this counter is also captured as an event. This can be any buffer between the hardware and the user buffers (those supplied on Receive requests). Further information as to potential different buffer pools is implementation specific.
NO_USER_BUFFER	34	2	B8	8	**User buffer unavailable[3].** The total number of times no user buffer was available for an incoming frame that passed all filtering. These are the buffers supplied by users on Receive requests. In conjunction with total frames received, provides a measure of user buffer related receive problems. The problem reflected in this counter is also captured as an event.
FAIL_COLLIS_DETECT	-	-	C0	8	**Collision detect check failure.** The approximate number of times that collision detect was not sensed after a transmission. If this counter contains a number roughly equal to the number of frames sent, either the collision detect circuitry is not working correctly or the test signal is not implemented.

[3]Always zero.

G

Error Messages

The error messages issued by the KA50/51/55/56 firmware fall into three categories: halt code messages, VMB error messages, and console messages.

G.1 Machine Check Register Dump

Some error conditions, such as machine check, generate an error summary register dump preceding the error message. For example, examining a nonexistent memory location results in the following display:

```
 >>e/p/l 20000000

MESR=00006000     MEAR=08406010     MMCDSR=01111100   MOAMR=00000000
CESR=80000200     CMCDSR=0000C108   CSEAR1=00000000   CSEAR2=00000000
CIOEAR1=00000000  CIOEAR2=00000000  CNEAR=00000000    ICSR=00000001
PCSTS=FFFFF800    PCADR=FFFFFFF8    TBSTS=C00000E0    TBADR=F5755754
NESTS=0000000C    NEOADR=E014066C   NEOCMD=8000F005   NEICMD=000003FF
NEDATHI=FFFFFFFF  NEDATLO=FF7F9FFF  CEFSTS=0001920A   CEFADR=E0000000
BCETSTS=000003E0  BCETIDX=FFFFFFE0  BCETAG=FFFFFE00   BCEDSTS=00000F00
BCEDIDX=001FFFF8  BCEDECC=00000000  CBTCR=00004000    DSER=00000080
QBEAR=00000000    DEAR=00000000     IPCR0=0000        ECR=0000008A
SCSICSR4=00       SCSICSR6=C0       SCSICSR5=00
?7D MACHINE CHECK  80060000 00000000 20048C68 20048C59 20048C55 40110080

>>>
```

G.2 Halt Code Messages

Except on power-up, which is not treated as an error condition, the following halt messages are issued by the firmware whenever the processor halts (Table G–1).

For example, if the processor encounters a HALT instruction while in kernel mode, the processor halts and the firmware displays the following before entering console I/O mode:

```
?06 HLT INST
PC = 800050D3
```

G.2 Halt Code Messages

The number preceding the halt message is the "halt code." This number is obtained from SAVPSL<13:8>(RESTART_CODE), IPR 43, which is saved on any processor restart operation.

Table G–1 HALT Messages

Code	Message	Description
?02	EXT HLT	External halt, caused by either console BREAK condition, Q22-bus BHALT_L, or DBR<AUX_HLT> bit was set while enabled.
_03	—	Power-up, no halt message is displayed. However, the presence of the firmware banner and diagnostic countdown indicates this halt reason.
?04	ISP ERR	In attempting to push state onto the interrupt stack during an interrupt or exception, the processor discovered that the interrupt stack was mapped NO ACCESS or NOT VALID.
?05	DBL ERR	The processor attempted to report a machine check to the operating system, and a second machine check occurred.
?06	HLT INST	The processor executed a HALT instruction in kernel mode.
?07	SCB ERR3	The SCB vector had bits <1:0> equal to 3.
?08	SCB ERR2	The SCB vector had bits <1:0> equal to 2.
?0A	CHM FR ISTK	A change mode instruction was executed when PSL<IS> was set.
?0B	CHM TO ISTK	The SCB vector for a change mode had bit <0> set.
?0C	SCB RD ERR	A hard memory error occurred while the processor was trying to read an exception or interrupt vector.
?10	MCHK AV	An access violation or an invalid translation occurred during machine check exception processing.
?11	KSP AV	An access violation or translation not valid occurred during processing of a kernel stack not valid exception.
?12	DBL ERR2	Double machine check error. A machine check occurred while trying to service a machine check.
?13	DBL ERR3	Double machine check error. A machine check occurred while trying to service a kernel stack not valid exception.
?19	PSL EXC5[1]	PSL<26:24> = 5 on interrupt or exception.

[1]For the last six cases, the VAX architecture does not allow execution on the interrupt stack while in a mode other than kernel. In the first three cases, an interrupt is attempting to run on the interrupt stack while not in kernel mode. In the last three cases, an REI instruction is attempting to return to a mode other than kernel and still run on the interrupt stack.

(continued on next page)

Table G–1 (Cont.) HALT Messages

Code	Message	Description
?1A	PSL EXC6[1]	PSL<26:24> = 6 on interrupt or exception.
?1B	PSL EXC7[1]	PSL<26:24> = 7 on interrupt or exception.
?1D	PSL REI5[1]	PSL<26:24> = 5 on an REI instruction
?1E	PSL REI6[1]	PSL<26:24> = 6 on an REI instruction.
?1F	PSL REI7[1]	PSL<26:24> = 7 on an REI instruction.
?3F	MICROVERIFY FAILURE	Microcode power-up self-test failed.

[1]For the last six cases, the VAX architecture does not allow execution on the interrupt stack while in a mode other than kernel. In the first three cases, an interrupt is attempting to run on the interrupt stack while not in kernel mode. In the last three cases, an REI instruction is attempting to return to a mode other than kernel and still run on the interrupt stack.

G.3 VMB Error Messages

VMB issues the errors listed in Table G–2.

Table G–2 VMB Error Messages

Code	Message	Description
?40	NOSUCHDEV	No bootable devices found.
?41	DEVASSIGN	Device is not present.
?42	NOSUCHFILE	Program image not found.
?43	FILESTRUCT	Invalid boot device file structure.
?44	BADCHKSUM	Bad checksum on header file.
?45	BADFILEHDR	Bad file header.
?46	BADIRECTORY	Bad directory file.
?47	FILNOTCNTG	Invalid program image format.
?48	ENDOFFILE	Premature end of file encountered.
?49	BADFILENAME	Bad filename given.
?4A	BUFFEROVF	Program image does not fit in available memory.
?4B	CTRLERR	Boot device I/O error.

(continued on next page)

Table G–2 (Cont.) VMB Error Messages

Code	Message	Description
?4C	DEVINACT	Failed to initialize boot device.
?4D	DEVOFFLINE	Device is offline.
?4E	MEMERR	Memory initialization error.
?4F	SCBINT	Unexpected SCB exception or machine check.
?50	SCB2NDINT	Unexpected exception after starting program image.
?51	NOROM	No valid ROM image found.
?52	NOSUCHNODE	No response from load server.
?53	INSFMAPREG	The Q22–bus map initialization failed.
?54	RETRY	No devices bootable, retrying.
?55	IVDEVNAM	Invalid device name.
?56	DRVERR	Drive error.

G.4 Console Error Messages

The error messages listed in Table G–3 are issued in response to a console command that has error(s).

Table G–3 Console Error Messages

Code	Message	Description
?61	CORRUPTION	The console program database has been corrupted.
?62	ILLEGAL REFERENCE	Illegal reference. The requested reference would violate virtual memory protection, the address is not mapped, the reference is invalid in the specified address space, or the value is invalid in the specified destination.
?63	ILLEGAL COMMAND	The command string cannot be parsed.
?64	INVALID DIGIT	A number has an invalid digit.
?65	LINE TOO LONG	The command was too large for the console to buffer. The message is issued only after receipt of the terminating carriage return.
?66	ILLEGAL ADRRESS	The address specified falls outside the limits of the address space.

(continued on next page)

Table G–3 (Cont.) Console Error Messages

Code	Message	Description
?67	VALUE TOO LARGE	The value specified does not fit in the destination.
?68	QUALIFIER CONFLICT	Qualifier conflict; for example, two different data sizes are specified for an EXAMINE command.
?69	UNKNOWN QUALIFIER	The switch is unrecognized.
?6A	UNKNOWN SYMBOL	The symbolic address in an EXAMINE or DEPOSIT command is unrecognized.
?6B	CHECKSUM	The command or data checksum of an X command is incorrect. If the data checksum is incorrect, this message is issued, and is not abbreviated to "Illegal command".
?6C	HALTED	The operator entered a HALT command.
?6D	FIND ERROR	A FIND command failed either to find the RPB or 128 KB of good memory.
?6E	TIME OUT	During an X command, data failed to arrive in the time expected (60 seconds).
?6F	MEMORY ERROR	A machine check occurred with a code indicating a read or write memory error.
?70	UNIMPLEMENTED	Unimplemented function.
?71	NO VALUE QUALIFIER	Qualifier does not take a value.
?72	AMBIGUOUS QUALIFIER	There were not enough unique characters to determine the qualifier.
?73	VALUE QUALIFIER	Qualifier requires a value.
?74	TOO MANY QUALIFIERS	Too many qualifiers supplied for this command.
?75	TOO MANY ARGUMENTS	Too many arguments supplied for this command.
?76	AMBIGUOUS COMMAND	There were not enough unique characters to determine the command.
?77	TOO FEW ARGUMENTS	Insufficient arguments supplied for this command.
?78	TYPEAHEAD OVERFLOW	The typeahead buffer overflowed.
?79	FRAMING ERROR	A framing error was detected on the console serial line.
?7A	OVERRUN ERROR	An overrun error was detected on the console serial line.
?7B	SOFT ERROR	A soft error occurred.
?7C	HARD ERROR	A hard error occurred.
?7D	MACHINE CHECK	A machine check occurred.

(continued on next page)

Table G–3 (Cont.) Console Error Messages

Code	Message	Description
?7E	CONSOLE STACK OVERRUN	SSC RAM stack overflowed into NVR.
?7F	COMMAND NOT SUPPORTED	Command on similar modules not supported on this product.
?80	ILLEGAL PASSWORD	Password is not 16 characters in length.
?81	INCORRECT PASSWORD	Password entered does not match previously entered password.
?82	PASSWORD FACILITY NOT ENABLED	A password has not been set.

H

Related Documents

The following documents contain information relating to the maintenance of systems that use the KA50/51/55/56 CPU modules.

Title	Part Number[1]
Guide to BA42B-Based MicroVAX 3100 Systems Service Information Kit	EK–M3100–IN
MicroVAX 3100 Models 30, 40, 80, 85, 90, 95, 96 System Illustrated Parts Breakdown	EK–MV310–IP
MicroVAX 3100 BA42B Enclosure Maintenance	EK–M3100–MG
MicroVAX 3100 BA42B Enclosure System Options	EK–M3100–OP
OpenVMS Factory Installed Software User Guide	EK–A0377–UG

[1] # = current revision, which is always shipped.

Glossary

ASCII

American standard code for information interchange.

BFLAG

Boot FLAG is the longword supplied in the SET BFLAG and BOOT /R5: commands that qualify the bootstrap operation. SHOW BFLAG displays the current value.

BHALT

Q22–bus Halt signal is usually tied to the front panel Halt switch.

BIP

Boot In Progress flag in CPMBX<2>

Bootstrap

A link between console mode (the system firmware) and programming mode (the operating system).

Bugcheck

Software or hardware error fatal to OpenVMS processor or system.

Cache memory

A small, high-speed memory placed between slower main memory and the processor. A cache increases effective memory transfer rates and processor speed.

CMOS

Complementary metal oxide semiconductor.

CPMBX

Console Program Mailbox is used to pass information between operating systems and the firmware.

CRC

Character code recognition. The use of pattern recognition techniques to identify characters by automatic means.

CQBIC

CVAX to Q22–bus interface chip.

CSR

Control status register. A register used to control the operation of a device and record the status of an operation or both.

CPU

Central processing unit. The main unit of a computer containing the circuits that control the interpretation and execution of instructions. The CPU holds the main storage, arithmetic unit, and special registers.

DCOK

Q22–bus signal indicating dc power is stable. This signal is tied to the Restart switch on the System Control Panel.

DE

Diagnostic Executive is a component of the ROM-based diagnostics responsible for set-up, execution, and clean-up of component diagnostic tests.

DMA

Direct memory access. A method of accessing a device's memory without interacting with the device's CPU.

DNA

Digital Network Architecture.

EPROM

Erasable programmable read-only memory. EPROM is a type of read-only memory that can be erased by using ultraviolet light, returning the device to a blank state.

ECC

Error Correction Code. Code that carries out automatic error correction by performing an exclusive "or" operation on the transferred data and applying a correction mask.

Factory Installed Software (FIS)

Operating system software loaded into a system disk during manufacture. On site, the FIS is bootstraped in the system, prompting a predefined menu of questions on the final configuration.

FEPROM

Flash Erasable Programmable Read-Only Memory (FEPROM). FEPROMs use electrical (bulk) erasure rather than ultraviolet erasure.

FIFO

First-in/first-out. A method used for processing or recovering data in which the oldest item is processed or recovered first.

Firmware

Functionally it consists of diagnostics, bootstraps, console, and halt entry/exit code.

FPU

Floating-point unit. A unit that handles the automatic positioning of the decimal point during arithmetic operations.

FRU

Field replaceable unit.

GPR

General Purpose Registers. On the KA52/53, they are the sixteen standard VAX longword registers R0 through R15. The last four registers, R12 through R15, are also known by their unique mnemonics: AP (Argument Pointer), FP (Frame Pointer), SP (Stack Pointer), and PC (Program Counter), respectively.

Initialization

The sequence of steps that prepare the system to start. Initialization occurs after a system has been powered up.

IPL

Interrupt Priority Level ranges from 0 to 31 (0 to 1F hex).

IPR

Internal Processor Registers implemented by the processor chip set. These longword registers are only accessible with the instructions MTPR (Move To Processor Register) and MFPR (Move From Processor Register) and require kernel mode privileges. This document uses the prefix "PR$_" when referencing these registers.

ISE

Integrated storage element. An intelligent disk drive used on the Digital Storage Systems Interconnect.

IT

Interval timer.

LED

Light emitting diode.

Machine check

An operating system action triggered by certain system errors that can be fatal to system operation. Once triggered, machine check handler software analyzes the error, comparing it to predetermined failure scenarios. Three outcomes are possible: the system continues to run, the software program is halted, or the system crashes.

µs

Microsecond (10e-6 seconds)

MMJ

Modified modular jack.

MOP

Maintenance Operations Protocol specifies message protocol for network loopback assistance, network bootstrap, and remote console functions.

ms

Millisecond (10e-3 seconds)

MSCP

Mass Storage Control Protocol is used in Digital disks and tapes.

NVR

Nonvolatile random access memory. A memory device that retains information in the absence of power.

NVRAM

Nonvolatile RAM. On the KA52/53, this is 1 Kb of battery backed-up RAM on the SSC.

PC

Program Counter or R15.

PCB

Process Control Block is a data structure pointed to by the PR$_PCBB register and contains the current process' hardware context.

PFN

Page Frame Number is an index of a page (512 bytes) of local memory. A PFN is derived from the bit field <23:09> of a physical address.

PR$_ICC

Interval Clock Control and Status, IPR 24.

PR$_IPL

Interrupt Priority Level, IPR 18.

PR$_MAPEN

Memory Management Mapping Enable, IPR 56.

PR$_PCBB

Process Control Block Base register, IPR 16.

PR$_RXCS

R(X)eceive Console Status, IPR 32.

PR$_RXDB

R(X)eceive Data Buffer, IPR 33.

PR$_SAVISP

SAVed Interrupt Stack Pointer, IPR 41.

PR$_SAVPC

SAVed Program Counter, IPR 42.

PR$_SAVPSL

SAVed Program Status Longword, IPR 43.

PR$_SCBB

System Control Block Base register, IPR 17.

PR$_SISR

Software Interrupt Summary Register, IPR 21.

PR$_TODR

Time Of Day Register, IPR 27, is commonly referred to as the Time Of Year register or TOY clock.

PR$_TXCS

T(X)ransmit Console Status, IPR 34.

PR$_TXDB

T(X)ransmit Data Buffer, IPR 35.

PROM

Programmable read-only memory. A read-only memory device that can be programmed.

PSL, PSW

Processor Status Longword is the VAX extension of the PSW (Processor Status Word). The PSW (lower word) contains instruction condition codes and is accessible by nonprivileged users; however, the upper word contains system status information and is accessible by privileged users.

QBMB

Q22–bus Map Base Register found in the CQBIC determines the base address in local memory for the scatter/gather registers.

QDSS

Q22–bus video controller for workstations.

QMR

Q22–bus Map Register.

QNA

Q22–bus Ethernet controller module.

RAM

Random access memory. A read/write memory device.

RAP

Register address port.

RIP

Restart In Progress flag in CPMBX<3>.

ROM

Read-only memory. A memory device that cannot be altered during the normal use of the computer.

RPB

Restart parameter block.

SCB

System Control Block. A data structure pointed to by PR$_SCBB. It contains a list of longword exception and interrupt vectors.

SCSI

Small computer system interface. An interface designed for connecting disks and other peripheral devices to computer systems. SCSI is defined by an American National Standards Institute (ANSI) standard.

SDD

Symptom-Directed Diagnosis. Online analysis of nonfatal system errors in order to locate potential system fatal errors before they occur.

SGEC

Second Generation Ethernet Chip.

SHAC

Single Host Adapter Chip.

SP

Stack pointer. An address location that contains the address of the processor-defined stack. The processor-defined stack is an area of memory set aside for temporary storage or for procedure and interrupt service linkages.

SRM

Standard Reference Manual, as in *VAX SRM*.

SSC

System Support Chip.

TOY

Time of year.

VAXcluster configuration

A highly integrated organization of OpenVMS systems that communicate over a high-speed communications path. VAXcluster configurations have all the functions of single-node systems, plus the ability to share CPU resources, queues, and disk storage. Like a single-node system, the VAXcluster configuration provides a single security and management environment. Member nodes can share the same operating environment or serve specialized needs.

VMB

Virtual machine bootstrap. The VMB program loads and runs the operating system.

OpenVMS

Virtual memory system. The operating system for a VAX computer.

Index

A

Acceptance testing, 4–13 to 4–14
Algorithm
  to find a valid RPB, 4–32
  to restart operating system, 4–31
ANALYZE/ERROR, 5–14
  interpreting CPU errors using, 5–15
  interpreting DMA to host transaction faults using, 5–28
  interpreting memory errors using, 5–18
  interpreting system bus faults using, 5–26
ANALYZE/SYSTEM, 5–21
Asynchronous communications interfaces
  support for, 2–4
Asynchronous communications options
  list of, 2–4

B

Binary load and unload (X command), 3–35
Bits
  RPB$V_DIAG, 4–24
  RPB$V_SOLICT, 4–24
Block diagram, 1–3
Boot Block Format, 4–23
BOOT command, 3–13
Boot Flags
  RPB$V_BBLOCK, 4–23
Bootstrap
  conditions, 4–17
  definition of, 4–17
  disk and tape, 4–23
Bootstrap (cont'd)
  failure, 4–18
  initialization, 4–18
  memory layout, 4–19
  memory layout after successful bootstrap, 4–21
  network, 4–24
  preparing for, 4–18
  primary, 4–20
  secondary, 4–20

C

Comment command (!), 3–38
! (comment command), 3–38
Communications devices, 2–4
Communications options, 2–4
Configuration
  memory, 1–9
Connectors
  function of, 1–6
  identification of, 1–5
Console command
  LOGIN, 3–21
Console commands
  address space control qualifiers, 3–9
  address specifiers, 3–3
  binary load and unload (X), 3–35
  BOOT, 3–13
  ! (comment), 3–38
  CONTINUE, 3–15
  data control qualifiers, 3–9
  DEPOSIT, 3–15
  EXAMINE, 3–16
  FIND, 3–17

Console commands (cont'd)
HALT, 3–18
HELP, 3–18
INITIALIZE, 3–20
keywords, 3–10
list of, 3–11
MOVE, 3–22
NEXT, 3–23
qualifier and argument conventions, 3–3
qualifiers, 3–9
REPEAT, 3–24
SEARCH, 3–25
SET, 3–27
SHOW, 3–28
START, 3–31
symbolic addresses, 3–4
syntax, 3–3
TEST, 3–31
UNJAM, 3–35
X (binary load and unload), 3–35
Console error messages
sample of, 5–41
Console I/O mode
special characters, 3–2
Console port, testing, 5–58
Console security feature
values, 3–28
CONTINUE command, 3–15
Controls
function of, 1–6
identification of, 1–5

D

DEPOSIT command, 3–15
Device Dependent Bootstrap Procedures, 4–23
Diagnostic executive, 4–6
error field, 5–42
Diagnostics
relationship to UETP, 5–56
Diagnostics, RZ-series, 4–5
Diagnostic tests
list of, 4–7
parameters for, 4–7
DNA Maintenance Operations Protocol (MOP), 4–24
Documents
related, H–1

E

Entry Point
definition of, B–1
Error during UETP, 5–57
diagnosing, 5–56
Error Log Utility
relationship to UETP, 5–56
Error messages
console, sample of, 5–41
EXAMINE command, 3–16
External mass storage devices, 2–2

F

FE utility, 5–47
Files–11 lookup, 4–23
FIND command, 3–17
Firmware
power-up sequence, 4–1
updating, 6–1
Flags
restart in progress, 4–31

G

General purpose registers (GPRs)
symbolic addresses for, 3–4

H

H3103 loopback connector, 5–58
H8572 loopback connector, 5–58
Halt
dispatch, C–1
HALT command, 3–18
Halt protection, override, 5–48
HELP command, 3–18

I

Indicators
  function of, 1–6
  identification of, 1–5
INIT, 4–18
Initialization
  following a processor halt, 4–31
  prior to bootstrap, 4–18
INITIALIZE command, 3–20
Initial power-up test
  See IPR
Internal mass storage devices
  list of, 2–2
IPL_31, 4–19
iSYS$TEST logical name, 5–56

K

KA50/51/55/56 CPU module
  block diagram of, 1–3
KA50/51/55/56 system
  communications options, 2–4
  configurations of, 2–1
  mass storage device configurations, 2–1
  memory configurations, 2–1
KA50 CPU module
  features of, 1–1
KA50 system
  configurations of, 1–1
KA51 CPU module
  features of, 1–1
KA51 system
  configurations of, 1–1
KA55 CPU module
  features of, 1–1
KA55 system
  configurations of, 1–1

L

Language selection menu
  messages, list of, 4–2
Local Memory Partitioning, 4–19
Log file generated by UETP
  OLDUETP.LOG, 5–57
LOGIN command, 3–21
Loopback connectors
  H3103, 5–58
  H8572, 5–58
  list of, 5–59
Loopback tests, 5–58
  console port, 5–58
  Ethernet, 5–58

M

Mass storage devices, 2–1
  external, 2–2
  internal, 2–2
  SCSI ID assignments, 2–4
Memory
  acceptance testing of, 4–13
  configurations, 1–10
  expansion connector identification, 1–9
  expansion of, 1–9
  isolating FRU, 4–14, 5–48
  rules for adding, 1–9
  testing, 5–48
Memory configuration
  KA50/51/55/56 system, 2–1
Memory modules, 1–9
Memory option
  installation of, 1–11
Memory test, 1–13
MEM test, 1–13
MicroVAX data types
  support of, 1–4
MicroVAX instructions
  support of, 1–5
MOM$LOAD, 4–25

MOP, functions, 5–53
MOP functions, 4–26
MOP program load sequence, 4–25
MOVE command, 3–22
MS44 memory modules, 1–9

N

Network listening, 4–30
NEXT command, 3–23
NVRAM
  CPMBX, E–2
  partitioning, E–1

O

OLDUETP.LOG file, 5–56
Onboard memory
  location of, 1–9
OpenVMS
  error handling, 5–4
  event record translation, 5–14
Operating System
  bootstrap, 4–17
  restarting a halted, 4–31
Operating System Restart
  definition of, 4–31

P

Parameters
  for diagnostic tests, 4–9
  in error display, 5–42
Patchable Control Store
  Error messages, 6–8
PFN bitmap, 4–18
Ports
  function of, 1–6
  identification of, 1–5
POST
  See Power-on self-tests
Power-on self-tests
  description, 4–2
  errors handled by, 4–5
  kernel, 4–3
Power-on self-tests (cont'd)
  mass storage, 4–5
power-up
  machine state, 4–14
  memory layout, 4–15
Power-up sequence, 4–1
Power-up tests, 4–1
Primary Bootstrap, 4–20

R

Registers
  initializing the general purpose, 4–18
Related documents, H–1
REPEAT command, 3–24
REQ_PROGRAM, 4–30
Restart, 4–31
Restart Parameter Block (RPB)
  RIP flag, 4–31
ROM-based diagnostics, 4–6 to 4–10
  console displays during, 5–41
  isolating failures with, 5–43
  list of, 4–6
  parameters, 4–7
  utilities, 4–6
RPB
  initialization, C–5
  locating, 4–32
RPB Signature Format, 4–32
RZ-series ISE
  diagnostics, 4–5

S

Scripts, 4–11
  list of, 4–12
SCSI ID assignments
  recommendations for, 2–4
SEARCH command, 3–25
Secondary Bootstrap, 4–20
SET command, 3–27
SET HOST/DUP command, 3–27
SHOW command, 3–28

SICL messages, 5–34
converting appended MEL files, 5–37
START command, 3–31
Symbolic addresses, 3–4
for any address space, 3–7
for GPRs, 3–4
Synchronous communications options
list of, 2–5
Synchronous communications standards
support for, 2–5
System hang, 5–58

T

TEST command, 3–31
Tests, diagnostic
list of, 4–6
parameters for, 4–9
Troubleshooting
procedures, general, 5–2
UETP, 5–57

U

UETINIT01.EXE image, 5–57
UETP
interpreting OpenVMS failures with, 5–56
UETP.LOG file, 5–56
UNJAM, 4–18
UNJAM command, 3–35
User Environment Test Package (UETP)
interpreting output of, 5–56
running multiple passes of, 5–56
typical failures reported by, 5–57
Utilities, diagnostic, 4–6

V

VAXELN
and VMB, 4–20
VAXsimPLUS, 5–3, 5–32
customizing, 5–39
enabling SICL, 5–40
installing, 5–38
Virtual Memory Boot (VMB), 4–21
definition of, 4–20
primary bootstrap, 4–20
secondary bootstrap, 4–23

W

Warmstart, 4–31

X

X command (binary load and unload), 3–35

MicroVAX 3100 Model 85, 90, 95, 96

KA50/51/55/56 CPU System Maintenance

Order Number: EK–M3100–SM. B01

June 1995

This manual gives maintenance information for systems that use the KA50, KA51, KA55 or KA56 CPU module.

Digital Equipment Corporation
Maynard, Massachusetts

First Printing, February 1995
Revised June, 1995

Digital Equipment Corporation makes no representations that the use of its products in the manner described in this publication will not infringe on existing or future patent rights, nor do the descriptions contained in this publication imply the granting of licenses to make, use, or sell equipment or software in accordance with the description.

© Digital Equipment Corporation 1995. All Rights Reserved.

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

The following are trademarks of Digital Equipment Corporation: DEC, Digital, MicroVAX, OpenVMS, VAX, VAXsimPLUS, and the DIGITAL logo.

All other trademarks and registered trademarks are the property of their respective holders.

S2869

This document was prepared using VAX DOCUMENT Version 2.1.

Contents

Preface xi

1 KA50/51/55/56 CPU Module Description

1.1 KA50/51/55/56 CPU Module 1–1
1.1.1 Physical Description 1–2
1.1.2 Functional Description 1–3
1.2 MS44 and MS44L Memory Modules 1–9
1.3 MS44 or MS44L Memory Option Installation 1–11
1.4 Memory Tests 1–13

2 Configuration

2.1 Memory Configurations 2–1
2.2 Mass Storage Devices 2–1
2.2.1 Internal Mass Storage Devices 2–2
2.2.2 External Mass Storage Devices 2–2
2.2.3 SCSI ID Numbers 2–4
2.3 Communications Options 2–4
2.3.1 Asynchronous Communications Options 2–4
2.3.2 Synchronous Communications Options 2–5

3 KA50/51/55/56 Firmware Commands

3.1 Console I/O Mode Control Characters 3–2
3.1.1 Command Syntax 3–3
3.1.2 Address Specifiers 3–3
3.1.3 Symbolic Addresses 3–4
3.1.4 Console Numeric Expression Radix Specifiers 3–8
3.1.5 Console Command Qualifiers 3–9
3.1.6 Console Command Keywords 3–10
3.2 Console Commands 3–13
3.2.1 BOOT 3–13

3.2.2 CONTINUE ... 3–15
3.2.3 DEPOSIT ... 3–15
3.2.4 EXAMINE ... 3–16
3.2.5 FIND ... 3–17
3.2.6 HALT ... 3–18
3.2.7 HELP ... 3–18
3.2.8 INITIALIZE ... 3–20
3.2.9 LOGIN ... 3–21
3.2.10 MOVE ... 3–22
3.2.11 NEXT ... 3–23
3.2.12 REPEAT ... 3–24
3.2.13 SEARCH ... 3–25
3.2.14 SET ... 3–27
3.2.15 SHOW ... 3–28
3.2.16 START ... 3–31
3.2.17 TEST ... 3–31
3.2.18 UNJAM ... 3–35
3.2.19 X—Binary Load and Unload ... 3–35
3.2.20 ! (Comment) ... 3–38

4 System Initialization and Acceptance Testing (Normal Operation)

4.1 Basic Initialization Flow ... 4–1
4.2 Power-On Self-Tests (POST) ... 4–2
4.2.1 Power-Up Tests for Kernel ... 4–3
4.2.2 Power-Up Tests for Mass Storage Devices ... 4–5
4.3 CPU ROM-Based Diagnostics ... 4–6
4.3.1 Diagnostic Tests ... 4–6
4.3.2 Scripts ... 4–11
4.4 Basic Acceptance Test Procedure ... 4–13
4.5 Machine State on Power-Up ... 4–14
4.6 Main Memory Layout and State ... 4–14
4.6.1 Reserved Main Memory ... 4–15
4.6.1.1 PFN Bitmap ... 4–15
4.6.1.2 Scatter/Gather Map ... 4–16
4.6.1.3 Firmware "Scratch Memory" ... 4–16
4.6.2 Contents of Main Memory ... 4–16
4.6.3 Memory Controller Registers ... 4–17
4.6.4 On-Chip and Backup Caches ... 4–17
4.6.5 Translation Buffer ... 4–17
4.6.6 Halt-Protected Space ... 4–17
4.7 Operating System Bootstrap ... 4–17

4.7.1 Preparing for the Bootstrap 4–18
4.7.2 Primary Bootstrap Procedures (VMB) 4–20
4.7.3 Device Dependent Secondary Bootstrap Procedures 4–23
4.7.3.1 Disk and Tape Bootstrap Procedure 4–23
4.7.3.2 MOP Ethernet Functions and Network Bootstrap Procedure 4–24
4.7.3.3 Network "Listening" 4–30
4.8 Operating System Restart 4–31
4.8.1 Locating the RPB 4–32

5 System Troubleshooting and Diagnostics

5.1 Basic Troubleshooting Flow 5–1
5.2 Product Fault Management and Symptom-Directed Diagnosis . . . 5–3
5.2.1 General Exception and Interrupt Handling 5–3
5.2.2 OpenVMS Error Handling 5–4
5.2.3 OpenVMS Error Logging and Event Log Entry Format 5–6
5.2.4 OpenVMS Event Record Translation 5–14
5.2.5 Interpreting CPU Faults Using ANALYZE/ERROR 5–15
5.2.6 Interpreting Memory Faults Using ANALYZE/ERROR 5–18
5.2.6.1 Uncorrectable ECC Errors 5–18
5.2.6.2 Correctable ECC Errors 5–22
5.2.7 Interpreting System Bus Faults Using ANALYZE/ERROR . . . 5–26
5.2.8 Interpreting DMA ⇔ Host Transaction Faults Using ANALYZE/ERROR 5–28
5.2.9 VAXsimPLUS and System-Initiated Call Logging (SICL) Support 5–32
5.2.9.1 Converting the SICL Service Request MEL File 5–37
5.2.9.2 VAXsimPLUS Installation Tips 5–38
5.2.9.3 VAXsimPLUS Post-Installation Tips 5–39
5.2.10 Repair Data for Returning FRUs 5–41
5.3 Power-On Self-Test (POST) and ROM-Based Diagnostic (RBD) Failures 5–41
5.3.1 FE Utility 5–47
5.3.2 Overriding Halt Protection 5–48
5.3.3 Isolating Memory Failures 5–48
5.4 Using MOP Ethernet Functions to Isolate Failures 5–53
5.5 Interpreting User Environmental Test Package (UETP) OpenVMS Failures 5–56
5.5.1 Interpreting UETP Output 5–56
5.5.1.1 UETP Log Files 5–56
5.5.1.2 Possible UETP Errors 5–57
5.6 Using Loopback Tests to Isolate Failures 5–58

5.6.1 Testing the Console Port 5–58
5.6.2 Embedded Ethernet Loopback Testing 5–58

6 FEPROM Firmware Update

6.1 Preparing the Processor for a FEPROM Update 6–2
6.2 Updating Firmware via Ethernet 6–3
6.3 Updating Firmware via Tape 6–6
6.4 FEPROM Update Error Messages 6–7

A Address Assignments

A.1 KA50/51/55/56 General Local Address Space Map A–1
A.2 KA50/51/55/56 Detailed Local Address Space Map A–3
A.3 External, Internal Processor Registers A–8
A.4 Global Q22–bus Address Space Map A–9
A.5 Processor Registers A–9
A.6 IPR Address Space Decoding A–21

B ROM Partitioning

B.1 Firmware EPROM Layout B–1
B.1.1 System Identification Registers B–3
B.1.1.1 PR$_SID (IPR 62) B–3
B.1.1.2 SIE (20040004) B–3
B.1.2 Call-Back Entry Points B–4
B.1.2.1 CP$GETCHAR_R4 ..... B–5
B.1.2.2 CP$MSG_OUT_NOLF_R4 ..... B–6
B.1.2.3 CP$READ_WTH_PRMPT_R4 ..... B–6
B.1.3 Boot Information Pointers B–7

C Data Structures and Memory Layout

C.1 Halt Dispatch State Machine C–1
C.2 Restart Parameter Block C–5
C.3 VMB Argument List C–9

D Configurable Machine State

E NVRAM Partitioning

E.1 SSC RAM Layout E–1
E.1.1 Public Data Structures E–2
E.1.1.1 Console Program MailBoX (CPMBX) E–2
E.1.1.2 Terminal Status E–3
E.1.1.3 Keyboard Status E–3
E.1.2 Service Vectors E–4
E.1.3 Firmware Stack E–4
E.1.4 Diagnostic State E–4
E.1.5 USER Area E–4

F MOP Counters

G Error Messages

G.1 Machine Check Register Dump G–1
G.2 Halt Code Messages G–1
G.3 VMB Error Messages G–3
G.4 Console Error Messages G–4

H Related Documents

Glossary

Index

Examples

1–1 Successful Running of Memory Test Script A8 1–13
1–2 Typical Failure After Running Memory Test Script A8 1–14
4–1 Successful Diagnostic Countdown 4–2
4–2 Successful Power-Up to List of Bootable Devices 4–5
4–3 Test 9E 4–8
5–1 Error Log Entry Indicating CPU Error 5–16
5–2 SHOW ERROR Display Using the OpenVMS Operating System 5–17

5–3	Error Log Entry Indicating Uncorrectable ECC Error	5–20
5–4	SHOW MEMORY Display Under the OpenVMS Operating System	5–21
5–5	Using ANALYZE/SYSTEM to Check the Physical Address in Memory for a Replaced Page	5–22
5–6	Error Log Entry Indicating Correctable ECC Error	5–25
5–7	Error Log Entry Indicating Q-Bus Error	5–27
5–8	Error Log Entry Indicating Polled Error	5–29
5–9	Device Attention Entry	5–31
5–10	SICL Service Request with Appended MEL File	5–38
5–11	Sample Output with Errors	5–41
5–12	FE Utility Example	5–48
5–13	Failure Due to a Missing SIMM (One 16 Mbyte Set)	5–49
5–14	Failure Due to a Missing SIMM (Two 16 Mbyte Sets)	5–50
5–15	Failure Due to a Bad SIMM	5–51
5–16	SIMM Wrong Size	5–52
6–1	FEPROM Update via Ethernet	6–5
6–2	FEPROM Update via Tape	6–7

Figures

1–1	KA50/51/55/56 CPU Module	1–2
1–2	KA50/51/55/56 CPU Module Block Diagram	1–4
1–3	KA50/51/55/56 Controls, Indicators, Ports, and Connectors	1–6
1–4	Memory Expansion Connectors	1–10
1–5	Memory Module Installation	1–12
2–1	SZ Expansion Box Numbering System	2–3
4–1	Console Banner	4–3
4–2	Memory Layout After Power-Up Diagnostics	4–15
4–3	Memory Layout Prior to VMB Entry	4–20
4–4	Memory Layout at VMB Exit	4–22
4–5	Boot Block Format	4–24
4–6	Locating the Restart Parameter Block	4–32
5–1	Event Log Entry Format	5–8
5–2	Machine Check Stack Frame Subpacket	5–9
5–3	Processor Register Subpacket	5–10

5–4	Memory Subpacket for ECC Memory Errors	5–11
5–5	Memory SBE Reduction Subpacket (Correctable Memory Errors)	5–11
5–6	CRD Entry Subpacket Header	5–12
5–7	Correctable Read Data (CRD) Entry	5–13
5–8	Trigger Flow for the VAXsimPLUS Monitor	5–34
5–9	Five-Level VAXsimPLUS Monitor Display	5–36
6–1	Firmware Update Utility Layout	6–2
6–2	W4 Jumper Setting for Updating Firmware	6–3
B–1	KA50/51/55/56 FEPROM Layout	B–2
B–2	SID : System Identification Register	B–3
B–3	SIE : System Identification Extension (20040004)	B–4
B–4	Boot Information Pointers	B–8
E–1	KA50/51/55/56 SSC NVRAM Layout	E–2
E–2	NVR0 (20140400) : Console Program MailBoX (CPMBX)	E–2
E–3	NVR1 (20140401)	E–3
E–4	NVR2 (20140402)	E–3

Tables

1–1	Functions of Controls, Indicators, Connectors	1–6
1–2	KA50/51/55/56 CPU Module Memory Configurations	1–10
2–1	KA50/51/55/56 Internal Mass Storage Devices	2–2
2–2	Supported Asynchronous Communications Options	2–4
2–3	Supported Synchronous Communications Options	2–5
2–4	DSW42-AA Communications Support	2–5
3–1	Console Symbolic Addresses	3–4
3–2	Symbolic Addresses Used in Any Address Space	3–8
3–3	Console Radix Specifiers	3–8
3–4	Console Command Qualifiers	3–9
3–5	Command Keywords by Type	3–11
3–6	Console Command Summary	3–11
4–1	LED Codes	4–4
4–2	Scripts Available to Customer Services	4–12
4–3	Network Maintenance Operations Summary	4–26
4–4	Supported MOP Messages	4–27
4–5	MOP Multicast Addresses and Protocol Specifiers	4–31

5–1	OpenVMS Error Handler Entry Types	5–7
5–2	Conditions That Trigger VAXsimPLUS Notification and Updating	5–33
5–3	Five-Level VAXsimPLUS Monitor Screen Displays	5–35
5–4	KA50/51/55/56 Console Displays as Pointers to FRUs	5–44
5–5	Loopback Connectors for Common Devices	5–60
A–1	Processor Registers	A–9
A–2	IPR Address Space Decoding	A–21
B–1	System Identification Register	B–3
B–2	System Identification Extension	B–4
B–3	Call-Back Entry Points	B–5
C–1	Firmware State Transition Table	C–3
C–2	Restart Parameter Block Fields	C–6
C–3	VMB Argument List	C–9
E–1	Bit Functions for NVR0	E–2
E–2	Bit Functions for NVR1	E–3
E–3	Bit Functions for NVR2	E–4
F–1	MOP Counter Block	F–1
G–1	HALT Messages	G–2
G–2	VMB Error Messages	G–3
G–3	Console Error Messages	G–4

Preface

This manual describes the KA50 CPU module used in the MicroVAX 3100 Model 90, the KA51 CPU module used in the MicroVAX 3100 Model 95, the KA55 CPU module used in the MicroVAX 3100 Model 85, and the KA56 CPU module used in the MicroVAX 3100 Model 96 system. It provides the configuration guidelines, ROM-based diagnostic information, and troubleshooting information for systems containing the KA50/51/55/56 CPU modules.

Audience

This manual is for Digital Services personnel who provide support and maintenance for systems that use the KA50/51/55/56 CPU module. It is also for customers who have a self-maintenance agreement with Digital Equipment Corporation.

Structure of This Manual

This manual is divided into six chapters, eight appendixes, a glossary, and an index:

- Chapter 1 describes the KA50/51/55/56 CPU module.
- Chapter 2 describes the KA50/51/55/56 system configurations.
- Chapter 3 describes the console commands that you can enter at the console prompt.
- Chapter 4 describes the system initialization, testing and bootstrap process that occurs at power-up.
- Chapter 5 describes the error log interpretation of diagnostic testing, the ROM-based diagnostic testing, and troubleshooting procedures for the KA50/51/55/56 systems. Also, this chapter provides information on testing DSSI storage devices, using MOP Ethernet functions to isolate errors, and interpreting UETP failures.
- Chapter 6 describes the FEPROM firmware.

- Appendix A gives the address assignments.
- Appendix B describes ROM partitioning and subroutine entry points.
- Appendix C gives definitions of the key global data structures used by the CPU firmware.
- Appendix D gives the normal state of all configurable bits in the CPU module as they are left after the successful completion of power-up ROM diagnostics.
- Appendix E describes how the CPU firmware partitions the SCC 1 KB battery-backed-up (BBU) RAM.
- Appendix F gives MOP counters.
- Appendix G describes the error codes and messages that the system exerciser test generates.
- Appendix H gives a list of related documents.

Note

Examples in this manual may vary slightly from your particular MicroVAX 3100 system, since they are from various VAX and MicroVAX systems which share common features, options, diagnostics, and so on.

Conventions

The following conventions are used in this manual:

Convention	Description
Ctrl/*x*	Ctrl/*x* indicates that you hold down the Ctrl key while you press another key or mouse button (indicated here by *x*).
x	A lowercase italic *x* indicates the generic use of a letter. For example, *xxx* indicates any combination of three alphabetic characters.
n	A lowercase italic *n* indicates the generic use of a number. For example, 19*nn* indicates a 4-digit number in which the last 2 digits are unknown.
{}	In format descriptions, braces indicate required elements. You must choose one of the elements.
[ ]	In format descriptions, brackets indicate optional elements. You can choose none, one, or all of the options.
( )	In format descriptions, parentheses delimit the parameter or argument list.
...	In format descriptions, horizontal ellipsis points indicate one of the following: • An item that is repeated • An omission such as additional optional arguments • Additional parameters, values, or other information that you can enter
|	In format descriptions, a vertical bar separates similar options, one of which you can choose.
italic type	Italic type emphasizes important information, indicates variables, and indicates the complete titles of manuals.
boldface type	Boldface type in examples indicates user input. Boldface type in text indicates the first instance of terms defined either in the text, in the glossary, or both.
n nnn.nnn nn	A space character separates groups of 3 digits in numerals with 5 or more digits. For example, *10 000* equals *ten thousand.*
n.nn	A period in numerals signals the decimal point indicator. For example, *1.75* equals *one and three-fourths.*
MONOSPACE	Text displayed on the screen is shown in monospace type.

Convention	Description
Radix indicators	The radix of a number is written as a word enclosed in parentheses, for example, 23(decimal) or 34(hexadecimal).
>>>	Three right angle brackets indicate the console prompt.
UPPERCASE	A word in uppercase indicates a command.
Note	A note contains information that is of special importance to the user.
Caution	A caution contains information to prevent damage to the equipment.
Warning	A warning contains information to prevent personal injury.

1

KA50/51/55/56 CPU Module Description

This chapter describes the KA50 **central processing unit** (CPU) module that is used in the MicroVAX 3100 Model 90, the KA51 CPU module that is used in the MicroVAX 3100 Model 95, the KA55 CPU module that is used in the MicroVAX 3100 Model 85 system, and the KA56 CPU module that is used in the MicroVAX 3100 Model 96. It gives information on the following:

- KA50/51/55/56 CPU modules
- MS44 or MS44L memory modules

The KA50, KA51, KA55 and KA56 are similar in design, and the information in this document is applicable for each of them except where noted. The differences between the KA50, KA51, KA55, and KA56 CPUs are as follows:

	KA50	KA51	KA55	KA56
Speed	286Mhz (14ns)	333Mhz (12ns)	250Mhz (16ns)	400Mhz (10ns)
VIC	2Kb	2Kb	disabled	2Kb
P-cache	8Kb	8Kb	8Kb	8Kb
B-cache	128Kb	512Kb	128Kb	512Kb

1.1 KA50/51/55/56 CPU Module

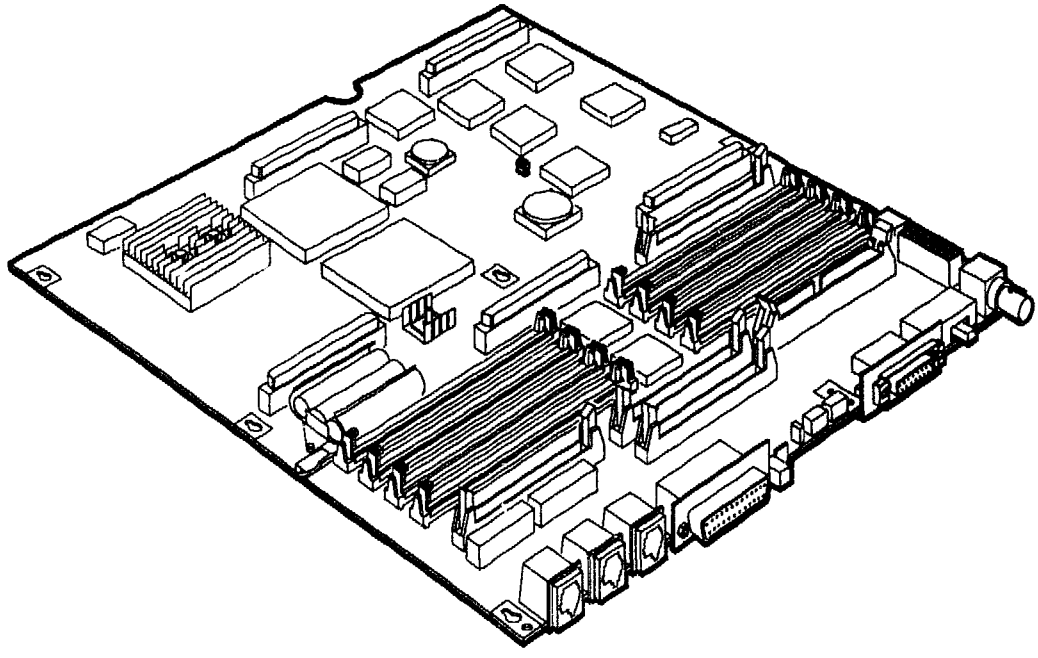
The KA50/51/55/56 CPU module is based on the NVAX chip set. It uses MS44 or MS44L memory modules and a set of supported **small computer system interface** (SCSI) devices. Figure 1–1 shows the KA50 CPU module; the KA51, KA55 and KA56 modules are similar.

1.1.1 Physical Description

The KA50/51/55/56 CPU module is the primary component of the MicroVAX 3100 system in which it is installed. The KA50/51/55/56 CPU module contains the following components:

- The NVAX processor chip—This chip is a complementary metal oxide semiconductor (CMOS) virtual memory microprocessor. The key features of the chip are as follows:
  - Support for the MicroVAX chip subset of the VAX instruction set
  - Support for the MicroVAX chip subset of the VAX data types
  - Full VAX memory management
  - 30-bit physical memory addressing

Figure 1–1 KA50/51/55/56 CPU Module



MLO-009931

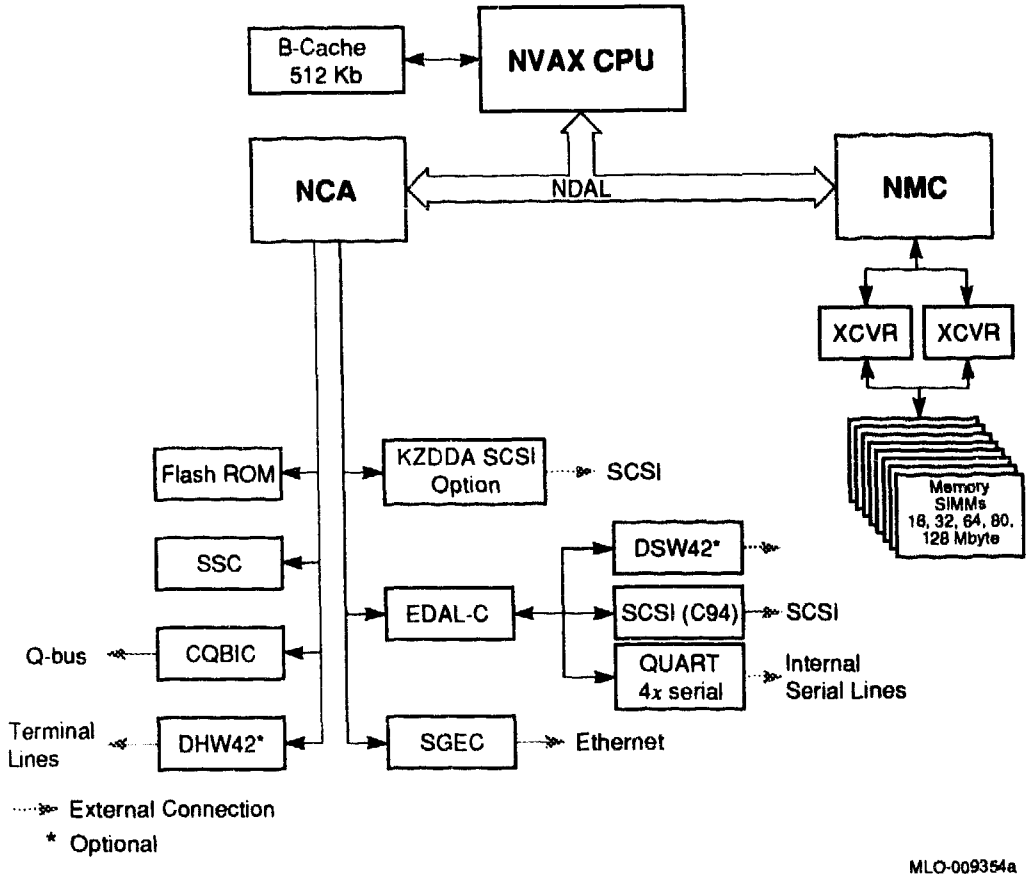
- DC244 NVAX memory controller (NMC) memory controller chip
- DC243 NVAX CP bus adapter (NMA) and input/output (I/O) control chip
- SCSI controller and SQWF buffer chip
- Time-of-year (TOY) clock SSC chip

- DC541 SGEC chip Ethernet controller for standard or ThinWire Ethernet
- DC7085 (QUART) serial line controller (4 serial lines, one with modem control)
- 128K bytes (KA50/55) or 512K bytes (KA51/56) of second level write-back cache memory
- Basic system memory (16M bytes of **random-access memory** (RAM) consisting of four MS44L-AA memory modules or 64M bytes of RAM consisting of four MS44-CA)
- Support for up to 128M bytes of RAM
- 512K bytes of **read-only memory** (ROM)—This ROM contains the boot and diagnostic firmware for the system.
- 32-byte network address ROM
- Four asynchronous communications ports as follows:
  - Three DEC423 ports—These ports are **modified modular jack** (MMJ) connectors.
  - One modem control port—This port is a D-sub 25-way connector.
- Provision for asynchronous communications options that provide one of the following:
  - Eight or 16 additional DEC423 ports
  - Eight additional modem ports
- Provision for synchronous communications options that provide:
  - Two synchronous ports

1.1.2 Functional Description

Figure 1–2 is block diagram of the CPU module. This example shows a KA51 and KA56. The diagrams for the KA50 and KA55 are the same except that there are only 128Kb of B-cache on those modules instead of the 512Kb shown.

Figure 1–2 KA50/51/55/56 CPU Module Block Diagram



The KA50/51/55/56 CPU module supports the following MicroVAX data types:

- Byte, word, longword, and quadword
- Character string
- Variable-length bit field
- Absolute queues
- Self-relative queues
- f_floating-point, d_floating-point, and g_floating-point

The operating system uses software emulation to support other MicroVAX data types. The KA50/51/55/56 CPU module supports the following MicroVAX instructions:

- Integer, arithmetic and logical
- Address
- Variable-length bit field
- Control
- Procedure call
- Miscellaneous
- Queue
- Character string instructions
- MOVC3/MOVC5
- CMPC3/CMPC5
- LOCC
- SCANC
- SKPC
- SPANC
- Operating system support
- f_floating-point, d_floating-point, and g_floating-point

The NVAX processor chip provides special microcode assistance to aid the macrocode emulation of the following instruction groups:

- Character string (other than those mentioned previously)
- Decimal string
- CRC
- EDITPC

The operating system uses software emulation to support other VAX instructions. Figure 1–3 shows the controls, indicators, ports, and connectors on the KA50/51/55/56 CPU module. Table 1–1 describes the functions of the controls, indicators, ports, and connectors.

Figure 1–3 KA50/51/55/56 Controls, Indicators, Ports, and Connectors

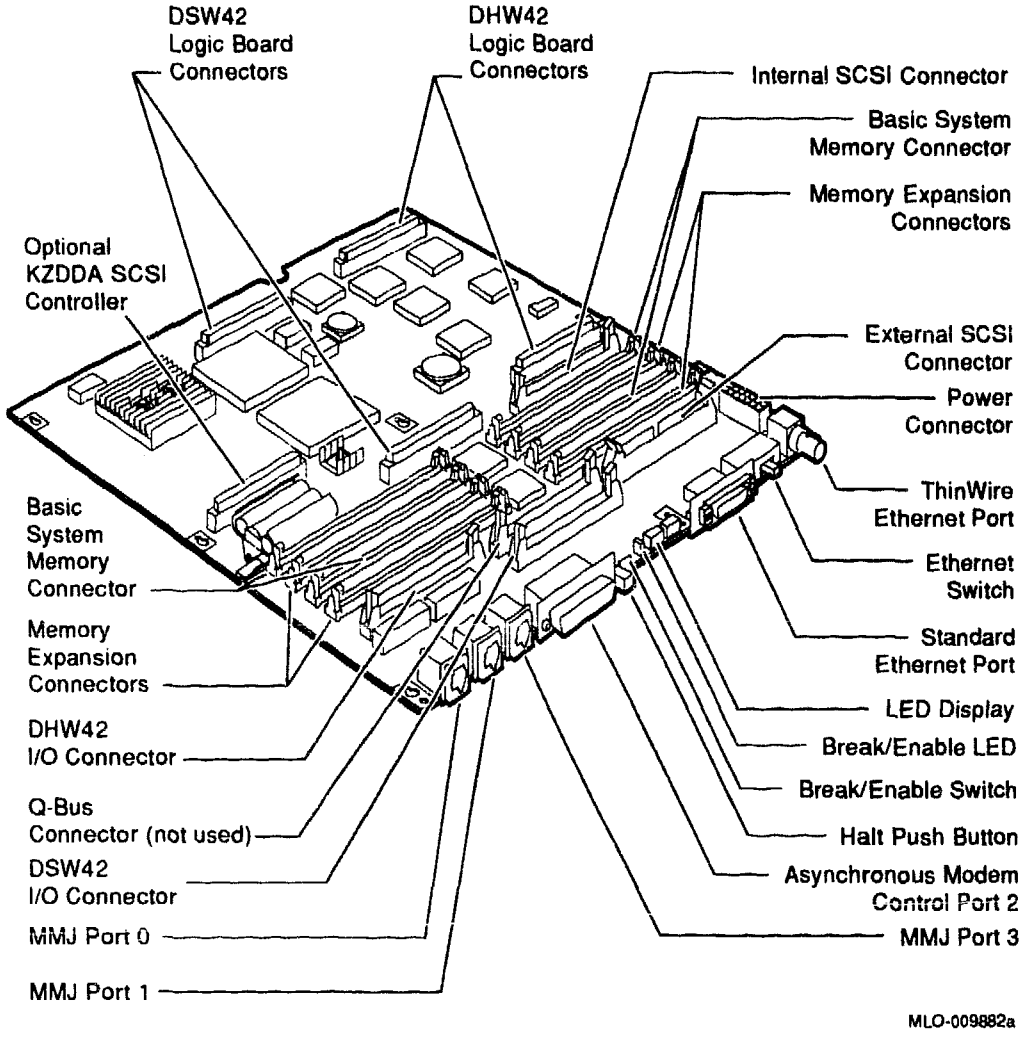


Table 1–1 Functions of Controls, Indicators, Connectors

Component	Description
Internal SCSI connector	A connector that provides a connection for SCSI devices mounted inside the system enclosure.

(continued on next page)

Table 1–1 (Cont.) Functions of Controls, Indicators, Connectors

Component	Description
Basic system memory connectors	Four connectors for the basic system memory modules.
Memory expansion connectors	Four connectors for an additional memory option.
External SCSI connector	A connector that provides a connection to SCSI devices that are external to the system enclosure. (Only functional when the *internal* SCSI connector has a cable installed.)
Power connector	A connector for dc power.
ThinWire Ethernet port	A port that provides a connection to a ThinWire Ethernet network.
Ethernet switch	A two-position switch that determines the type of Ethernet that the system uses as follows: • Left position—selects the standard Ethernet type • Right position—selects the ThinWire Ethernet type
Standard Ethernet port	A port that provides a connection to a standard Ethernet network.
LED display	A set of six LEDs that provide power-up and self-test diagnostic code information.
Break/Enable LED	A LED indicator that shows the function of MMJ port 3 as follows: • On—Break enable • Off—Break disable on port 3
Break/Enable switch[1]	A two-position switch that determines the function of MMJ port 3 as follows: • Up position—MMJ port 3 functions as a console port; in this state, you can press the Break key on the keyboard of a terminal connected to MMJ port 3 to put the system in console mode. • Down position—MMJ port 3 functions as a console port only, and the Break key is disabled.

[1]The system recognizes the position of this switch only when the system is turned on.

(continued on next page)

Table 1–1 (Cont.) Functions of Controls, Indicators, Connectors

Component	Description
Halt button	A momentary-contact push button that puts the system in console mode.
Asynchronous modem control port 2	EIA-232 compatible asynchronous port with modem control.
MMJ port 3	DEC423 compatible asynchronous port. This port functions as the primary console port.
MMJ port 1	DEC423 compatible asynchronous port.
MMJ port 0	DEC423 compatible asynchronous port.
DSW42 I/O connector	A connector that provides a connection for the DSW42 input/output cable.
DHW42 I/O connector	A connector that provides a connection for the DHW42 input/output cable.
DSW42 logic board connectors	Two connectors that provide connections for a DSW42 logic board.
DHW42 logic board connectors	Two connectors that provide connections for a DHW42 logic board.
KZDDA SCSI connector option	Connector which provides a physical interface between the CPU module and external SCSI devices on an optional second SCSI bus (SCSI-B).

1.2 MS44 and MS44L Memory Modules

The MS44 and the MS44L memory modules provide memory expansion for the KA50/51/55/56 CPU module. The KA50/51/55/56 CPU module supports one variant of the MS44 memory option and one variant of the MS44L option as follows:

- The MS44L-BC (16M bytes), which contains four MS44L-AA (4M bytes) memory modules
- The MS44-DC (64M bytes), which contains four MS44-CA (16M bytes) memory modules

Note

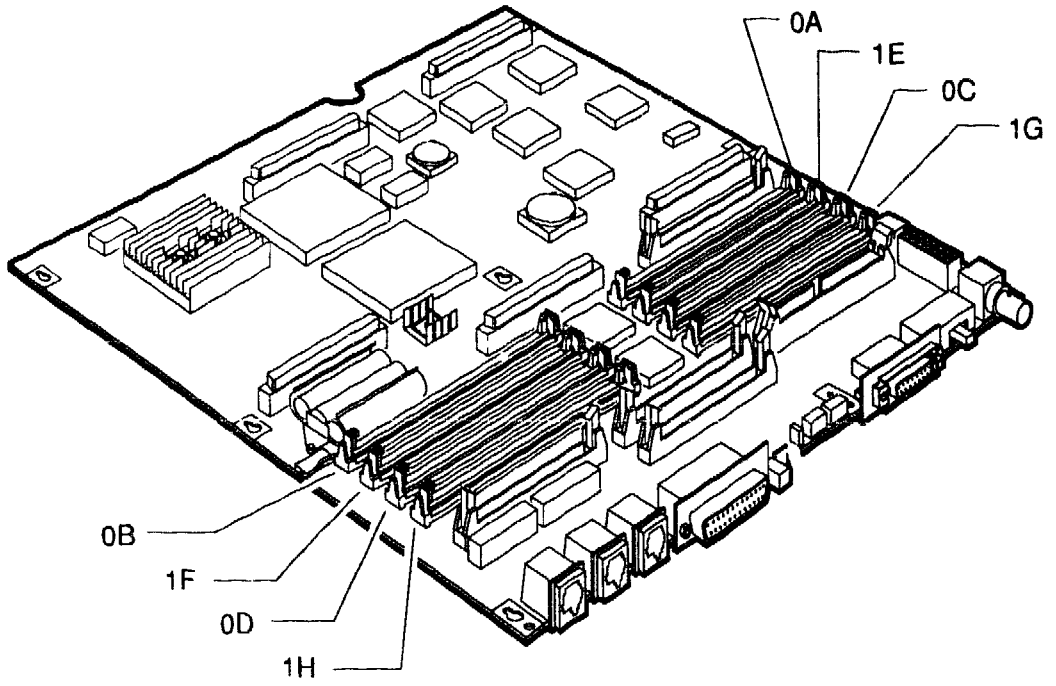
Use only MS44 or MS44L memory modules qualified by Digital.

The rules for adding MS44 or MS44L memory options are as follows:

- You must install all four of the memory modules contained in a memory option. This means that you can expand memory in 16M byte or 64M byte increments only.
- You can install memory options only in a set of connectors that have the same numeral in the connector label. The sets are identified by the following labels:
  - 0A, 0B, 0C, 0D
  - 1E, 1F, 1G, 1H

Figure 1–4 shows the location of the basic memory (16M bytes or 64M bytes) and the memory expansion connectors. Table 1–2 lists the memory configurations.

Figure 1–4 Memory Expansion Connectors



Note: 0A 0B 0C and 0D are identifiers for the basic system memory connectors.

GA_EN00083A_92A

Table 1–2 KA50/51/55/56 CPU Module Memory Configurations

Total Memory (bytes)	Increment 1[1] (0A + 0B + 0C + 0D)[2]	Increment 2 (1E + 1F + 1G + 1H)[2]
16M	MS44L-BC	
32M	MS44L-BC	MS44L-BC
64M	MS44-DC	
80M	MS44-DC	MS44L-BC
128M	MS44-DC	MS44-DC

[1]Basic system memory.

[2]0A, 0B, 0C, 0D, 1E, 1F, 1G, and 1H are connector identifiers (see Figure 1–4).

1.3 MS44 or MS44L Memory Option Installation

The MS44 and MS44L memory options consist of four memory modules each. Install an MS44 or MS44L memory option on the KA50/51/55/56 CPU module as follows:

1. Position the KA50/51/55/56 CPU module, component side up, so that the edge connectors are facing away from you.
2. Identify the connectors on the KA50/51/55/56 CPU module into which you must install the memory option (see Figure 1–4 and Table 1–2).
3. Insert the first memory module, with the side containing the bar code facing away from you, into the connector on the KA50/51/55/56 CPU module (see Figure 1–5).

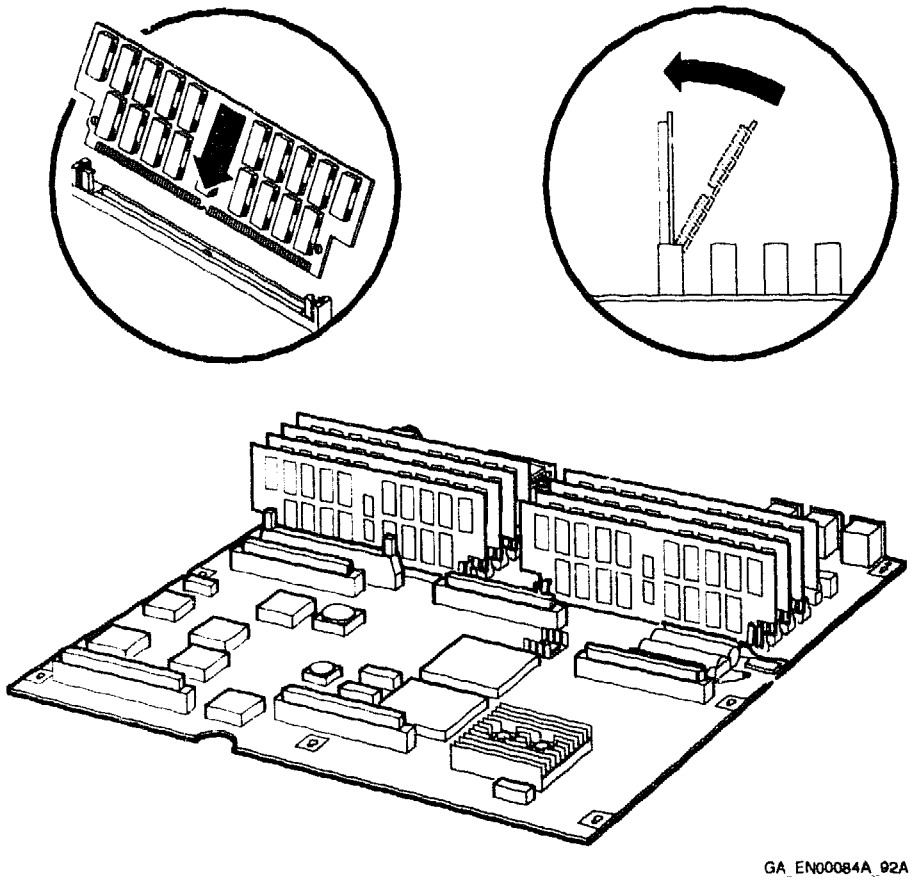
Caution

The connectors are keyed to ensure that you install the memory modules with the correct orientation. Do not force the modules into the connectors with an incorrect orientation.

Caution

Make sure that you fully install the memory module into the connector before you tilt the module toward the front of the enclosure.

Figure 1–5 Memory Module Installation



4. Tilt the memory module toward the front of the enclosure until the metal locking clips on the connector lock the memory module in position.
5. Repeat the procedure in step 1 for the subsequent memory modules. Insert them into the other connectors in the set on the KA50/51/55/56 CPU module.
6. Run the MEM diagnostic test, refer to Section 1.4 after you reinstall the KA50/51/55/56 CPU module into the system enclosure to check that the memory is working correctly.

Caution

When removing memory modules, you must release the metal clips on the connectors of the CPU module.

1.4 Memory Tests

The memory tests check the system memory contained on the MS44 and/or MS44L memories. The tests run automatically as part of the power-up tests and initialization, when you turn on the system. The memory tests are a group of individual tests which can be called individually or normally as a group under a specific script number.

The recommended method to verify a new memory installation is to run the memory test script A8 which will call all of the memory tests and run them on all memory present.

Examples of successful and unsuccessful runs of memory test script A8 are shown in Example 1–1 and Example 1–2.

The individual memory tests are listed following the examples.

Example 1–1 Successful Running of Memory Test Script A8

```
>>>T A8
9D..31..30..4F..4E..4D..4C..4B..4A..48..48..48..48..48..48..48..
48..48..48..47..40..80..
>>>
```

The failure is reported by the count bad pages test 40 at end of the script. Issuing the SHOW MEMORY command shows which memory set caused the failure. Bad pages were detected in memory set 0.

```
>>>SHOW MEMORY

16 MB RAM, SIMM Set (0A,0B,0C,0D) present
Memory Set 0: 00000000 to 00FFFFFF, 16MB, 32256 good pages, 512 bad pages

16 MB RAM, SIMM Set (1E,1F,1G,1H) present
Memory Set 1: 01000000 to 01FFFFFF, 16MB, 32768 good pages, 0 bad pages

Total of 32MB, 65024 good pages, 512 bad pages, 112 reserved pages
>>>
```

Example 1–2 Typical Failure After Running Memory Test Script A8

```
>>>T A8
9D..31..30..4F..4E..4D..4C..4B..4A..48..48..48..48..48..48..48..
48..48..48..47..40..

? Test_Subtest_40_06  Loop_Subtest=00  Err_Type=FF  DE_Memory_count_pages.lis
Vec=0000  Prev_Errs=0004  P1=00000001  P2=00000002  P3=00000001
P4=00000000 P5=00000020  P6=00008000  P7=00000020  P8=00000000
P9=00000000 P10=00FCD44B r0=00FF4008  r1=00000007  r2=00000000
r3=FFFFFFFF  r4=00000068  r5=00000000 r6=00000000  r7=00000002
r8=00FF4000  r9=20140758  r10=FFFFFFFE r11=FFFFFFFF
dser=0000 cesr=00000000 intmsk=00 icsr=01 pcsts=FC00 pcadr=FFFFFFF8 pcctl=FC13
cctl=00000021 bcetsts=0000 bcedsts=0000 cefsts=00000200 nests=00
mmcdsr=01111000 mesr=00080000
```

Test DC - Check for No Memory Present

The only purpose of this test is to check for the specific condition of no valid memory present in the system. This occurs if no memory is present, or if memory is present and one or more SIMMs is missing or not plugged in correctly.

Test 31 - Size and Setup Memory CSRs

Find out how much memory is available and configure into consecutive memory starting at address 00000000. Verify proper configuration data in the CSRs.

Test 30 - Build a Bitmap in Memory

Set up a bitmap in RAM to be used by the memory tests. Test the area before setting up a bitmap.

This test looks for a 1 MB KB section of memory to be used for the bitmap, busmap and reserved console area and structures to run diagnostics. The test starts at the top of available memory and tests one section of memory at the top of each 4 MB section of memory until a good section is found for the maps or the bottom of memory is reached, in which case the test fails.

Test 4F - Data Pattern Tests

Verifies that each bit in the data path can be written to a one and a zero individually. This test also checks for shorts between individual paths. The test needs to be run once for each array of memory chips.

This test uses various fix patterns and also floating 1's and 0's patterns across all 72 data bits (64 data, 8 ECC). The test always checks both even and odd QWs of data so that all four SIMMs in a memory set are tested.

Test 4E - Masked Write Cycles with No Errors, BYTE, WORD

This test verifies masked write cycles to memory.

Test 4D - Address Uniqueness Test

The main purpose of the test is to verify that each set on each board can be uniquely addressed. The test writes a unique pattern to each location to be tested then verifies all locations.

Test 4C - MEMORY ECC, Verify Error Detection and Reporting

The main purpose of this test is to test ECC logic. It is not intended to test the memory RAMs explicitly.

The test verifies that single and double bit errors are reported and logged correctly in the MESR. It also verifies that single bit errors cause interrupts through vector 54 when enabled and that double bit errors cause a machine check.

In addition, the test also verifies that multiple bit errors can be detected using data patterns that generate all of the syndrome values for multiple bit errors.

Test 4B - MEMORY Verify Masked Write Cycles with Errors

The test verifies operation of masked write cycles when the location contains errors. In addition, it verifies that errors are reported and that single bit errors are corrected.

Test 4A - MEMORY ECC, Verify Ability to Correct Single Bit Errors

This test verifies the correct operation of the error correction logic (ECC). It does this by verifying that single bit errors can be detected and corrected in any of the 64 data bits and that single bit errors are detected in the eight check bits.

Test 48 - MEMORY Address/Shorts Test

This test verifies that all locations in each set can be uniquely written to and that each of the 64 data bits in each QW can be written to a one and to a zero. This test also writes all locations in memory with good ECC.

The test runs on a hexaword basis with all caches enabled to fully utilize caching to speed up the test. Two primary data patterns of AAAAAAAA_AAAAAAAA and 55555555_55555555 are used by the test. The ECC checkbits for these patterns are complements of each other. By running this test, all data and ECC bits in all locations in memory will be written as a 1 and a 0. The test also detects addressing errors.

Test 47 - MEMORY Data Retention, Verify Refresh Logic

This test verifies that the refresh logic is working for all memory boards. The test loads patterns into memory, waits a specified amount of time, then verifies the patterns.

Test 40 - MEMORY Count Bad Pages Marked In Bitmap

This test is normally run last in a script of memory tests. Its only purpose is to read the bitmap when done and check to see if any pages in memory were marked bad, if so, report an error.

Note

If this test fails, do SHOW MEMORY to see which set has bad pages in it.

2
Configuration

This chapter describes the KA50/51/55/56 system configurations. It gives information on the following:

- Memory configurations
- Mass storage devices
- Communications options

2.1 Memory Configurations

A KA50/51/55/56 system has a basic memory of 16M bytes or 64M bytes. This consists of four MS44L-AA memory modules or four MS44-CA memory modules. You can add memory in 16M byte or 64M byte increments, up to a maximum of 128M bytes. See Section 1.2 for information on the memory configurations.

2.2 Mass Storage Devices

A KA50/51/55/56 system supports mass storage devices in the following categories:

- Internal mass storage devices—These devices are mounted inside the system enclosure.
- External mass storage devices—These devices are self-contained units that you can connect to the system externally.

2.2.1 Internal Mass Storage Devices

Table 2–1 shows some of the internal mass storage devices that a KA50/51/55 /56 system supports.

Table 2–1 KA50/51/55/56 Internal Mass Storage Devices

Option Name	Description	Size[1] (in)	Capacity
RZ23L	Disk drive	3.5	120-MB
RZ24	Disk drive	3.5	209-MB
RZ24L	Disk drive	3.5	245-MB
RZ25	Disk drive	3.5	400-MB
RZ25L	Disk drive	3.5	535-MB
RZ25M	Disk drive	3.5	545-MB
RZ26	Disk drive	3.5	1.05-GB
RZ26L	Disk drive	3.5	1.05-GB
RZ28	Disk drive	3.5	2.10-GB
TZ30[2]	Tape drive	5.25	95-MB cartridge
TZK10/TZK11[2]	Tape drive	5.25	Range of cartridges
TLZ06/TLZ07[2]	Tape drive	5.25	Range of cassettes
RX23/RX26[2]	Diskette drive	3.5	Range of diskettes
RRD42[2]	CDROM drive	5.25	600-MB CDROM
RRD43[2]	CDROM drive	5.25	600-MB CDROM

[1]Size of half-height device.

[2]Removable media device.

The system enclosure determines the combinations of internal mass storage devices in a KA50/51/55/56 system. See the *MicroVAX 3100 BA42B Enclosure Maintenance* manual for more information.

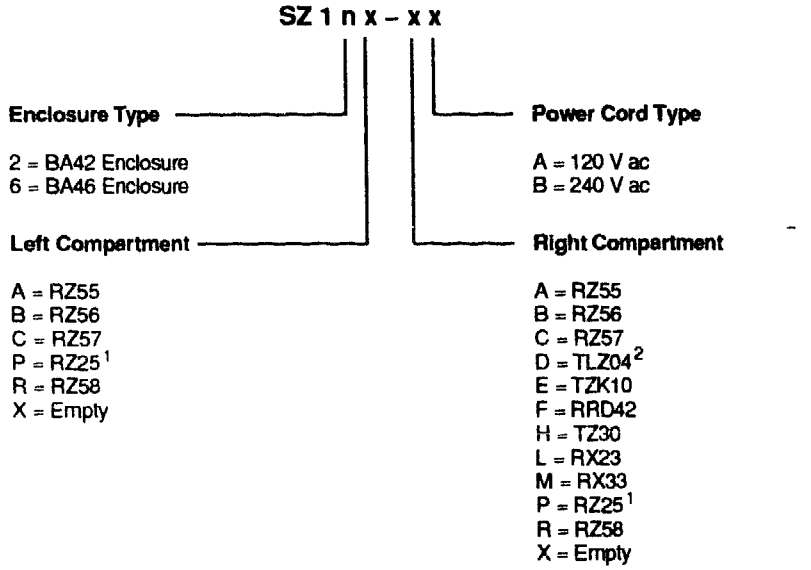
2.2.2 External Mass Storage Devices

The external mass storage devices connect to KA50/51/55/56 systems through the SCSI connector on the back of the system enclosure. In KA50/51/55/56 systems, the SCSI bus supports a maximum of seven mass storage devices. Therefore, the number of external mass storage devices that you can connect depends on the number of mass storage devices that are mounted inside the system enclosure.

The maximum number of mass storage devices in the system enclosure is five. This means that you can connect at least two external mass storage devices.

A KA50/51/55/56 system supports the SZ series of mass storage expansion boxes. The SZ number defines the contents of each expansion box. Figure 2–1 shows the numbering system for SZ expansion boxes.

Figure 2–1 SZ Expansion Box Numbering System



[1] The RZ25 disk drive fits in the BA42 enclosure only.
[2] The TLZ04 tape drive fits in the BA46 enclosure only.

With the KZDDA SCSI option, a second SCSI connector, a KA50/51/55/56 system can support seven additional external devices on a second (external) SCSI bus.

A KA50/51/55/56 system also supports other types of external mass storage devices. See the latest Systems and Options Catalog (SOC) for a listing of supported external SCSI devices.

When you are adding mass storage devices, use these guidelines. Also, refer to documentation for your SCSI expander, if any.

- You can add a maximum of four external SCSI devices. A fully configured SZ12 enclosure contains two SCSI devices.
- You can add a maximum of two SCSI tape devices. Depending on the configuration, the system may support two TLZ04 tape drives.

- The BA40 single drive expansion box contains one SCSI device.
- The RRD42 CDROM drive is a single SCSI device. You can add a maximum of three RRD42 CDROM drives.
- Terminate the SCSI bus correctly. Failure to do this can cause a system failure or corrupt data.
- Digital recommends that you connect all SCSI devices to the same ac power source.
- Do not add or remove devices that are connected to the SCSI bus while the power is on.
- Digital does not guarantee the correct operation of a SCSI bus that does not use the cables supplied by Digital or is not configured in accordance with Digital recommendations.

2.2.3 SCSI ID Numbers

Each mass storage device must have a unique SCSI ID number. SCSI ID 6 is typically used for the SCSI controller.

2.3 Communications Options

A KA50/51/55/56 system supports the following types of communications options:

- Asynchronous communications options
- Synchronous communications options

Each communications option has components that are installed in the system enclosure and components that connect to the system externally.

2.3.1 Asynchronous Communications Options

Table 2–2 lists the asynchronous communications options that KA50/51/55/56 systems support.

Table 2–2 Supported Asynchronous Communications Options

Option	Description
DHW42-AA	Eight-line DEC423 asynchronous option
DHW42-BA	Sixteen-line DEC423 asynchronous module option
DHW42-CA	Eight-line EIA-232 modem asynchronous module option
DHW42-UP	Eight-line to 16-line DEC423 asynchronous upgrade option

2.3.2 Synchronous Communications Options

Table 2–3 lists the synchronous communications options that KA50/51/55/56 systems support.

Table 2–3 Supported Synchronous Communications Options

Option	Description
Model 100	
DSW42-AA[1]	Two-line EIA-232/V.24 synchronous option with two external cables, BC19D-02 (17-01110-01)

[1]This option is supplied with two external cables that support the EIA-232/V.24 interface.

The the DSW42-AA option also supports the communications interfaces listed in Table 2–4, but you must order the external cables separately.

Table 2–4 DSW42-AA Communications Support

Communications Interface	External Cable
EIA-423/V.10	BC19E-02[1] (17-01111-01)
EIA-422/V.11	BC19B-02[1] (17-01108-01)

[1]Two required for DSW42-AA.

3

KA50/51/55/56 Firmware Commands

This chapt r describes the console mode control characters, the command syntax, the command modifiers, and all of the console commands. You can enter these commands when the system is in console mode. Console mode is indicated when the console prompt (>>>) is displayed. If the system is running the operating system software, refer to the *MicroVAX 3100 Model 85 Customer Technical Information* manual, the *MicroVAX 3100 Model 90 Customer Technical Information* manual, the *MicroVAX 3100 Model 95 Customer Technical Information* manual, or the *MicroVAX 3100 Model 96 Customer Technical Information* manual, for information on returning the system to console mode.

If the console security feature is enabled and a security password is set, you must log in to privileged console mode before using most of these commands. Refer to the appropriate MicroVAX 3100 Customer Technical Information manual (above) for information on the console security feature.

Note

The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95, and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, will appear on the console and/or printouts from time to time.

3.1 Console I/O Mode Control Characters

In console I/O mode, several characters have special meaning:

RETURN	Also <CR>. The carriage return ends a command line. No action is taken on a command until after it is terminated by a carriage return. A null line terminated by a carriage return is treated as a valid, null command. No action is taken, and the console prompts for input. Carriage return is echoed as carriage return, line feed (<CR><LF>).
<X	When you press <X, the console deletes the previously typed character. The resulting display differs, depending on whether the console is a video or a hardcopy terminal. For hardcopy terminals, the console echoes a backslash (\), followed by the deletion of the character. If you press additional rubouts, the additional deleted characters are echoed. If you type a nonrubout character, the console echoes another backslash, followed by the character typed. The result is to echo the characters deleted, surrounding them with backslashes. For example: EXAMI;E <X<XNE<CR> The console echoes: EXAMI;E\E;\NE<CR> The console sees the command line: EXAMINE<CR> For video terminals, the previous character is erased and the cursor is restored to its previous position. The console does not delete characters past the beginning of a command line. If you press more rubouts than there are characters on the line, the extra rubouts are ignored. A rubout entered on a blank line is ignored.
CTRL/A and F14	Toggle insertion/overstrike mode for command line editing. By default, the console powers up to overstrike mode.
CTRL/B or up_ arrow (or down_ arrow)	Recalls previous command(s). Command recall is only operable if sufficient memory is available. This function may then be enabled and disabled using the SET RECALL command.
CTRL/D and left arrow	Move cursor left one position.
CTRL/E	Moves cursor to the end of the line.
CTRL/F and right arrow	Move cursor right one position.
CTRL/H, backspace, and F12	Move cursor to the beginning of the line.
CTRL/U	Echoes ^U<CR> and deletes the entire line. Entered but otherwise ignored if typed on an empty line.
CTRL/S	Stops output to the console terminal until CTRL/Q is typed Not echoed.
CTRL/Q	Resumes output to the console terminal. Not echoed.

CTRL/R	Echoes <CR><LF>, followed by the current command line. Can be used to improve the readability of a command line that has been heavily edited.
CTRL/C	Echoes ^C<CR> and aborts processing of a command. When entered as part of a command line, deletes the line.
CTRL/O	Ignores transmissions to the console terminal until the next CTRL/O is entered. Echoes ^O when disabling output, not echoed when it re-enables output. Output is re-enabled if the console prints an error message, or if it prompts for a command from the terminal. Output is also enabled by entering console I/O mode, by pressing the BREAK key, and by pressing CTRL/C.

3.1.1 Command Syntax

The console accepts commands up to 80 characters long. Longer commands produce error messages. The character count does not include rubouts, rubbed-out characters, or the RETURN at the end of the command.

You can abbreviate a command by entering only as many characters as are required to make the command unique. Most commands can be recognized from their first character. See Table 3–5.

The console treats two or more consecutive spaces and tabs as a single space. Leading and trailing spaces and tabs are ignored. You can place command qualifiers after the command keyword or after any symbol or number in the command.

All numbers (addresses, data, counts) are hexadecimal (hex), but symbolic register names contain decimal register numbers. The hex digits are 0 through 9 and A through F. You can use uppercase and lowercase letters in hex numbers (A through F) and commands.

The following symbols are qualifier and argument conventions:

[ ] An optional qualifier or argument

{ } A required qualifier or argument

3.1.2 Address Specifiers

Several commands take one or more addresses as arguments. An address defines the address space and the offset into that space. The console supports five address spaces:

Physical memory
Virtual memory
General purpose registers (GPRs)
Internal processor registers (IPRs)
The PSL

The address space that the console references is inherited from the previous console reference, unless you explicitly specify another address space. The initial address space is physical memory.

3.1.3 Symbolic Addresses

The console supports symbolic references to addresses. A symbolic reference defines the address space and the offset into that space. Table 3–1 lists symbolic references supported by the console, grouped according to address space. You do not have to use an address space qualifier when using a symbolic address.

Table 3–1 Console Symbolic Addresses

Symb	Addr	Symb	Addr	Symb	Addr	Symb	Addr
/G—General Purpose Registers							
R0	00	R4	04	R8	08	R12 (AP)	0C
R1	01	R5	05	R9	09	R13 (FP)	0D
R2	02	R6	06	R10	0A	R14 (SP)	0E
R3	03	R7	07	R11	0B	R15 (PC)	0F
/M—Processor Status Longword							
PSL	—						
/I—Internal Processor Registers							
pr$_ksp	00	pr$_pcbb	10	pr$_rxcs	20	—	30
pr$_esp	01	pr$_scbb	11	pr$_rxdb	21	—	31
pr$_ssp	02	pr$_ipl	12	pr$_txcs	22	—	32
pr$_usp	03	pr$_astlv	13	pr$_txdb	23	—	33
pr$_isp	04	pr$_sirr	14	—	24	—	34
—	05	pr$_sisr	15	—	25	—	35
—	06	—	16	pr$_mcesr	26	—	36
—	07	—	17	—	27	pr$_ ioreset	37

Note: All symbolic values in this table are in hexadecimal.

(continued on next page)

Table 3–1 (Cont.) Console Symbolic Addresses

Symb	Addr	Symb	Addr	Symb	Addr	Symb	Addr
			/i—Internal Processor Registers				
pr$_p0br	08	pr$_iccs	18	—	28	pr$_ mapen	38
pr$_p0lr	09	pr$_nicr	19	—	29	pr$_tbia	39
pr$_p1br	0A	pr$_icr	1A	pr$_savpc	2A	pr$_tbis	3A
pr$_p1lr	0B	pr$_todr	1B	pr$_savpsl	2B	—	3B
pr$_sbr	0C	—	1C	—	2C	—	3C
pr$_slr	0D	—	1D	—	2D	—	3D
—	0E	—	1E	—	2E	pr$_sid	3E
—	0F	—	1F	—	2F	pr$_ tbchk	3F
pr$_ecr	7D						
pr$_cctl	A0	pr$_neoadr	B0	pr$_vmar	D0	—	F0
—	A1	—	B1	pr$_vtag	D1	—	F1
pr$_bcdecc	A2	pr$_ neocmd	B2	pr$_vdata	D2	pr$_pcadr	F2
pr$_bcetsts	A3	—	B3	pr$_icsr	D3	—	F3
pr$_bcetidx	A4	pr$_ nedathi	B4	—	D4	pr$_pcsts	F4
pr$_bcetag	A5	—	B5	—	D5	—	F5
pr$_ bcedsts	A6	pr$_ nedatlo	B6	—	D6	—	F6
pr$_ bcedidx	A7	—	B7	pr$_ pamode	E7	—	F7
pr$_ bcedecc	A8	pr$_neicmd	B8	—	E8	pr$_pcctl	F8
pr$_cefadr	AB	—	B9	—	E9	—	F9
pr$_cefsts	AC	—	BA	pr$_tbadr	EC	—	FA
pr$_nests	AE	—	BB	pr$_tbsts	ED	—	FB
pr$_bctag	01000000	pr$_bcflush	01400000	pr$_pctag	01800000	pr$_ pcdap	01C00000

(continued on next page)

Table 3–1 (Cont.) Console Symbolic Addresses

Symb	Addr	Symb	Addr	Symb	Addr	Symb	Addr
			/P—Physical (VAX I/O Space)				
qbio	20000000	qbmem	30000000	qbmbr	20080010	—	—
rom or feprom	20040000	—	—	bdr	20084000	—	—
scr	20080000	dser	20080004	qbear	20080008	dear	2008000C
ipcr0	20001f40	ipcr1	20001f42	ipcr2	20001f44	ipcr3	20001f46
sscram/ nvr	20140400	ssccr	20140010	cbtcr	20140020	dledr	20140030
ad0mat	20140130	ad0msk	20140134	ad1mat	20140140	ad1msk	20140144
tcr0	20140100	tir0	20140104	tnir0	20140108	tivr0	2014010c
tcr1	20140110	tir1	20140114	tnir1	20140118	tivr1	2014011c
nicsr0	20008000	nicsr1	20008004	nicsr2	20008008	nicsr3	2000800C
nicsr4	20008010	nicsr5	20008014	nicsr6	20008018	nicsr7	2000801C
—	20008020	nicsr9	20008024	nicsr10	20008028	nicsr11	2000802C
nicsr12	20008030	nicsr13	20008034	nicsr14	20008038	nicsr15	2000803C
sgec_setup	20008000	sgec_txpoll	20008004	sgec_rxpoll	20008008	sgec_rba	2000800C
sgec_tba	20008010	sgec_status	20008014	sgec_mode	20008018	sgec_sbr	2000801C
—	20008020	sgec_wdt	20008024	sgec_mfc	20008028	sgec_ verlo	2000802C
sgec_verhi	20008030	sgec_proc	20008034	sgec_bpt	20008038	sgec_cmd	2000803C
shac_sswcr	20004230	shac_ sshma	20004244	shac_pqbbr	20004248	shac_psr	2000424c
shac_pesr	20004250	shac_pfar	20004254	shac_ppr	20004258	shac_ pmcsr	2000425C
shac_ pcq0cr	20004280	shac_ pcq1cr	20004284	shac_ pcq2cr	20004288	shac_ pcq3cr	2000428C
shac_ pdfqcr	20004290	shac_ pmfqcr	20004294	shac_psrcr	20004298	shac_pecr	2000429C
shac_pdcr	200042A0	shac_picr	200042A4	shac_pmtcr	200042A8	shac_ pmtecr	200042AC

(continued on next page)

Table 3–1 (Cont.) Console Symbolic Addresses

Symb	Addr	Symb	Addr	Symb	Addr	Symb	Addr
			/P—Physical (VAX I/O Space)				
nmccwb	21000110	modr	21010000	—	—	—	—
memcon0	21018000	memcon1	21018004	memcon2	21018008	memcon3	2101800c
memcon4	21018010	memcon5	21018014	memcon6	21018018	memcon7	2101801c
memsig8	21018020	memsig9	21018024	memsig10	21018028	memsig11	2101802c
memsig12	21018030	memsig13	21018034	memsig14	21018038	memsig15	2101803c
mear	21018040	mser	21018044	nmcdsr	21018048	moamr	2101804C
cesr	21020000	cmcdsr	21020004	csear1	21020008	csear2	2102000c
cioear1	21020010	cioear2	21020014	cnear	21020018	—	—
scdadrB	21C00000	scddirB	21C00004	scsicsr0B	22000080	scsicsr1B	22000084
scsicsr2B	22000088	scsicsr3B	2200008c	scsicsr4B	22000090	scsicsr5B	22000094
scsicsr6B	22000098	scsicsr7B	2200009c	scsicsr8B	22000A0	scsicsr9B	220000A4
scsicsraB	220000A8	scsicsrbB	220000Ac	scsicsrcB	220000B0	scsimapB	23000000
intmskB	21C00008	intreqB	21C0000c	—	—	—	—
csr	25000000	rbuf	25000004	lpr	25000004	tcr	26000008
msr	2500000C	tdr	2500000C	ssr	25800000	—	—
scdadr	25C00000	scddir	25C00004	intmsk	25C00008	intreq	25C0000C
scsicsr0	26000080	scsicsr1	26000084	scsicsr2	26000088	scsicsr3	2600008C
scsicsr4	26000090	scsicsr5	26000094	scsicsr6	25c00098	scsicsr7	25C0009C
scsicsr8	260000A0	scsicsr9	260000A4	scsicsra	260000A8	scsicsrb	260000AC
scsicsrc	260000BU	scsimap	27000000	—	—	—	—

Table 3–2 lists symbolic addresses that you can use in any address space.

Table 3–2 Symbolic Addresses Used In Any Address Space

Symbol	Description
*	The location last referenced in an EXAMINE or DEPOSIT command.
+	The location immediately following the last location referenced in an EXAMINE or DEPOSIT command. For references to physical or virtual memory spaces, the location referenced is the last address, plus the size of the last reference (1 for byte, 2 for word, 4 for longword, 8 for quadword). For other address spaces, the address is the last address referenced plus one.
-	The location immediately preceding the last location referenced in an EXAMINE or DEPOSIT command. For references to physical or virtual memory spaces, the location referenced is the last address minus the size of this reference (1 for byte, 2 for word, 4 for longword, 8 for quadword). For other address spaces, the address is the last address referenced minus one.
@	The location addressed by the last location referenced in an EXAMINE or DEPOSIT command.

3.1.4 Console Numeric Expression Radix Specifiers

By default, the console treats any numeric expression used as an address or a datum as a hexadecimal integer. The user may override the default radix by using one of the specifiers listed in Table 3–3.

Table 3–3 Console Radix Specifiers

Form 1	Form 2	Radix
%b	^b	Binary
%o	^o	Octal
%d	^d	Decimal
%x	^x	Hexadecimal, default

For instance, the value 19 is by default hexadecimal, but it may also be represented as %b11001, %o31, %d25, and %x19 (or in the alternate form as ^b11001, ^o31, ^d25, and ^x19).

3.1.5 Console Command Qualifiers

You can enter console command qualifiers in any order on the command line after the command keyword. The three types of qualifiers are data control, address space control, and command specific. Table 3–4 lists and describes the data control and address space control qualifiers. Command specific qualifiers are listed in the descriptions of individual commands.

Table 3–4 Console Command Qualifiers

Qualifier	Description
Data Control	
/B	The data size is byte.
/W	The data size is word.
/L	The data size is longword.
/Q	The data size is quadword.
/N:{count}	An unsigned hexadecimal integer that is evaluated into a longword. This qualifier determines the number of additional operations that are to take place on EXAMINE, DEPOSIT, MOVE, and SEARCH commands. An error message appears if the number overflows 32 bits.
/STEP:{size}	Step. Overrides the default increment of the console current reference. Commands that manipulate memory, such as EXAMINE, DEPOSIT, MOVE, and SEARCH, normally increment the console current reference by the size of the data being used.
/WRONG	Wrong. On writes, 3 is used as the value of the ECC bits, which always generates double bit errors. Ignores ECC errors on main memory reads.

(continued on next page)

Table 3–4 (Cont.) Console Command Qualifiers

Qualifier	Description
Address Space Control	
/G	General purpose register (GPR) address space, R0–R15. The data size is always longword.
/I	Internal processor register (IPR) address space. Accessible only by the MTPR and MFPR instructions. The data size is always longword.
/V	Virtual memory address space. All access and protection checking occur. If access to a program running with the current PSL is not allowed, the console issues an error message. Deposits to virtual space cause the PTE<M> bit to be set. If memory mapping is not enabled, virtual addresses are equal to physical addresses. Note that when you examine virtual memory, the address space and address in the response is the physical address of the virtual address.
/P	Physical memory address space.
/M	Processor status longword (PSL) address space. The data size is always longword.
/U	Access to console private memory is allowed. This qualifier also disables virtual address protection checks. On virtual address writes, the PTE<M> bit is not set if the /U qualifier is present. This qualifier is not inherited; it must be respecified on each command.

3.1.6 Console Command Keywords

Table 3–5 lists command keywords by type. Table 3–6 lists the parameters, qualifiers, and arguments for each console command. Parameters, used with the SET and SHOW commands only, are listed in the first column along with the command.

You should not use abbreviations in programs. Although it is possible to abbreviate by using the minimum number of characters required to uniquely identify a command or parameter, these abbreviations may become ambiguous at a later time if an updated version of the firmware contains new commands or parameters.

Table 3–5 Command Keywords by Type

Processor Control	Data Transfer	Console Control
BOOT	DEPOSIT	CONFIGURE
CONTINUE	EXAMINE	FIND
HALT	MOVE	REPEAT
INITIALIZE	SEARCH	SET
NEXT	X	SHOW
START		TEST
		LOGIN
UNJAM		!

Table 3–6 Console Command Summary

Command	Qualifiers	Argument	Other(s)
BOOT	/R5:{boot_flags} /{boot_flags}	[{boot_device}[,{boot_device}]...]	—
CONFIGURE	—	—	—
CONTINUE	—	—	—
DEPOSIT	/B /W /L /Q — /G /I /V /P /M /U /N:{count} /STEP:{size} /WRONG	{address}	{data} [{data}]
EXAMINE	/B /W /L /Q — /G /I /V /P /M /U /N:{count} /STEP:{size} /WRONG /INSTRUCTION	[{address}]	—
FIND	/MEM /RPB	—	—
HALT	—	—	—
HELP	—	—	—
INITIALIZE	—	—	—
LOGIN	—	—	—

(continued on next page)

Table 3–6 (Cont.) Console Command Summary

Command	Qualifiers	Argument	Other(s)
MOVE	/B /W /L /Q — /V /P /U /N:{count} /STEP:{size} /WRONG	{src_address}	{dest_ address}
NEXT	—	[{count}]	—
REPEAT	—	{command}	—
SEARCH	/B /W /L /Q — /V /P /U /N:{count} /STEP:{size} /WRONG /NOT	{start_address}	{pattern} [{mask}]
SET BFLAG	—	{bitmap}	—
SET BOOT	—	[{boot_device}[,{boot_ device}]...	—
SET CONTROLP	—	{0/1}	—
SET HALT	—	{halt_action}	—
SCSI_ID	—	{bus}[1] {id}	—
SET HOST	/DUP /DSSI /BUS:{0/1}	{node_number}	[{task}]
SET HOST	/DUP /UQSSP {/DISK ! /TAPE } /DUP /UQSSP	{controller_number} {csr_address}	[{task}] [{task}]
SET HOST	/MAINTENANCE /UQSSP /SERVICE /MAINTENANCE /UQSSP	{controller_number} {csr_address}	
SET LANGUAGE	—	{language_type}	—
SET RECALL	—	{0/1}	—
SHOW BFL(A)G	—	—	—
SHOW BOOT	—	—	—
SHOW CONTROLP	—	—	—
SHOW DSSI	—	—	—
SHOW HALT	—	—	—
SHOW LANGUAGE	—	—	—

[1]For Open VMS version 1.3 and earlier, only one argument, the id, is used. For later versions, two arguments are accepted; the first refers to the bus, the second to the id; if only one argument is supplied, the system defaults to bus 0, and the argument is taken as the id.

(continued on next page)

Table 3–6 (Cont.) Console Command Summary

Command	Qualifiers	Argument	Other(s)
SHOW MEMORY	/FULL	—	—
SHOW QBUS	—	—	—
SHOW RECALL	—	—	—
SHOW RLV12	—	—	—
SHOW SCSI	—	—	—
SHOW SCSI_ID	—	—	—
SHOW TRANSLATION		{phys_address}	—
SHOW UQSSP	—	—	—
SHOW VERSION	—	—	—
START	—	{address}	—
TEST	—	{test_number}	[{parameters}]
UNJAM	—	—	—
X	—	{address}	{count}

3.2 Console Commands

The following sections describe all the console commands, give the command formats with their qualifiers, and describe the significance of each qualifier.

3.2.1 BOOT

The BOOT command initializes the processor and transfers execution to Virtual Memory Boot (VMB). VMB attempts to boot the operating system from the specified device or list of devices, or from the default boot device if none is specified. The console qualifies the bootstrap operation by passing a boot flags bitmap to VMB in R5.

Format:

BOOT [qualifier-list] [{boot_device},{boot_device},...]

If you do not enter either the qualifier or the device name, the default value is used. Explicitly stating the boot flags or the boot device overrides, but does not permanently change, the corresponding default value.

When specifying a list of boot devices (up to 32 characters, with devices separated by commas and no spaces), the system checks the devices in the order specified and boots from the first one that contains bootable software.

Note

If included in a string of boot devices, the Ethernet device, EZA0, should be placed only as the last device of the string. The system will continuously attempt to boot from EZA0.

Set the default boot device and boot flags with the SET BOOT and SET BFLAG commands. If you do not set a default boot device, the processor times out after 30 seconds and attempts to boot from the Ethernet device, EZA0.

Qualifiers:

Command specific:

/R5:{boot_flags}	A 32-bit hex value passed to VMB in R5. The console does not interpret this value. Use the SET BFLAG command to specify a default boot flags longword. Use the SHOW BFLAG command to display the longword.
/{boot_flags}	Same as /R5:{boot_flags}
[device_name]	A character string of up to 32 characters. When specifying a list of boot devices, the device names should be separated by commas and no spaces. Apart from checking the length, the console does not interpret or validate the device name. The console converts the string to uppercase, then passes VMB a string descriptor to this device name in R0. Use the SET BOOT command to specify a default boot device or list of devices. Use the SHOW BOOT command to display the default boot device. The factory default device is the Ethernet device, EZA0. Refer to the *MicroVAX 3100 Customer Technical Information* manuals for a list of the boot devices supported by the system.

Examples:

```
>>>SHOW BOOT
DKA300
>>>SHOW BFLAG
00000000
>>>B    !Boot using default boot flags and device.
(BOOT/R5:0 DKA300)

  2..
-DKA300
```

3.2.2 CONTINUE

The CONTINUE command causes the processor to begin instruction execution at the address currently contained in the program counter (PC). This address is the address stored in the PC when the system entered console mode or an address that the user specifies using the DEPOSIT command. The CONTINUE command does not perform a processor initialization. The console enters program I/O mode.

Format:

CONTINUE

Example:

```
>>>CONTINUE
$       !OpenVMS DCL prompt
```

3.2.3 DEPOSIT

The DEPOSIT command deposits data into the address specified. If you do not specify an address space or data size qualifier, the console uses the last address space and data size used in a DEPOSIT, EXAMINE, MOVE, or SEARCH command. After processor initialization, the default address space is physical memory and the default data size is longword. If you specify conflicting address space or data sizes, the console ignores the command and issues an error message.

Format:

DEPOSIT [qualifier-list] {address} {data} [data...]

Qualifiers:

Data control: /B, /W, /L, /Q, /N:{count}, /STEP:{size}, /WRONG

Address space control: /G, /I, /M, /P, /V, /U

Arguments:

{address} A longword address that specifies the first location into which data is deposited. The address can be an actual address or a symbolic address.

{data} The data to be deposited. If the specified data is larger than the deposit data size, the firmware ignores the command and issues an error response. If the specified data is smaller than the deposit data size, it is extended on the left with zeros.

[{data}] Additional data to be deposited (as much as can fit on the command line).

Examples:

```
>>>D/P/B/N:1FF 0 0          ! Clear first 512 bytes of
                            ! physical memory.
>>>D/V/L/N:3 1234 5         ! Deposit 5 into four longwords
                            ! starting at virtual memory address
                            ! 1234.
>>>D/N:8 R0 FFFFFFFF        ! Loads GPRs R0 through R8 with -1.
>>>D/L/P/N:10/ST:200 0 8    ! Deposit 8 in the first longword of
                            ! the first 17 pages in physical
                            ! memory.
>>>D/N:200 - 0              ! Starting at previous address, clear
                            ! 513 longwords or 2052 bytes.
```

3.2.4 EXAMINE

The EXAMINE command examines the contents of the memory location or register specified by the address. If no address is specified, + is assumed. The display line consists of a single character address specifier, the physical address to be examined, and the examined data.

EXAMINE uses the same qualifiers as DEPOSIT. However, the /WRONG qualifier causes EXAMINE to ignore ECC errors on reads from physical memory. The EXAMINE command also supports an /INSTRUCTION qualifier, which will disassemble the instructions at the current address.

Format:

EXAMINE [qualifier-list] [address]

Qualifiers:

Data control: /B, /W, /L, /Q, /N:{count}, /STEP:{size}, /WRONG

Address space control: /G, /I, /M, /P, /V, /U

Command specific:

/INSTRUCTION Disassembles and displays the VAX MACRO–32 instruction at the specified address.

Arguments:

[{address}] A longword address that specifies the first location to be examined. The address can be an actual or a symbolic address. If no address is specified, + is assumed.

Examples:

```
>>>EX PC                          ! Examine the PC.
  G 0000000F FFFFFFFC
>>>EX SP                          ! Examine the SP.
  G 0000000E 00000200
>>>EX PSL                         ! Examine the PSL.
  M 00000000 041F0000
>>>E/M                            ! Examine PSL another way.
  M 00000000 041F0000
>>>E R4/N:5                       ! Examine R4 through R9.
  G 00000004 00000000
  G 00000005 00000000
  G 00000006 00000000
  G 00000007 00000000
  G 00000008 00000000
  G 00000009 801D9000

>>>EX PR$_SCBB                    !Examine the SCBB, IPR 17
  I 00000011 2004A000             ! (decimal).

>>>E/P 0                          ! Examine local memory 0.
  P 00000000 00000000

>>>EX /INS 20040000               ! Examine 1st byte of ROM.
  P 20040000   11 BRB     20040019

>>>EX /INS/N:5 20040019           ! Disassemble from branch.
  P 20040019   D0 MOVL    I^#20140000,@#20140000
  P 20040024   D2 MCOML   @#20140030,@#20140502
  P 2004002F   D2 MCOML   S^#0E,@#20140030
  P 20040036   7D MOVQ    R0,@#201404B2
  P 2004003D   D0 MOVL    I^#201404B2,R1
  P 20040044   DB MFPR    S^#2A,B^44(R1)

>>>E/INS                          ! Look at next instruction.
  P 20040048   DB MFPR    S^#2B,B^48(R1)

>>>
```

3.2.5 FIND

The FIND command searches main memory, starting at address zero for a page-aligned 128-Kbyte segment of good memory, or a restart parameter block (RPB). If the command finds the segment or RPB, its address plus 512 is left in Stack Pointer (SP) R14. If it does not find the segment or RPB, the console issues an error message and preserves the contents of SP. If you do not specify a qualifier, /RPB is assumed.

Format:

FIND [qualifier-list]

Qualifiers:

Command specific:

/MEMORY — Searches memory for a page-aligned block of good memory, 128K bytes in length. The search looks only at memory that is deemed usable by the bitmap. This command leaves the contents of memory unchanged.

/RPB — Searches all physical memory for an RPB. The search does not use the bitmap to qualify which pages are looked at. The command leaves the contents of memory unchanged.

Examples:

```
>>>EX SP                    ! Check the SP.
  G 0000000E 00000000
>>>FIND /MEM                ! Look for a valid 128 Kbytes.
>>>EX SP                    ! Note where it was found.
  G 0000000E 00000200
>>>FIND /RPB                ! Check for valid RPB.
?2C FND ERR 00C00004        ! None to be found here.
>>>
```

3.2.6 HALT

The HALT command has no effect. It is included for compatibility with other VAX consoles.

Format:

HALT

Example:

```
>>>HALT     ! Pretend to halt.
>>>
```

3.2.7 HELP

The HELP command provides information about command syntax and usage.

Format:

HELP

Example:

```
>>>HELP
Following is a brief summary of all the commands supported by the
console:

UPPERCASE  denotes a keyword that you must type in
|          denotes an OR condition
[]         denotes optional parameters
<>         denotes a field specifying a syntactically correct value
..         denotes one of an inclusive range of integers
...        denotes that the previous item may be repeated

Valid qualifiers:
    /B /W /L /Q /INSTRUCTION
    /G /I /V /P /M
    /STEP: /N: /NOT
    /WRONG /U

Valid commands:
    BOOT [[/R5:]<boot_flags>] [<boot_device>]
    CONTINUE
    DEPOSIT [<qualifiers>] <address> <datum> [<datum>...]
    EXAMINE [<qualifiers>] [<address>]
    FIND [/MEMORY | /RPB]
    HALT
    HELP
    INITIALIZE
    LOGIN
    MOVE [<qualifiers>] <address> <address>
    NEXT [<count>]
    REPEAT <command>
    SEARCH [<qualifiers>] <address> <pattern> [<mask>]
    SET BFLG <boot_flags>
    SET BOOT <boot_device>
    SET DSSI_ID <bus_number> <id>
    SET HALT <0..4 |DEFAULT|RESTART|REBOOT|HALT|RESTART_REBOOT>
    SET HOST/DUP/DSSI/BUS:<0..3> <node_number> [<task>]
    SET LANGUAGE <1..15>
    SET PSE <0..1 |DISABLED | ENABLED>
    SET PSWD <password>
    SET RECALL <0..1 | DISABLED | ENABLED>
    SET SCSI_ID  <0..7>
    SHOW BFLG
    SHOW BOOT
    SHOW CONFIG
    SHOW DEVICE
    SHOW DSSI  [0..3]
    SHOW DSSI_ID
    SHOW ERRORS
    SHOW ESTAT
    SHOW ETHERNET
    SHOW HALT
    SHOW LANGUAGE
    SHOW MEMORY [/FULL]
```

```
        SHOW PSE
        SHOW RECALL
        SHOW SAVED_STATE
        SHOW SCSI
        SHOW SCSI_ID
        SHOW TESTS
        SHOW TRANSLATION <physical_address>
        SHOW VERSION
        START <address>
        TEST [<test_code> [<parameters>]]
        UNJAM
        X <address> <count>

>>>
```

3.2.8 INITIALIZE

The INITIALIZE command performs a processor initialization.

Format:

INITIALIZE

The following registers are initialized:

Register	State at Initialization
PSL	041F0000
IPL	1F
ASTLVL	4
SISR	0
ICCS	Bits <6> and <0> clear; the rest are unpredictable.
RXCS	0
TXCS	80
MAPEN	0
Caches	Flushed
Instruction buffer	Unaffected
Console previous reference	Longword, physical, address 0
TODR	Unaffected
Main memory	Unaffected
General registers	Unaffected
Halt code	Unaffected

Register	State at Initialization
Bootstrap-in-progress flag	Unaffected
Internal restart-in-progress flag	Unaffected

The firmware clears all error status bits and initializes the following:

- CDAL bus timer
- Address decode and match registers
- Programmable timer interrupt vectors
- The QUART LPR register is set to 9600 baud
- All error status bits are cleared

Example:

```
>>>INIT
>>>
```

3.2.9 LOGIN

Allows you to put the system in privileged console mode. When the console security feature is enabled and when you put the system in secure console mode, the system operates in unprivileged console mode. You can access only a subset of the console commands. To access the full range of console commands, you must use this command. This command may only be executed in secure console mode. The format of this command is as follows:

LO[GIN]

When you enter the command, the system prompts you for a password as follows:

```
Password:
```

You must enter the current console security password. If you do not enter the correct password, the system displays the error message, INCORRECT PASSWORD. When you enter the console security password, the system operates in privileged console mode. In this mode, you can use all the console commands. The system exits from privileged console mode when you enter one of the following console commands:

- BOOT
- CONTINUE
- HALT

- START

3.2.10 MOVE

The MOVE command copies the block of memory starting at the source address to a block beginning at the destination address. Typically, this command has an /N qualifier so that more than one datum is transferred. The destination correctly reflects the contents of the source, regardless of the overlap between the source and the data.

The MOVE command actually performs byte, word, longword, and quadword reads and writes as needed in the process of moving the data. Moves are supported only for the physical and virtual address spaces.

Format:

MOVE [qualifier-list] [src_address] [dest_address]

Qualifiers:

Data control: /B, /W, /L, /Q, /N:[count], /STEP:[size], /WRONG

Address space control: /V, /U, /P

Arguments:

[src_address] A longword address that specifies the first location of the source data to be copied.

[dest_address] A longword address that specifies the destination of the first byte of data. These addresses may be an actual address or a symbolic address. If no address is specified, + is assumed.

Examples:

```
>>>EX/N:4 0              ! Observe destination.
P 00000000 00000000
P 00000004 00000000
P 00000008 00000000
P 0000000C 00000000
P 00000010 00000000

>>>EX/N:4 200            ! Observe source data.
P 00000200 58DD0520
P 00000204 585E04C1
P 00000208 00FF8FBB
P 0000020C 5208A8D0
P 00000210 540CA8DE

>>>MOV/N:4 200 0         ! Move the data.
```

```
>>>EX/N:4 0                ! Observe moved data.
P 00000000 58DD0520
P 00000004 585E04C1
P 00000008 00FF8FBB
P 0000000C 5208A8D0
P 00000010 540CA8DE
>>>
```

3.2.11 NEXT

The NEXT command executes the specified number of macro instructions. If no count is specified, 1 is assumed.

After the last macro instruction is executed, the console reenters console I/O mode.

Format:

NEXT [count]

The console implements the NEXT command, using the trace trap enable and trace pending bits in the PSL and the trace pending vector in the SCB.

The console enters the "Spacebar Step Mode". In this mode, subsequent spacebar strokes initiate single steps and a carriage return forces a return to the console prompt.

The following restrictions apply:

- If memory management is enabled, the NEXT command works only if the first page in SSC RAM is mapped in S0 (system) space.
- Overhead associated with the NEXT command affects execution time of an instruction.
- The NEXT command elevates the IPL to 31 for long periods of time (milliseconds) while single-stepping over several commands.
- Unpredictable results occur if the macro instruction being stepped over modifies either the SCBB or the trace trap entry. This means that you cannot use the NEXT command in conjunction with other debuggers.

Arguments:

[count] A value representing the number of macro instructions to execute.

Examples:

```
>>>DEP 1000 50D650D4                ! Create a simple program.
>>>DEP 1004 125005D1
>>>DEP 1008 00FE11F9
>>>EX /INSTRUCTION /N:5 1000        ! List it.
  P 00001000   D4 CLRL    R0
  P 00001002   D6 INCL    R0
  P 00001004   D1 CMPL    S^#05,R0
  P 00001007   12 BNEQ    00001002
  P 00001009   11 BRB     00001009
  P 0000100B   00 HALT
>>>DEP PR$ SCBB 200                ! Set up a user SCBB...
>>>DEP PC 1000                      ! ...and the PC.
>>>
>>>N                               ! Single step...
  P 00001002   D6 INCL    R0         ! SPACEBAR
  P 00001004   D1 CMPL    S^#05,R0   ! SPACEBAR
  P 00001007   12 BNEQ    00001002   ! SPACEBAR
  P 00001002   D6 INCL    R0         ! CR
>>>N 5                             ! ...or multiple step the program.
  P 00001004   D1 CMPL    S^#05,R0
  P 00001007   12 BNEQ    00001002
  P 00001002   D6 INCL    R0
  P 00001004   D1 CMPL    S^#05,R0
  P 00001007   12 BNEQ    00001002
>>>N 7
  P 00001002   D6 INCL    R0
  P 00001004   D1 CMPL    S^#05,R0
  P 00001007   12 BNEQ    00001002
  P 00001002   D6 INCL    R0
  P 00001004   D1 CMPL    S^#05,R0
  P 00001007   12 BNEQ    00001002
  P 00001009   11 BRB     00001009
>>>N
  P 00001009   11 BRB     00001009
>>>
```

3.2.12 REPEAT

The REPEAT command repeatedly displays and executes the specified command. Press Ctrl/C to stop the command. You can specify any valid console command except the REPEAT command.

Format:

REPEAT {command}

Arguments:

(command) A valid console command other than REPEAT.

Examples:

```
>>>REPEAT EX PR$_TODR !Watch the clock.
  I 0000001B 5AFE78CE
  I 0000001B 5AFE78D1
  I 0000001B 5AFE78FD
  I 0000001B 5AFE7900
  I 0000001B 5AFE7903
  I 0000001B 5AFE7907
  I 0000001B 5AFE790A
  I 0000001B 5AFE790D
  I 0000001B 5AFE7910
  I 0000001B 5AFE793C
  I 0000001B 5AFE793F
  I 0000001B 5AFE7942
  I 0000001B 5AFE7946
  I 0000001B 5AFE7949
  I 0000001B 5AFE794C
  I 0000001B 5AFE794F
  I 0000001B 5^C
>>>
```

3.2.13 SEARCH

The SEARCH command finds all occurrences of a pattern and reports the addresses where the pattern was found. If the /NOT qualifier is present, the command reports all addresses in which the pattern did not match.

Format:

SEARCH [qualifier-list] [address] [pattern] [[mask]]

SEARCH accepts an optional mask that indicates bits to be ignored (*don't care* bits). For example, to ignore bit 0 in the comparison, specify a mask of 1. The mask, if not present, defaults to 0.

A match occurs if (pattern and not mask) = (data and not mask), where:

Pattern is the target data.
Mask is the optional don't care bitmask (which defaults to 0).
Data is the data at the current address.

SEARCH reports the address under the following conditions:

/NOT Qualifier	Match Condition	Action
Absent	True	Report address
Absent	False	No report
Present	True	No report
Present	False	Report address

The address is advanced by the size of the pattern (byte, word, longword, or quadword), unless overridden by the /STEP qualifier.

Qualifiers:

Data control: /B, /W, /L, /Q, /N:{count}, /STEP:{size}, /WRONG

Address space control: /P, /V, /U

Command specific:

/NOT Inverts the sense of the match.

Arguments:

{start_address} A longword address that specifies the first location subject to the search. This address can be an actual address or a symbolic address. If no address is specified, + is assumed.

{pattern} The target data.

[{mask}] A mask of the bits desired in the comparison.

Examples:

```
>>>DEP /P/L/N:1000 0 0                ! Clear some memory.
>>>
>>>DEP 300 12345678                   ! Deposit some search data.
>>>DEP 401 12345678
>>>DEP 502 87654321
>>>
>>>SEARCH /N:1000 /ST:1 0 12345678    ! Search for all occurrences
  P 00000300 12345678                 ! of 12345678 on any byte
  P 00000401 12345678                 ! boundary.  Then try on
>>>SEARCH /N:1000 0 12345678          ! longword boundaries.
  P 00000300 12345678                 ! Search for all non-zero
>>>SEARCH /N:1000 /NOT 0 0            ! longwords.
  P 00000300 12345678
  P 00000400 34567800
  P 00000404 00000012
  P 00000500 43210000
  P 00000504 00008765
>>>SEARCH /N:1000 /ST:1 0 1 FFFFFFFE  ! Search for odd-numbered
                                      ! longwords on any boundary.
  P 00000502 87654321
  P 00000503 00876543
  P 00000504 00008765
  P 00000505 00000087
>>>SEARCH /N:1000 /B 0 12             ! Search for all occurrences
  P 00000303 12                       ! of the byte 12.
  P 00000404 12
>>>SEARCH /N:1000 /ST:1 /w 0 FE11     ! Search for all words that
>>>                                   ! could be interpreted as
>>>                                   ! a spin (10$: brb 10$).
>>>                                   ! Note that none were found.
```

3.2.14 SET

The SET command sets the parameter to the value you specify.

Format:

SET {parameter} {value}

Parameters:

BFLAG	Sets the default R5 boot flags. The value must be a hex number of up to eight digits.
BOOT	Sets the default boot device. The value must be a valid device name or list of device names as specified in the BOOT command description in Section 3.2.1.
HALT	Sets the user-defined halt action. Acceptable values are the keywords "default", "restart", "reboot", "halt", "restart_reboot", or a number in the range 0 to 4 inclusive.

HOST	Invoke the DUP or MAINTENANCE driver on the selected node. Only SET HOST/DUP accepts a value parameter. The hierarchy of the SET HOST qualifiers listed below suggests the appropriate usage. Each qualifier only supports additional qualifiers at levels below it.
LANGUAGE	Sets console language and keyboard type. If the current console terminal does not support the multinational character set (MCS), then this command has no effect and the console message appears in English. Values are 1 through 15.
PSE	Allows you to enable or disable the console security feature of the system. The SET PSE command accepts the following values: • 0—Console security disabled • 1—Console security enabled When the console security feature is enabled, only a subset of the console commands is available to the user. To enable the complete set of console commands once the console security feature is enabled, you must use the LOGIN command (see Section 3.2.9).
PSWD	Allows you to set or change the console security password.
RECALL	Sets command recall state to either ENABLED (1) or DISABLED (0).
SCSI_ID	Sets the SCSI ID of the SCSI controller to a number in the range 0 to 7. The SCSI ID of the SCSI controller is set to 6 before the system is shipped. For the KZDDA option second SCSI bus, You must enter two arguments; the bus, then the id.

Qualifiers: Listed in the parameter descriptions above.

Examples:

```
>>>
>>>SET BFLAG 220
>>>
>>>SET BOOT DUA0
>>>
>>>SET LANGUAGE 5
>>>
>>>SET HALT RESTART
>>>
```

3.2.15 SHOW

The SHOW command displays the console parameter you specify.

Format:

SHOW [parameter]

Parameters:

BFLAG	Displays the default R5 boot flags.
BOOT	Displays the default boot device.
CONFIG	Displays the system configuration. The command displays information about the devices that the firmware has tested. It also displays the device errors that the most recent device test detected.
DEVICE	Displays all devices in the system.
HALT	Shows the user-defined halt action.
ESTAT	Shows results from last run of the system exerciser, tests 100 to 107. Data is volatile and is destroyed by running other tests or boots, etc. SHOW ESTAT normally done immediately after running the system test.
ERRORS	Shows saved data on tests which failed.
ETHERNET	Displays hardware Ethernet address for all Ethernet adapters that can be found. Displays as blank if no Ethernet adapter is present.
LANGUAGE	Displays console language and keyboard type. Refer to the corresponding SET LANGUAGE command for the meaning.
MEMORY	Displays main memory configuration. /FULL—Additionally, displays the normally inaccessible areas of memory, such as the PFN bitmap pages, and the console scratch memory pages. Also reports the addresses of bad pages, as defined by the bitmap.
PSE	Displays the condition of the console security feature of the system.
RECALL	Shows the current state of command recall, either ENABLED or DISABLED.

This information is obtained from the media type field of the MSCP command GET UNIT STATUS. The console does not display device information if a node is not running (or cannot run) an MSCP server.)

SCSI	Shows any SCSI devices in the system.
TRANSLATION	Shows any virtual addresses that map to the specified physical address. The firmware uses the current values of page table base and length registers to perform its search; it is assumed that page tables have been properly built.
VERSION	Displays the current firmware version.

Qualifiers: Listed in the parameter descriptions above.

Examples:

```
>>>
>>>SHOW BFLAG
00000220
>>>
>>>SHOW BOOT
DUA0
>>>SHOW CONTROLP
>>>
>>>SHOW ETHERNET
Ethernet Adapter
-EZA0 (08-00-2B-0B-29-14)
>>>
>>>SHOW HALT
restart
>>>
>>>SHOW LANGUAGE
English (United States/Canada)
>>>

>>>show memory

16 MB RAM, SIMM Set (0A,0B,0C,0D) present
Memory Set 0: 04000000 to 04FFFFFF, 16MB, 32768 good pages, 0 bad pages

64 MB RAM, SIMM Set (1E,1F,1G,1H) present
Memory Set 1: 00000000 to 03FFFFFF, 64MB, 131072 good pages, 0 bad pages

Total of 80MB, 163840 good pages, 0 bad pages, 136 reserved pages
>>>


;show memory / full
>>>show mem/full

16 MB RAM, SIMM Set (0A,0B,0C,0D) present
Memory Set 0: 00000000 to 00FFFFFF, 16MB, 32768 good pages, 0 bad pages

Total of 16MB, 32768 good pages, 0 bad pages, 104 reserved pages

Memory Bitmap
-00FF3000 to 00FF3FFF, 8 pages

Console Scratch Area
-00FF4000 to 00FF7FFF, 32 pages

Scan of Bad Pages
>>>
```

```
>>>SHOW SCSI
SCSI Adapter 0 (761300), SCSI ID 7
-DKA100 (DEC TLZ04)
>>>
>>>SHOW TRANSLATION 1000
 V 80001000
>>>
>>>SHOW VERSION
KA50 Vn.n VMBn.n

>>>
```

3.2.16 START

The START command starts instruction execution at the address you specify. If no address is given, the current PC is used. If memory mapping is enabled, macro instructions are executed from virtual memory, and the address is treated as a virtual address. The START command is equivalent to a DEPOSIT to PC, followed by a CONTINUE. It does not perform a processor initialization.

Format:

START [[address]]

Arguments:

[address] The address at which to begin execution. This address is loaded into the user's PC.

Example:

```
>>>START 1000
```

3.2.17 TEST

The TEST command invokes a diagnostic test program specified by the test number. If you enter a test number of 0 (zero), the power-up diagnostics are executed. The console accepts an optional list of up to five additional hexadecimal arguments.

Refer to Chapter 5 for a detailed explanation of the diagnostics.

Format:

TEST [[test_number] [[test_arguments]]]

Arguments:

[test_number] A two-digit hex number specifying the test to be executed. No meaning to console, but meaning to tests themselves. **T 9E** lists arguments used by applicable tests.

(test_arguments) Up to five additional test arguments. These arguments are accepted, but they have no meaning to the console.

Example:

```
>>>TEST 0
72..71..70..69..68..67..66..65..64..63..62..61..60..59..58..57..
56..55..54..53..52..51..50..49..48..47..46..45..44..43..42..41..
40..39..38..37..36..35..34..33..32..31..30..29..28..27..26..25..
24..23..22..21..20..19..18..17..16..15..14..13..12..11..10..09..
08..07..06..05..04..03..
Tests completed.
>>>
```

Example:

```
>>>              ! Display the CPU registers.

>>>T 9C

  savpc=20048C68 savpsl=20048C68    sbr=03FA0000    slr=00003040
   p0br=80000000   p0lr=00182000   p1br=00000000   p1lr=00000000
    sid=13001401    sie=03020801  mapen=00000000
   tcr0=00000000   tir0=00000000  tnir0=00000000  tivr0=00000078
   tcr1=00000001   tir1=02AF768E  tnir1=0000000F  tivr1=0000007C
    bdr=3FFB08FF  ssccr=00D05070   scbb=20053400
DZ  csr=0020        tcr=0008        msr=0F75
    scr=0000D000   dser=00000000  qbear=0000000F   dear=00000000
  qbmbr=03FF8000   ipcr=0000
nicsr0=1FFF0003  3=00004030   4=00004050   5=8039FF00   6=83E0F000  7=00000000
nicsr9=04E204E2 10=00040000  11=00000000  12=00000000  13=00000000 15=0000FFFF
NISA=08-00-2B-29-1C-7A    intmsk=00 intreq=00 scdadr=00000000 scddir=0
SCSI_CSRs 0=00 1=00 2=00 3=00 4=00 6=05 5=05 7=00 8=16 9=5B A=5B B=00 C=04
VIC.....icsr=00000001    vmar=000007E0     ecr=000000CA
PC......pcctl=FFFFFC13   pcsts=FFFFF800   pcadr=FFFFFFF8
BC_128K..cctl=00000007 bcetsts=000003E0 bcetidx=FFFFFFE0  bcetag=FFFFFE00
......bcedsts=00000F00 bcedidx=001FFFF8 bcedecc=00000000
........nests=00000000  neoadr=E0055F70  neocmd=8000FF04  neicmd=000003FF
......nedathi=FFFFFFFF nedatlo=FF7F9FFF  cefsts=00019200  cefadr=E00002C0
MEMORY...mesr=00006000    mear=08406010     Add=21018040  mmcdsr=01111000
......memcon0=80000005 memcon1=00000007   moamr=00000000     ssr=C0CE
NCA......cesr=00000000  cmcdsr=0000C108   cnear=00000000
.......csear1=00000000  csear2=00000000 cioear1=00000000 cioear2=000002C0
.........iccs=00000000    nicr=FFFFD8F0     icr=FFFFD8F0    todr=00000000
>>>
```

Example:

```
>>>
>>>       ! list diagnostics and scripts
>>>TEST

Test
#   Address   Name                Parameters
-------------------------------------------------------------------------------
    20052200  SCB
    20055850  De_executive
30  2006A53C  Memory_Init_Bitmap  *** mark_Hard_SBEs ******
31  2006AB34  Memory_Setup_CSRs   **********
32  2005D148  NMC_registers       **********
33  2005D324  NMC_powerup         **
34  2005E6D8  SSC_ROM             ***
35  2005FB90  B_Cache_diag_mode   bypass_test_mask *********
37  20061590  Cache_w_Memory      bypass_test_mask *********
40  2006B5E0  Memory_count_pages  SIMM_set0 SIMM_set1 Soft_errs_allowed *****
41  20068CEC  Board_Reset         *
42  20061880  Chk_for_Interrupts  **********
46  200610C4  P_Cache_diag_mode   bypass_test_mask *********
47  2006AD04  Memory_Refresh      start_a end_incr cont_on_err time_seconds *****
48  2006B028  Memory_Addr_shorts start_add end_add * cont_on_err pat2 pat3 ****
4A  2006A23C  Memory_ECC_SBEs    start_add end_add add_incr cont_on_err ******
4B  2006940C  Memory_Byte_Errors start_add end_add add_incr cont_on_err ******
4C  20069BA0  Memory_ECC_Logic   start_add end_add add_incr cont_on_err ******
4D  20068FE8  Memory_Address     start_add end_add add_incr cont_on_err ******
4E  20069188  Memory_Byte        start_add end_add add_incr cont_on_err ******
4F  2006B7F4  Memory_Data        start_add end_add add_incr cont_on_err ******
51  2005803C  FPA                **********
52  20058530  SSC_Prog_timers    which_timer wait_time_us ***
53  20058818  SSC_TOY_Clock      repeat_test_250ms_ea Tolerance ***
54  20057C18  Virtual_Mode       ********
55  20058E6C  Interval_Timer     *****
56  2006507C  SHAC_LPBCK         From_bus To_bus passes *******
58  20065D24  SHAC_RESET         dssi_bus port_number time_secs not_pres
59  20062778  SGEC_LPBCK_ASSIST  time_secs **
5C  20062D10  SHAC               SHAC_number *********
5F  200619B8  SGEC               loopback_type no_ram_tests ******
62  20058B1C  console_QDSS       mark_not_present selftest_r0 selftest_r1 ******
63  20058CA4  QDSS_any           input_csr selftest_r0 selftest_r1 ******
80  2005D3C0  CQBIC_memory       bypass_test_mask *********
81  200596CC  Qbus_MSCP          IP_csr ******
82  200598AC  Qbus_DELQA         device_num_addr ****
83  2005A85C  QZA_Intlpbck1      controller_number ********
84  2005BF1C  QZA_Intlpbck2      controller_number *********
85  20059A9C  QZA_memory         incr test_pattern controller_number *******
86  20059F44  QZA_DMA            Controller_number main_mem_buf ********
90  20058494  CQBIC_registers    *
```

```
91  20058410  CQBIC_powerup       **
99  2005DC4C  Flush_Ena_Caches    dis_flush_VIC dis_flush_BC dis_flush_PC
9A  20063FB0  INTERACTION         pass_count disable_device ****
9B  20068E48  Init_memory         ***
9C  2006631C  List_CPU_registers  *
9D  2006C250  Utility             Flags *********
9E  2005903C  List_diagnostics    script_number *
9F  200681CC  Create_A0_Script    **********
C1  20057888  SSC_RAM_Data        *
C2  20057A78  SSC_RAM_Data_Addr   *
C5  200589E8  SSC_registers       *
D0  20060C70  V_Cache_diag_mode   bypass_test_mask *********
D2  2005DE90  O_Bit_diag_mode     bypass_test_mask *********
DA  2006139C  PB_Flush_Cache      **********
DB  2005E850  Speed               print_speed *********
DC  2006C060  NO_Memory_present   *
DD  2005F0DC  B_Cache_Data_debug  start_add end_add add_incr *******
DE  2005EC64  B_Cache_Tag_Debug   start_add end_add add_incr *******
DF  2005E2A8  O_BIT_DEBUG         start_add end_add add_incr seg_incr ******
E0  2006D4D4  SCSI                environment reset_bus time_s *******
E1  2006D7CC  SCSI_Utility        environment util_nbr target_ID lun ******
E2  2006DA2C  SCSI_MAP            bypass_test addr_incr_data_tst ********
E4  2006DFC8  DZ                  environment *********
E8  2006E1DC  SYNC                environment *********
E9  2006E2B4  SYNC_Utility        environment *********
EC  2006E398  ASYNC               environment *********
F0  2006D638  SCSI_option         environment reset_bus time_s *******
F1  2006D900  SCSI_Opt_Utility    environment util_nbr target_ID lun ******
F2  2006DA40  SCSI_MAP_Option     bypass_test addr_incr_data_tst ********

Scripts
#   Description

A0  User defined scripts
A1  Powerup tests, Functional Verify, continue on error, numeric countdown
A3  Functional Verify, stop on error, test # announcements
A4  Loop on A3 Functional Verify
A6  Memory tests, mark only multiple bit errors
A7  Memory tests
A8  Memory acceptance tests, mark single and multi-bit errors, call A7
A9  Memory tests, stop on error
B2  Extended tests plus BF, then loop
B5  Extended tests, then loop
BF  DZ, SYNC, ASYNC with loopbacks
```

```
Load & start system exerciser
100 Customer mode, 2 passes
101 CSSE mode, 2 passes
102 CSSE mode, continous until ^C
103 Manuf mode, continous until ^C
104 Manuf TINA mode, continous until ^C
105 Manuf mode, 2 passes
106 CSSE mode, select tests, continous until ^C
107 Manuf mode, select tests, continous until ^C

>>>
>>>
```

3.2.18 UNJAM

The UNJAM command performs an I/O bus reset, by writing a 1 (one) to IPR 55 (decimal). SHAC and SGEC are explicitly reset, EDAL_INTREQ register error bits are cleared and SCSI_DMA map registers are cleared.

Format:

UNJAM

Example:

```
>>>UNJAM
>>>
```

3.2.19 X—Binary Load and Unload

The X command is for use by automatic systems communicating with the console.

The X command loads or unloads (that is, writes to memory, or reads from memory) the specified number of data bytes through the console serial line (regardless of console type) starting at the specified address.

Format:

X {address} {count} CR {line_checksum} {data} {data_checksum}

If bit 31 of the count is clear, data is received by the console and deposited into memory. If bit 31 is set, data is read from memory and sent by the console. The remaining bits in the count are a positive number indicating the number of bytes to load or unload.

The console accepts the command upon receiving the carriage return. The next byte the console receives is the command checksum, which is not echoed. The command checksum is verified by adding all command characters, including the checksum and separating space (but not including the terminating carriage return, rubouts, or characters deleted by rubout), into an 8-bit register initially set to zero. If no errors occur, the result is zero. If the command checksum

is correct, the console responds with the input prompt and either sends data to the requester or prepares to receive data. If the command checksum is in error, the console responds with an error message. The intent is to prevent inadvertent operator entry into a mode where the console is accepting characters from the keyboard as data, with no escape mechanism possible.

If the command is a load (bit 31 of the count is clear), the console responds with the input prompt (>>>), then accepts the specified number of bytes of data for depositing to memory, and an additional byte of received data checksum. The data is verified by adding all data characters and the checksum character into an 8-bit register initially set to zero. If the final content of the register is nonzero, the data or checksum are in error, and the console responds with an error message.

If the command is a binary unload (bit 31 of the count is set), the console responds with the input prompt (>>>), followed by the specified number of bytes of binary data. As each byte is sent, it is added to a checksum register initially set to zero. At the end of the transmission, the two's complement of the low byte of the register is sent.

If the data checksum is incorrect on a load, or if memory or line errors occur during the transmission of data, the entire transmission is completed, then the console issues an error message. If an error occurs during loading, the contents of the memory being loaded are unpredictable.

The console represses echo while it is receiving the data string and checksums.

The console terminates all flow control when it receives the carriage return at the end of the command line in order to avoid treating flow control characters from the terminal as valid command line checksums.

You can control the console serial line during a binary unload using control characters (Ctrl/C, Ctrl/S, Ctrl/O, and so on). You cannot control the console serial line during a binary load, since all received characters are valid binary data.

The console has the following timing requirements:

- It must receive data being loaded with a binary load command at a rate of at least one byte every 60 seconds.
- It must receive the command checksum that precedes the data within 60 seconds of the carriage return that terminates the command line.
- It must receive the data checksum within 60 seconds of the last data byte.

If any of these timing requirements are not met, then the console aborts the transmission by issuing an error message and returning to the console prompt.

The entire command, including the checksum, can be sent to the console as a single burst of characters at the specified character rate of the console serial line. The console is able to receive at least 4 Kbytes of data in a single X command.

3.2.20 ! (Comment)

The comment character (an exclamation point) is used to document command sequences. It can appear anywhere on the command line. All characters following the comment character are ignored.

Format: !

Example:

```
>>>! The console ignores this line.
>>>
```

4

System Initialization and Acceptance Testing (Normal Operation)

This chapter describes the system initialization, testing, and bootstrap processes that occur at power-up. In addition, the acceptance test procedure to be performed when installing a system or whenever adding or replacing FRUs is described.

Note

The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95 and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, will appear on the console and/or printouts from time to time.

4.1 Basic Initialization Flow

On power-up, the firmware identifies the console device, optionally performs a language inquiry, and runs the diagnostics.

The firmware waits for power to stabilize by monitoring SCR<15>(POK). Once power is stable, the firmware verifies that the console battery backup RAM (BBU RAM) is valid (backup battery is charged) by checking SSCCR<31>(BLO). If it is invalid or zero (battery is discharged), the BBU RAM is initialized.

After the battery check, the firmware tries to determine the type of terminal attached to the console serial line. It uses this information to determine if multinational support is appropriate.

The console uses the saved console language if the contents of the BBU RAM are valid.

If the firmware detects that the contents of the BBU RAM are invalid, the firmware prompts you for the language to be used for displaying the following system messages (if the console terminal supports the multinational character set):

```
Loading system software.
Failure.
Restarting system software.
Performing normal system tests.
Tests completed.
Normal operation not possible.
Bootfile.
Memory configuration error.
No default boot device has been specified.
Available devices.
Device?
Retrying network bootstrap.
```

The position of the Break Enable/Disable switch has no effect on these conditions. The firmware will not prompt for a language if the console terminal, such as the VT100, does not support the multinational character set (MCS).

Following a successful diagnostic countdown (see Example 4–1), the console may prompt you for a default boot device.

Example 4–1 Successful Diagnostic Countdown

```
KA50-A VX.X, VMB 2.14
Performing normal system tests.
72..71..70..69..68..67..66..65..64..63..62..61..60..59..58..57..
56..55..54..53..52..51..50..49..48..47..46..45..44..43..42..41..
40..39..38..37..36..35..34..33..32..31..30..29..28..27..26..25..
24..23..22..21..20..19..18..17..16..15..14..13..12..11..10..09..
08..07..06..05..04..03..
Tests completed.
>>>
```

4.2 Power-On Self-Tests (POST)

Power-on self-tests provide core testing of the system kernel comprised of the CPU and memory. Certain registers are flushed, and data structures are set up to initialize and set the system to a known state for the operating system.

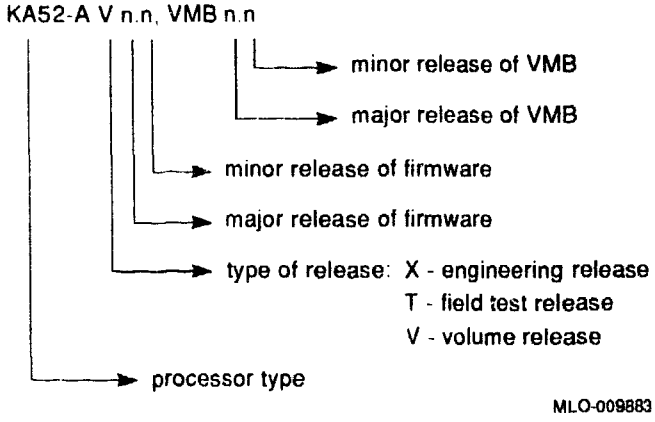
4.2.1 Power-Up Tests for Kernel

In a nonmanufacturing environment where the intended console device is the serial line unit (SLU), the console program performs the following actions at power-up:

1. Checks for POK.
2. Establishes SLU as console device.
3. Prints banner message.

   The banner message contains the processor name, the version of the firmware, and the version of VMB. The letter code in the firmware version indicates if the firmware is pre-field test, field test, or official release. The first digit indicates the major release number and the trailing digit indicates the minor release number (Figure 4–1).

Figure 4–1 Console Banner



4. Displays language inquiry menu on console if console supports multinational character set (MCS) *and any* of the following are true:
   - Battery is dead.
   - Contents of SSC RAM are invalid.
5. Calls the diagnostic executive (DE) with Test Code = 0.
   a. DE executes script A1 (Tests system module and memory).

While the diagnostics are running, the LEDs display a test code. A different countdown appears on the console terminal. Refer to Table 5–4 for a complete explanation of the power-up test display. Table 4–1 lists the LED codes and the associated actions performed at power-up. Example 4–2 shows a successful power-up to a list of bootable devices.

b. DE passes control back to the console program.

6. Issues end message and >>> prompt.

Table 4–1 LED Codes

LED Value	Actions
F	Initial state on power-up, no code has executed
E	Entered ROM space, some instructions have executed
D	Waiting for power to stabilize (POK)
C	SSC RAM, SSC registers, and ROM checksum tests
B	O-bit memory, interval timer, and virtual mode tests
A	FPA tests
9	Backup cache tests
8	NMC, NCA, memory, and I/O interaction tests
7	CQBIC, SYNC, and ASYNC tests
6	Console and QUART tests
5	SCSI tests
4	SGEC Ethernet subsystem tests
3	"Console I/O" mode
2	Control passed to VMB
1	Control passed to secondary bootstrap
0	"Program I/O" mode, control passed to operating system

Example 4–2 Successful Power-Up to List of Bootable Devices

```
KA50-A VX.X, VMB 2.14
Performing normal system tests.
72..71..70..69..68..67..66..65..64..63..62..61..60..59..58..57..
56..55..54..53..52..51..50..49..48..47..46..45..44..43..42..41..
40..39..38..37..36..35..34..33..32..31..30..29..28..27..26..25..
24..23..22..21..20..19..18..17..16..15..14..13..12..11..10..09..
08..07..06..05..04..03..
Tests completed.
Loading system software.
No default boot device has been specified.
Available devices.
-DIA0 (RF73)
-DIA1 (RF73)
-MIA5 (TF85)
-EZA0 (08-00-2B-06-10-42)
Device? [EZA0]:
```

4.2.2 Power-Up Tests for Mass Storage Devices

An RZ-series ISE may fail either during initial power-up or during normal operation. In both cases, the failure is indicated by the lighting of the red fault LED on the drive's front panel. The ISE also has a red fault LED, but it is not visible from the outside of the system enclosure.

If the drive is unable to execute the Power-On Self-Test (POST) successfully, the red fault LED remains lit and the ready LED does not come on, or both LEDs remain on.

POST is also used to handle two types of error conditions in the drive:

- *Controller errors* are caused by the hardware associated with the controller function of the drive module. A controller error is fatal to the operation of the drive, since the controller cannot establish a logical connection to the host. The red fault LED lights. If this occurs, replace the drive module.
- *Drive errors* are caused by the hardware associated with the drive control function of the drive module. These errors are not fatal to the drive, since the drive can establish a logical connection and report the error to the host. Both LEDs go out for about 1 second, then the red fault LED lights.

4.3 CPU ROM-Based Diagnostics

The KA50/51/55/56 ROM-based diagnostic facility is the primary diagnostic tool for troubleshooting and testing of the CPU, memory, and Ethernet. ROM based diagnostics have significant advantages:

- Load time is virtually nonexistent.
- The boot path is more reliable.
- Diagnosis is done in a more primitive state.

The ROM-based diagnostics can detect failures in field-replaceable units (FRUs) other than the CPU module. For example, they can isolate to two memory SIMMS. (Table 5–4 lists the FRUs indicated by ROM-based diagnostic error messages.)

The diagnostics run automatically on power-up. While the diagnostics are running, the LED displays a hexadecimal number; while booting the operating system, 2 through 0 display.

The ROM-based diagnostics are a collection of individual tests with parameters that you can specify. A data structure called a *script* points to the tests (see Section 4.3.2). There are several field and manufacturing scripts.

A program called the *diagnostic executive* determines which of the available scripts to invoke. The script sequence varies if the system is in the manufacturing environment. The diagnostic executive interprets the script to determine what tests to run, the correct order to run the tests, and the correct parameters to use for each test.

The diagnostic executive also controls tests so that errors can be detected and reported. It ensures that when the tests are run, the machine is left in a consistent and well-defined state.

4.3.1 Diagnostic Tests

Example 4–3 shows a list of the ROM-based tests and utilities. To get this listing, enter T 9E at the console prompt (T is the abbreviation of TEST). The column headings have the following meanings:

Note

Base addresses shown in this document may not be the same as the addresses you see when you run T 9E. Run T 9E to get a list of actual addresses. See Example 4–3.

- Test is the test number or utility code.
- Address is the base address of where the test or utility starts in ROM. If a test fails, entering T FE displays diagnostic state to the console. You can subtract the base address of the failing test from the last_exception_pc to find the index into the failing test's diagnostic listing.
- Name is a brief description of the test or utility.
- Parameters shows the parameters for each diagnostic test or utility. These parameters are encoded in ROM and are provided by the diagnostic executive. Tests accept up to 10 parameters. The asterisks (*) represent parameters that are used by the tests but that you cannot specify individually. These parameters are displayed in error messages, each one preceded by identifiers P1 through P10.

Example 4–3 Test 9E

```
>>>T 9E

Test
#   Address   Name                Parameters
-----------------------------------------------------------------------
    20051200  SCB
    20054D28  De_executive
30  200645A4  Memory_Init_Bitmap *** mark_Hard_SBEs ******
31  20064E9C  Memory_Setup_CSRs  **********
32  20065288  NMC_registers      **********
33  20065440  NMC_powerup        **
34  2005DB60  SSC_ROM            ***
35  200682F4  B_Cache_diag_mode  bypass_test_mask *********
37  200691EC  Cache_w_Memory     bypass_test_mask *********
40  200631F4  Memory_count_pages SIMM_set0 SIMM_set1 Soft_errs_allowed *****
41  200581F8  Board_Reset        *
42  2005BEF0  Chk_for_Interrupts **********
46  2006801C  P_Cache_diag_mode  bypass_test_mask *********
47  20064B84  Memory_Refresh     start_a end incr cont_on_err time_seconds *****
48  200622E4  Memory_Addr_shorts start_add end_add * cont_on_err pat2 pat3 ****
4A  200642C8  Memory_ECC_SBEs    start_add end_add add_incr cont_on_err ******
4B  20062B24  Memory_Byte_Errors start_add end_add add_incr cont_on_err ******
4C  20063C70  Memory_ECC_Logic   start_add end_add add_incr cont_on_err ******
4D  20062144  Memory_Address     start_add end_add add_incr cont_on_err ******
4E  200628A0  Memory_Byte        start_add end_add add_incr cont_on_err ******
4F  20063408  Memory_Data        start_add end_add add_incr cont_on_err ******
51  2005C408  FPA                **********
52  2005C8C4  SSC_Prog_timers    which_timer wait_time_us ***
53  2005CBA8  SSC_TOY_Clock      repeat_test_250ms_ea Tolerance ***
54  2005C008  Virtual_Mode       ********
55  2005CD74  Interval_Timer     *****
58  20061060  SHAC_RESET         port_number time_secs not_pres
59  200602AC  SGEC_LPBCK_ASSIST  time_secs **
5C  2006082C  SHAC               bypass_test_mask *******
5F  2005F52C  SGEC               loopback_type no_ram_tests ******
63  2005D99C  QDSS_any           input_csr selftest_r0 selftest_r1 ******
80  20065884  CQBIC_memory       bypass_test_mask *********
81  2005D5D0  Qbus_MSCP          IP_csr ******
82  2005D7AC  Qbus_DELQA         device_num_addr ****
83  20059570  QZA_Intlpbck1      controller_number ********
84  2005AC74  QZA_Intlpbck2      controller_number *********
85  2005877C  QZA_memory         incr test_pattern controller_number *******
86  20058C74  QZA_DMA            Controller_number main_mem_buf ********
90  2005C82C  CQBIC_registers    *
91  2005C7A8  CQBIC_powerup      **
99  20065644  Flush_Ena_Caches   dis_flush_VIC dis_flush_BC dis_flush_PC
9A  2005DCB4  INTERACTION        pass_count disable_device ****
9B  200654DC  Init_memory        ***
9C  2005DC80  List_CPU_registers *
9D  2005EB60  Utility            Modify_CPU_type *********
9E  2005CF40  List_diagnostics   script_number *
9F  20061610  Create_A0_Script   **********
```

(continued on next page)

Example 4–3 (Cont.) Test 9E

```
C1  200583F8  SSC_RAM_Data          *
C2  200585E8  SSC_RAM_Data_Addr     *
C5  2005F414  SSC_registers         *
C6  20058320  SSC_powerup           *********
D0  20067BC8  V_Cache_diag_mode     bypass_test_mask *********
D2  200660E8  O_Bit_diag_mode       bypass_test_mask *********
DA  20068FFC  PB_Flush_Cache        **********
DB  20066908  Speed                 print_speed *********
DC  200650C8  NO_Memory_present     *
DD  20067118  B_Cache_Data_debug    start_add end_add add_incr *******
DE  20066CB8  B_Cache_Tag_Debug     start_add end_add add_incr *******
DF  200664F0  O_BIT_DEBUG           start_add end_add add_incr seq_incr ******
E0  200694D8  SCSI                  environment reset_bus time_s *******
E1  200695D8  SCSI_Utility          environment util_nbr target_ID lun ******
E2  200696A4  SCSI_MAP              bypass_test addr_incr_data_tst ********
E4  20069A60  DZ                    environment *********
E8  20069BF0  SYNC                  environment *********
E9  20069CC4  SYNC_Utility          environment *********
EC  20069DA8  ASYNC                 environment *********

Scripts
#   Description

A0  User defined scripts
A1  Powerup tests, Functional Verify, continue on error, numeric countdown
A3  Functional Verify, stop on error, test # announcements
A4  Loop on A3 Functional Verify
A6  Memory tests, mark only multiple bit errors
A7  Memory tests
A8  Memory acceptance tests, mark single and multi-bit errors, call A7
A9  Memory tests, stop on error
B2  Extended tests plus BF, then loop
B5  Extended tests, then loop
BF  DZ, SYNC, ASYNC with loopbacks

Load & start system exerciser
100 Customer mode, 2 passes
101 CSSE mode, 2 passes
102 CSSE mode, continous until ^C
103 Manuf mode, continous until ^C
104 Manuf TINA mode, continous until ^C
105 Manuf mode, 2 passes
106 CSSE mode, select tests, continous until ^C
107 Manuf mode, select tests, continous until ^C

>>>
```

User Determined Parameters

Parameters that you can specify are written out, as shown in the following examples:

```
30  2005C33C  Memory_Init_Bitmap *** mark_Hard_SBEs ******
54  20055181  Virtual_Mode       *********
```

For example, the virtual mode test contains several parameters, but you cannot specify any that appear in the table as asterisks. To run this test individually, enter:

```
>>>T 54
```

The MEM_bitmap test, for example, accepts 10 parameters, but you can only specify mark_hard_SBEs because the rest are asterisks. To map out solid, single-bit ECC memory errors, type:

```
>>>T 30 0 0 0 1
```

Even though you cannot change the first three parameters, you need to enter zeros (0) as placeholders. The zeros are placeholders for parameters 1 through 3, which allows the program to parse the command line correctly. The diagnostic executive then provides the proper value for the test.

You enter 1 for parameter 4 to indicate that the test should map out solid, single-bit as well as multibit ECC memory errors. You then terminate the command line by pressing RETURN. You do not need to specify parameters 5 through 10; placeholders are needed only for parameters that precede the user-definable parameter.

For the most part tests and scripts can be run without any special setup. If a test or script is run interactively without an intervening power up, such as after a system crash or shutdown, enter the UNJAM and INIT commands before running the tests or script. This will ensure that the CPU is in a well known state. If the commands are not entered, misleading errors may occur.

Other considerations to be aware of when running individual tests or scripts interactively:

- When using the TEST or REPEAT TEST commands, you must specify a test number, test code or script number following the TEST command before pressing RETURN.
- The memory bitmap and Q–bus scatter-gather map are created in main memory and the memory tests are run with these data structures left intact. Therefore, the upper portion of memory should not be accessed to avoid corrupting these data structures. The location of the maps is displayed using the SHOW MEMORY/FULL command.

4.3.2 Scripts

Most of the tests shown by utility 9E are arranged into scripts. A *script* is a data structure that points to various tests and defines the order in which they are run. Scripts should be thought of as diagnostic tables—these tables do not contain the actual diagnostic tests themselves, instead scripts simply define what tests or scripts should be run, the order that the tests or scripts should be run, and any input parameters to be parsed by the Diagnostic Executive.

Different scripts can run the same set of tests, but these tests can be run in a different order and/or with different parameters and flags. A script also contains the following information:

- The parameters and flags that need to be passed to the test.
- The locations from which the tests can be run. For example, certain tests can be run only from the FEPROM. Other tests are program-independent code, and can be run from FEPROM or main memory to enhance execution speed.
- What is to be shown, if anything, on the console.
- What is to be shown, if anything, in the LED display.
- What action to take on errors (halt, repeat, continue).

The power-up script runs every time the system is powered on. You can also invoke the power-up script at any time by entering T 0.

Additional scripts are included in the FEPROMs for use in manufacturing and engineering environments. Customer Services personnel can run these scripts and tests individually, using the T command. When doing so, note that certain tests may be dependent upon a state set up from a previous test. For this reason, use the UNJAM and INITIALIZE commands before running an individual test. You do not need these commands on system power-up because the system power-up leaves the machine in a defined state.

Customer Services Engineers (CSE) with a detailed knowledge of the system hardware and firmware can also create their own scripts by using the 9F User Script Utility. Table 4–2 lists the scripts available to Customer Services.

Table 4–2 Scripts Available to Customer Services

Script[1]	Enter with TEST Command	Description
A0	A0	Runs user-defined script. Enter T 9F to create.
A1	A1, 0	Primary power-up script; builds memory bitmap; marks hard single-bit errors and multi-bit errors. Continues or. error.
A3	A3, A4	Runs power-up tests, halts on first error.
A4	A4	Loops on A3. Press Ctrl C to exit.
A6	A6	Memory test script; initializes memory bitmap and marks only multiple bit errors.
A7	A7, A8	Memory test portion invoked by script A8. Reruns the memory tests without rebuilding and reinitializing the bitmap. Run script A8 once before running script A7 separately to allow mapping out of both single-bit and double-bit main memory ECC errors.
A8	A8	Memory acceptance. Running script A8 with script A7 tests main memory more extensively. It enables hard single-bit and multibit main memory ECC errors to be marked bad in the bitmap. Invokes script A7 when it has completed its tests.
A9	A9	Memory tests. Halts and reports the first error. Does not reset the bitmap or busmap. It is a quick way to specify which test caused a failure when a hard error is present.
AD	AD	Console program. Runs memory tests, marks bitmap, resets busmap, and resets caches. Calls script AE.
AE	AE, AD	Console program. Resets memory CSRs and resets caches. Also called by the INIT command.
AF	AF	Console program. Resets busmap and resets caches.
B2[2]	B2	Runs extended tests, calls the BF script, then loops. Press Ctrl C to exit.
B5	B5	Runs extended tests, then loops. Press Ctrl C to exit.
BF[2]	BF	Runs tests requiring loopback connectors for QUART, SYNC, and, ASYNC options if present. Press Ctrl C to exit.

[1]Scripts AD, AE, and AF exist primarily for console program; error displays and progress messages are suppressed (not recommended for CSE use).

[2]B2 and BF require loopback connectors.

4.4 Basic Acceptance Test Procedure

Perform the acceptance testing procedure listed below, after installing a system, or whenever adding or replacing the following:

CPU module
MS44 memory SIMM
SCSI device
SYNC device
ASYNC device

1. Run two error-free passes of the power-up scripts by entering the following command:

```
>>>T B5
```

Script B5 will halt on an error so that the error message will not scroll off the screen.

Press Ctrl/C to terminate the scripts. Refer to Chapter 5 if failures occur.

To check the memory configuration and to ensure there are no bad pages, enter the following command line:

```
>>>SHOW MEM/FULL

16 MB RAM, SIMM Set (0A,0B,0C,0D) present
Memory Set 0: 00000000 to 00FFFFFF, 16MB, 32768 good pages, 0 bad pages

Total of 16MB, 32768 good pages, 0 bad pages, 104 reserved pages

Memory Bitmap
-00FF3000 to 00FF3FFF, 8 pages

Console Scratch Area
-00FF4000 to 00FF7FFF, 32 pages

Scan of Bad Pages

Q-bus Map
-01FF8000 to 01FFFFFF, 64 pages

Scan of Bad Pages

>>>
```

The Q22–bus map always spans the top 32 Kbytes of good memory. The memory bitmap always spans two pages (1 Kbyte) for each 4 Mbytes of memory configured. Each bit within the memory bit map represents a page of memory.

To identify registers and register bit fields, see the *KA50/51/55/56 CPU Technical Manual.*

Examine MEMCON 0–1 to verify the memory configuration. Each pair of MEMCONs maps one memory module as follows:

MEMCON0	Set 0; 0A, 0B, 0C, 0D
MEMCON1	Set 1; 1E, 1F, 1G, 1H

4.5 Machine State on Power-Up

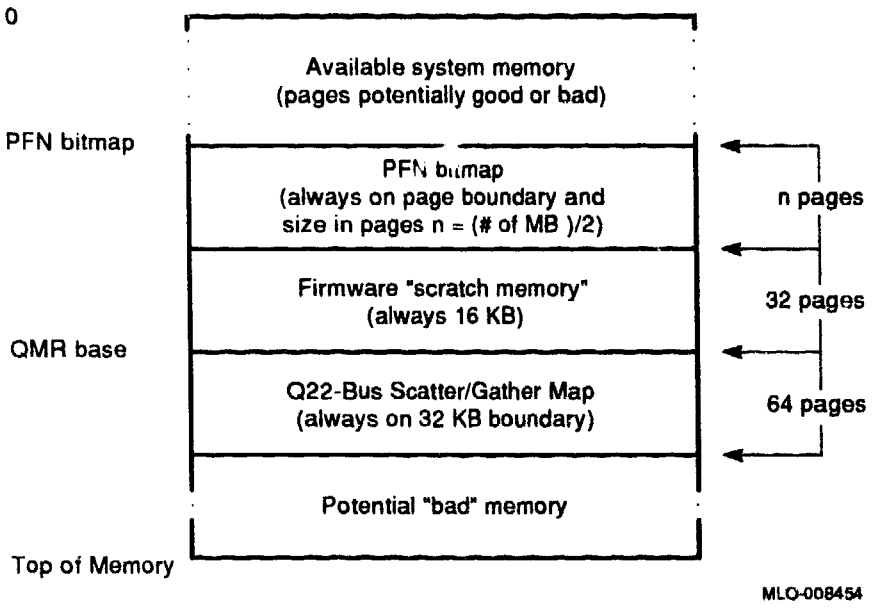
This section describes the state of the kernel after a power-up halt.

The descriptions in this section assume the system has just powered-up and the power-up diagnostics have successfully completed. The state of the machine is not defined if individual diagnostics are run or for any other halts other than a power-up halt (SAVPSL<13:8>(RESTART_CODE) = 3). Refer to Appendix D for a description of the normal state of CPU configurable bits following completion of power-up tests.

4.6 Main Memory Layout and State

Main memory is tested and initialized by the firmware on power-up. Figure 4–2 is a diagram of how main memory is partitioned after diagnostics.

Figure 4–2 Memory Layout After Power-Up Diagnostics



4.6.1 Reserved Main Memory

In order to build the scatter/gather map and the bitmap, the firmware attempts to find a physically contiguous page-aligned 1M byte block of memory at the highest possible address.

Of the 1M byte, the upper 32 KB is dedicated to the Q22–bus scatter/gather map, as shown in Figure 4–2. Of the lower portion, up to 32K bytes at the bottom of the block is allocated to the Page Frame Number (PFN) bitmap. The size of the PFN bitmap is dependent on the extent of physical memory. Each bit in the bitmap maps one page (512 bytes) of memory. The remainder of the block between the bitmap and scatter/gather map (minimally 16 KB) is allocated for the firmware.

4.6.1.1 PFN Bitmap

The PFN bitmap is a data structure that indicates which pages in memory are deemed usable by operating systems. The bitmap is built by the diagnostics as a side effect of the memory tests on power-up. The bitmap always starts on a page boundary. The bitmap requires 1 KB for every 4 MB of main memory, hence, a 8 MB system requires 2 KB, 16 MB requires 4 KB, 32 MB requires 8 KB, and a 64 MB requires 16 KB. There may be memory above the bitmap which has both good and bad pages.

Each bit in the PFN bitmap corresponds to a page in main memory. There is a one to one correspondence between a page frame number (origin 0) and a bit index in the bitmap. A one in the bitmap indicates that the page is "good" and can be used. A zero indicates that the page is "bad" and should not be used.

The PFN bitmap is protected by a checksum stored in the NVRAM. The checksum is a simple byte wide, two's complement checksum. The sum of all bytes in the bitmap and the bitmap checksum should result in zero.

4.6.1.2 Scatter/Gather Map

On power-up, the scatter/gather map is initialized by the firmware to map to the first 4M bytes of main memory. Main memory pages will not be mapped if there is a corresponding page in Q22–bus memory.

On a processor halt other than power-up, the contents of the scatter/gather map is undefined, and is dependent on operating system usage.

Operating systems should not move the location of the scatter/gather map, and should access the map only on aligned longwords through the local I/O space of 20088000 to 2008FFFC, inclusive. The Q22–bus map base register (QMBR), is set up by the firmware to point to this area, and should not be changed by software.

4.6.1.3 Firmware "Scratch Memory"

This section of memory is reserved for the firmware. However, it is only used after successful execution of the memory diagnostics and initialization of the PFN bitmap and scatter/gather map. This memory is primarily used for diagnostic purposes.

4.6.2 Contents of Main Memory

The contents of main memory are undefined after the diagnostics have run. Typically, nonzero test patterns will be left in memory.

The diagnostics will "scrub" all of main memory, so that no power-up induced errors remain in the memory system. On the KA50/51/55/56 memory subsystem, the state of the ECC bits and the data bits are undefined on initial power-up. This can result in single and multiple bit errors if the locations are read before written, because the ECC bits are not in agreement with their corresponding data bits. An aligned longword write to every location (done by diagnostics) eliminates all power-up induced errors.

4.6.3 Memory Controller Registers

The SHOW MEMORY command defines the mapping of addresses to specific SIMM sets as follows:

- MEMCON0 is used with SIMM bank 0 (the 0A, 0B, 0C, and 0D memory slots)
- MEMCON1 is used with SIMM bank 1 (the 1E, 1F, 1G, and 1H memory slots)

Additional information should be captured from the NMCDSR, MOAMR, MSER, and MEAR as needed.

4.6.4 On-Chip and Backup Caches

All three caches are tested.

4.6.5 Translation Buffer

The CPU translation buffer is tested by diagnostics on power-up, but not used by the firmware because it runs in physical mode. The translation buffer can be invalidated by using PR$_TBIA, IPR 57.

4.6.6 Halt-Protected Space

On the KA50/51/55/56, halt-protected space spans the first half of the 512K byte FEPROM from 20040000 to 2007FFFF. The second half of the FEPROM has data which is loaded into memory and run.

The firmware always runs in halt-protected space. When passing control to the bootstrap, the firmware exits the halt-protected space, so if halts are enabled, and the halt line is asserted, the processor will then halt before booting.

4.7 Operating System Bootstrap

Bootstrapping is the process by which an operating system loads and assumes control of the system. The KA50/51/55/56 supports bootstrap of the VAX/OpenVMS and VAXELN operating systems. Additionally, the KA50/51 /55 will boot MDM diagnostics and any user application image which conforms to the boot formats described herein.

On the KA50/51/55/56 a bootstrap occurs whenever a BOOT command is issued at the console or whenever the processor halts and the conditions specified in Table G–1 for automatic bootstrap are satisfied.

4.7.1 Preparing for the Bootstrap

Prior to dispatching to the primary bootstrap (VMB), the firmware initializes the system to a known state. The initialization sequence follows:

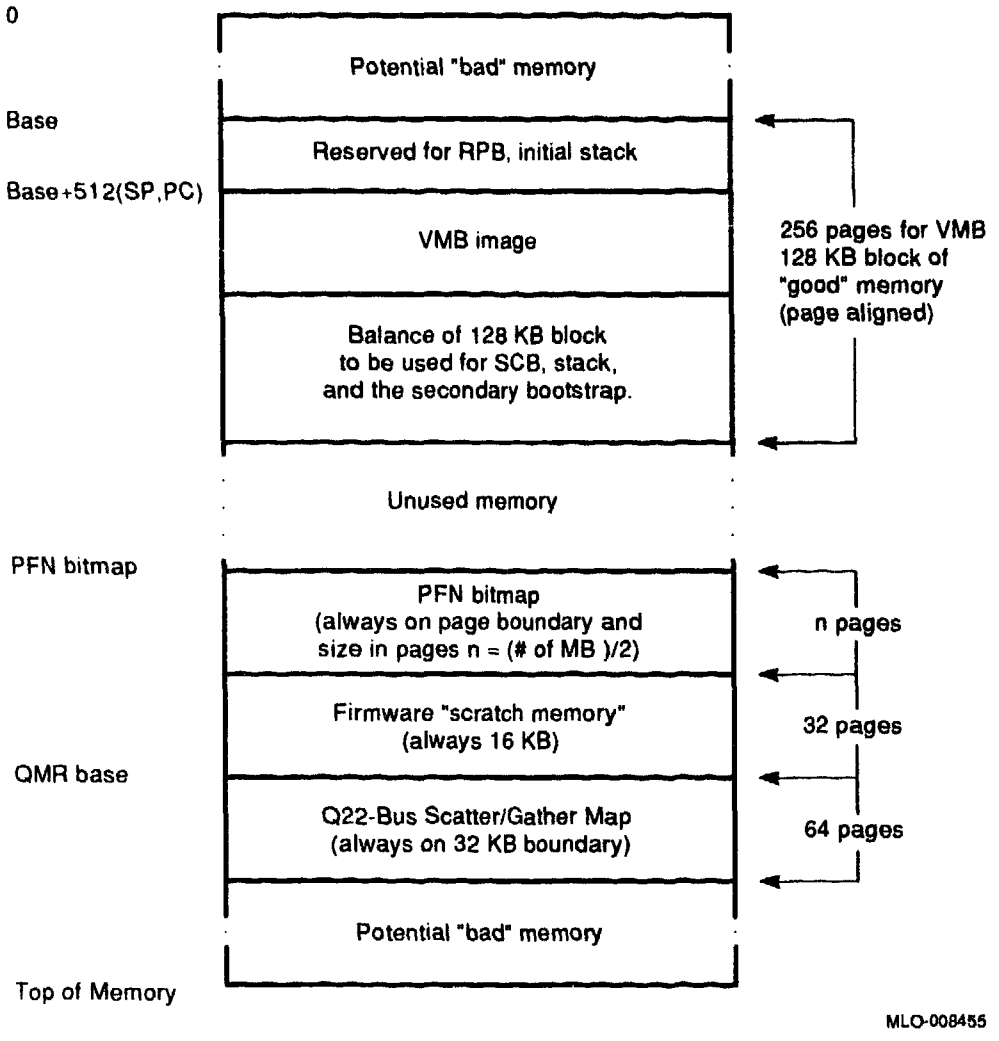
1. Check the console program mailbox "bootstrap in progress" bit (CPMBX<2>(BIP)). If it is set, bootstrap fails.
2. If this is an automatic bootstrap, display the message "Loading system software." on the console terminal.
3. Set CPMBX<2>(BIP).
4. Validate the Page Frame Number (PFN) bitmap. If PFN bitmap checksum is invalid, then:
   a. Perform an UNJAM.
   b. Perform an INIT.
   c. Retest memory and rebuild PFN bitmap.
5. Validate the boot device name. If none exists, supply a list of available devices and prompt user for a device. If no device is entered within 30 seconds, use EZA0.
6. Write a form of this BOOT request including the active boot flags and boot device on the console, for example "(BOOT/R5:0 DUA0)".
7. Initialize the Q22–bus scatter/gather map.
   a. Set IPCR<8>(AUX_HLT).
   b. Clear IPCR<5>(LMEAE).
   c. Perform an UNJAM.
   d. Perform an INIT.
   e. If an arbiter, map all vacant Q22–bus pages to the corresponding page in local memory and validate each entry if that page is "good".
   f. Set IPCR<5>(LMEAE).
8. Search for a 128K byte contiguous block of good memory as defined by the PFN bitmap. If 128K bytes cannot be found, the bootstrap fails.
9. Initialize the general purpose registers as follows:

R0	Address of descriptor of boot device name; 0 if none specified
R2	Length of PFN bitmap in bytes
R3	Address of PFN bitmap
R4	Time-of-day of bootstrap from PR$_TODR
R5	Boot flags
R10	Halt PC value
R11	Halt PSL value (without halt code and map enable)
AP	Halt code
SP	Base of 128-Kbyte good memory block + 512
PC	Base of 128-Kbyte good memory block + 512
R1, R6, R7, R8, R9, FP	0

10. Copy the VMB image from FEPROM to local memory beginning at the base of the 128 KB good memory block + 512.
11. Exit from the firmware to memory resident VMB.

On entry to VMB the processor is running at IPL 31 on the interrupt stack with memory management disabled. Also, local memory is partitioned as shown in Figure 4–3.

Figure 4–3 Memory Layout Prior to VMB Entry



4.7.2 Primary Bootstrap Procedures (VMB)

Virtual Memory Boot (VMB) is the primary bootstrap for booting VAX processors. On the KA50/51/55/56 module, VMB is resident in the firmware and is copied into main memory before control is transferred to it. VMB then loads the secondary bootstrap image and transfers control to it.

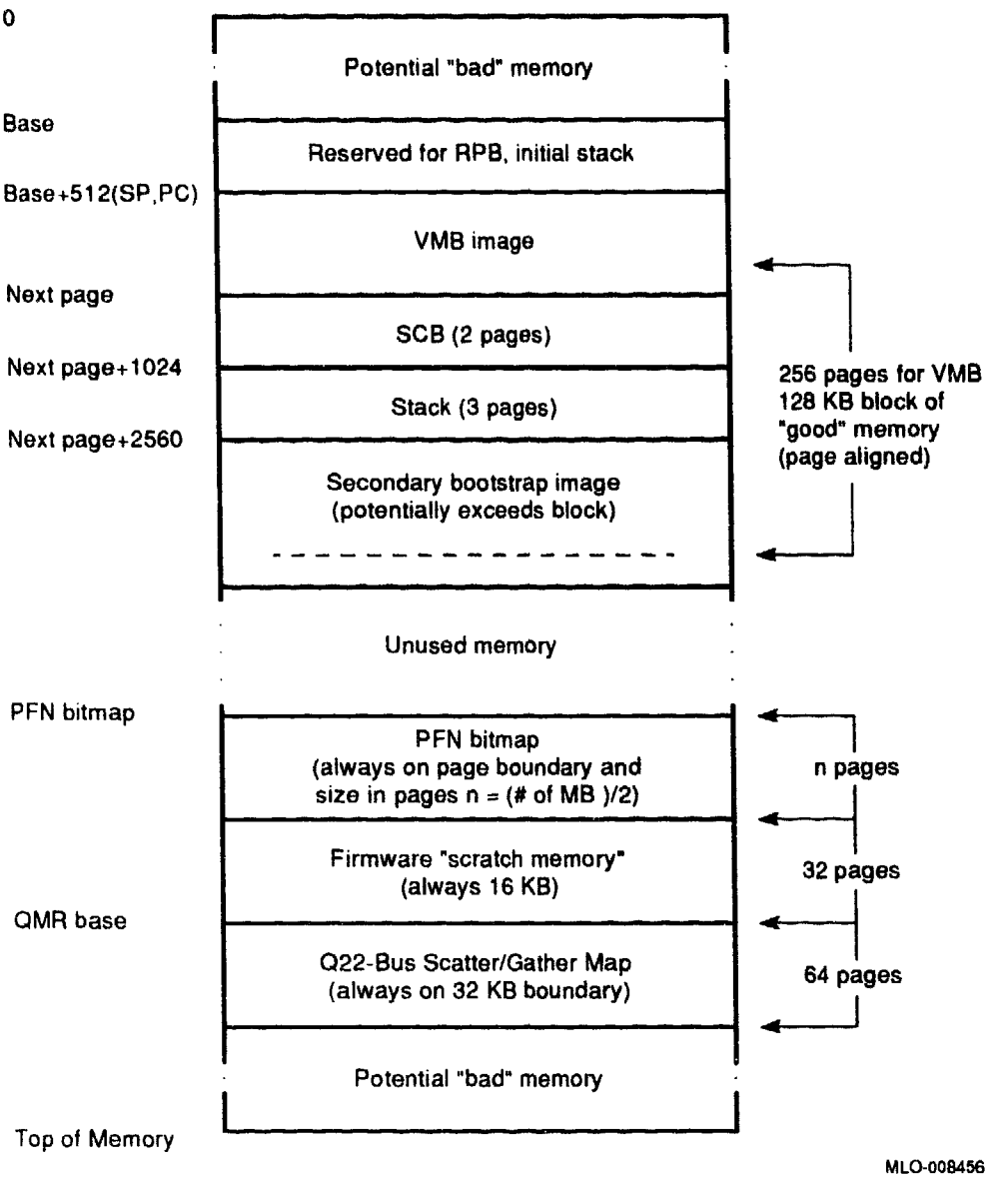
In certain cases, such as VAXELN, VMB actually loads the operating system directly. However, for the purpose of this discussion "secondary bootstrap" refers to any VMB loadable image.

VMB inherits a well defined environment and is responsible for further initialization. The following summarizes the operation of VMB.

1. Initialize a two-page SCB on the first-page boundary above VMB.
2. Allocate a three-page stack above the SCB.
3. Initialize the Restart Parameter Block (RPB).
4. Initialize the secondary bootstrap argument list.
5. If not a PROM boot, locate a minimum of three consecutive valid QMRs.
6. Write "2" to the diagnostic LEDs and display "2.." on the console to indicate that VMB is searching for the device.
7. Optionally, solicit from the console a "Bootfile: " name.
8. Write the name of the boot device from which VMB will attempt to boot on the console, for example, "-DIA0".
9. Copy the secondary bootstrap from the boot device into local memory above the stack. If this fails, the bootstrap fails.
10. Write "1" to the diagnostic LEDs and display "1.." on the console to indicate that VMB has found the secondary bootstrap image on the boot device and has loaded the image into local memory.
11. Clear CPMBX<2>(BIP) and CPMBX<3>(RIP).
12. Write "0" to the diagnostic LEDs and display "0.." on the console to indicate that VMB is now transferring control to the loaded image.
13. Transfer control to the loaded image with the following register usage.

R5	Transfer address in secondary bootstrap image
R10	Base address of secondary bootstrap memory
R11	Base address of RPB
AP	Base address of secondary boot parameter block
SP	Base address of secondary boot parameter block

Figure 4–4 Memory Layout at VMB Exit



In the event that an operating system has an extraordinarily large secondary bootstrap which overflows the 128 KB of "good" memory, VMB loads the remainder of the image in memory above the "good" block. However, if there are not enough contiguous "good" pages above the block to load the remainder of the image, the bootstrap fails.

4.7.3 Device Dependent Secondary Bootstrap Procedures

The following sections describe the various device dependent boot procedures.

4.7.3.1 Disk and Tape Bootstrap Procedure

The disk and tape bootstrap supports Files–11 lookup (supporting only the ODS level 2 file structure) or the boot block mechanism (used in PROM boot also). Of the standard DEC operating systems, OpenVMS and ELN use the Files–11 bootstrap procedure, and Ultrix-32 uses the boot block mechanism.

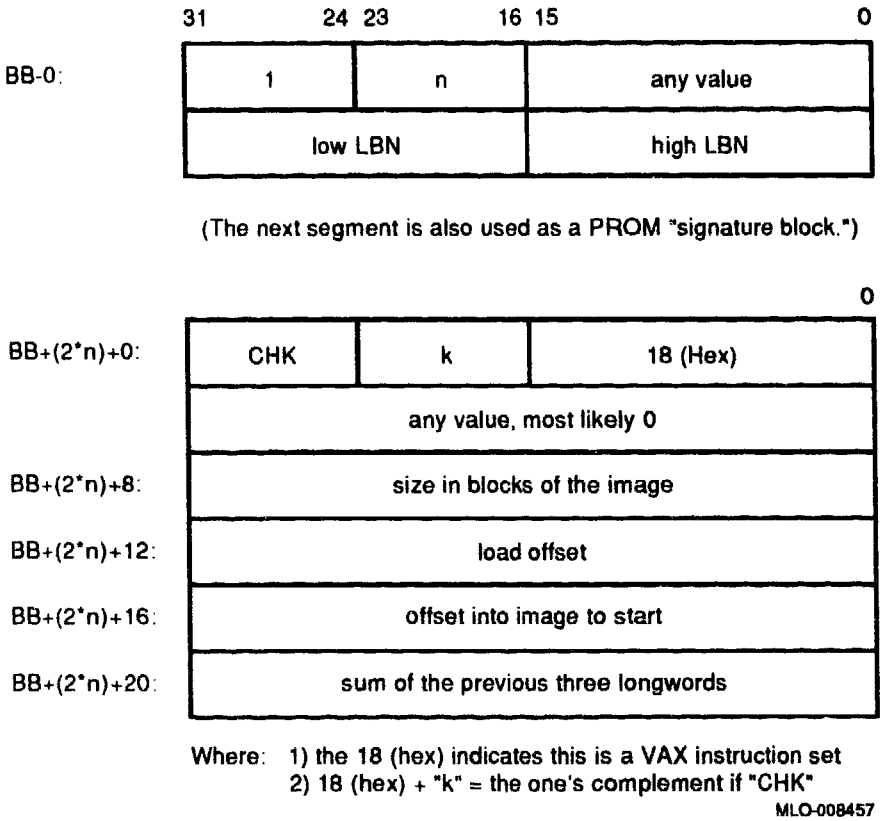
VMB first attempts a Files–11 lookup, unless the RPB$V_BBLOCK boot flag is set. If VMB determines that the designated boot disk is a Files–11 volume, it searches the volume for the designated boot program, usually [SYS0.SYSEXE]SYSBOOT.EXE. However, VMB can request a diagnostic image or prompt the user for an alternate file specification. If the boot image cannot be found, VMB fails.

If the volume is not a Files–11 volume or the RPB$V_BBLOCK boot flag was set, the boot block mechanism proceeds as follows:

1. Read logical block 0 of the selected boot device (this is the boot block).
2. Validate that the contents of the boot block conform to the boot block format (see below).
3. Use the boot block to find and read in the secondary bootstrap.
4. Transfer control to the secondary bootstrap image, just as for a Files–11 boot.

The format of the boot block must conform to that shown in Figure 4–5.

Figure 4–5 Boot Block Format



4.7.3.2 MOP Ethernet Functions and Network Bootstrap Procedure

Whenever a network bootstrap is selected on the KA50/51/55/56, the VMB code makes continuous attempts to boot from the network. VMB uses the DNA Maintenance Operations Protocol (MOP) as the transport protocol for network bootstraps and other network operations. Once a network boot has been invoked, VMB turns on the designated network link and repeats load attempt, until either a successful boot occurs, a fatal controller error occurs, or VMB is halted from the operator console.

The KA50/51/55/56 supports the load of a standard operating system, a diagnostic image, or a user-designated program via network bootstraps. The default image is the standard operating system, however, a user may select an alternate image by setting either the RPB$V_DIAG bit or the RPB$V_SOLICT bit in the boot flag longword R5. Note that the RPB$V_SOLICT bit has precedence over the RPB$V_DIAG bit. Hence, if both bits are set, then the solicited file is requested.

Note

VMB accepts a maximum 39 characters for a file specification for solicited boots. However, MOP V3 only supports a 15-character file name. If the network server is running the OpenVMS operating system, the following defaults apply to the file specification: the directory MOM$LOAD:, and the extension .SYS. Therefore, the file specification need only consist of the filename if the default directory and extension attributes are used.

The KA50/51/55/56 VMB uses the MOP program load sequence for bootstrapping the module and the MOP "dump/load" protocol type for load related message exchanges. The types of MOP message used in the exchange are listed in Table 4–3 and Table 4–4.

Table 4–3 Network Maintenance Operations Summary

Function	Role	Transmit		Receive
		MOP Ethernet and IEEE 802.3 Messages[1]		
Dump	Requester	—		—
	Server	—		—
Load	Requester	REQ_PROGRAM[2]	to solicit	VOLUNTEER
		REQ_MEM_LOAD	to solicit & ACK	MEM_LOAD
			or	MEM_LOAD_w_XFER
			or	PARAM_LOAD_w_XFER
	Server	—		—
Console	Requester	—		—
	Server	COUNTERS	in response to	REQ_COUNTERS
		SYSTEM_ID[3]	in response to	REQUEST_ID
				BOOT
Loopback	Requester	—		—
	Server	LOOPED_DATA[4]	in response to	LOOP_DATA
		IEEE 802.3 Messages[5]		
Exchange ID	Requester	—		—
	Server	XID_RSP	in response to	XID_CMD
Test	Requester	—		—

[1]All unsolicited messages are sent in Ethernet (MOP V3) and IEEE 802.2 (MOP V4), until the MOP version of the server is known. All solicited messages are sent in the format used for the request.

[2]The initial REQ_PROGRAM message is sent to the dumpload multicast address. If an assistance VOLUNTEER message is received, then the responder's address is used as the destination to repeat the REQ_PROGRAM message and for all subsequent REQ_MEM_LOAD messages.

[3]SYSTEM_ID messages are sent out every 8 to 12 minutes to the remote console multicast address and, on receipt of a REQUEST_ID message, they are sent to the initiator.

[4]LOOPED_DATA messages are sent out in response to LOOP_DATA messages. These messages are actually in Ethernet LOOP TEST format, not in MOP format, and when sent in Ethernet frames, omit the additional length field (padding is disabled).

[5]IEEE 802.2 support of XID and TEST is limited to Class 1 operations.

(continued on next page)

Table 4–3 (Cont.) Network Maintenance Operations Summary

Function	Role	Transmit		Receive
		IEEE 802.3 Messages[5]		
	Server	TEST_RSP	in response to	TEST_CMD

[5]IEEE 802.2 support of XID and TEST is limited to Class 1 operations.

Table 4–4 Supported MOP Messages

Message Type	Message Fields						
	DUMP/LOAD						
MEM_LOAD_w_ XFER	Code 00	Load # nn	Load addr aa-aa-aa-aa		Image data None		Xfer addr aa-aa-aa-aa
MEM_LOAD	Code 02	Load # nn	Load addr aa-aa-aa-aa		Image data dd-...		
REQ_PROGRAM	Code 08	Device 25 LQA 49 SGEC	Format 01 V3 04 V4	Program 02 Sys	SW ID[3] C-17 [1] C-128 [2] If C[1] >00 Len 00 No ID FF OS FE Maint	Procesr 00 Sys	Info (see SYSTEM_ ID)
REQ_MEM_ LOAD	Code 0A	Load # nn	Error ee				

[1]MOP V3.0 only.

[2]MOP x4.0 only.

[3]Software ID field is loaded from the string stored in the 40-byte field, RPB$T_FILE, of the RPB on a solicited boot.

(continued on next page)

Table 4–4 (Cont.) Supported MOP Messages

Message Type	Message Fields						
DUMP/LOAD							
PARM_LOAD_w_ XFER	Code 14	Load # nn	Prm typ 01 02 03 04 05 06 00 End	Prm len I-16 I-06 I-16 I-06 0A 08	Prm val Target name [1] Target addr [1] Host name [1] Host addr [1] Host time [1] Host time [2]		Xfer addr aa-aa-aa-aa
VOLUNTEER	Code 03						
REMOTE CONSOLE							
REQUEST_ID	Code 05	Rsrvd xx	Recpt # nn-nn				
SYSTEM_ID	Code 07	Rsrvd xx	Recpt # nn-nn or 00-00	Info type 01-00 Version 02-00 Functions 07-00 HW addr 64-00 Device 90-01 Datalink 91-01 Bufr size		Info len 03 02 06 01 01 02	Info value 04-00-00 00-59 ee-ee-ee-ee-ee-ee 25 or 49 01 06-04
REQ_COUNTERS	Code 09	Recpt # nn-nn					
COUNTERS	Code 0B	Recpt # nn-nn	Counter block				
BOOT [4]	Code 06	Verification vv-vv-vv-vv-vv-vv-vv-vv	Procesr 00 Sys	Control xx	Dev ID C-17	SW ID [3] (see REQ_PROGRAM)	Script ID [2] C-128

[1]MOP V3.0 only.

[2]MOP x4.0 only.

[3]Software ID field is loaded from the string stored in the 40-byte field, RPB$T_FILE, of the RPB on a solicited boot.

[4]A BOOT message is not verified, because in this context, a boot is already in progress. However, a received BOOT message will cause the boot backoff timer to be reset to it's minimum value.

(continued on next page)

Table 4–4 (Cont.) Supported MOP Messages

Message Type	Message Fields				
LOOPBACK					
LOOP_DATA	Skpcnt nn-nn	Skipped bytes bb-...	Function 00-02 Forward data	Forward addr ee-ee-ee-ee-ee-ee	Data dd-...
LOOPED_DATA	Skpcnt nn-nn	Skipped bytes bb-...	Function 00-01 Reply	Recpt # nn-nn	Data dd-...
IEEE 802.2					
XID_CMD/RSP	Form 81	Class 01	Rx window size (K) 00		
TEST_CMD/RSP	Optional data				

VMB, the requester, starts by sending a REQ_PROGRAM message to the MOP 'dump/load' multicast address. It then waits for a response in the form of a VOLUNTEER message from another node on the network, the MOP server. If a response is received, then the destination address is changed from the multicast address to the node address of the server and the same REQ_PROGRAM message is retransmitted to the server as an Acknowledge.

Next, VMB begins sending REQ_MEM_LOAD messages to the server. The server responds with either:

- MEM_LOAD message, while there is still more to load.
- MEM_LOAD_w_XFER, if it is the end of the image.
- PARAM_LOAD_w_XFER, if it is the end of the image and operating system parameters are required.

The "load number" field in the load messages is used to synchronize the load sequence. At the beginning of the exchange, both the requester and server initialize the load number. The requester only increments the load number if a load packet has been successfully received and loaded. This forms the Acknowledge to each exchange. The server will resend a packet with a specific load number, until it sees the load number incremented. The final Acknowledge is sent by the requester and has a load number equivalent to the load number of the appropriate LOAD_w_XFER message + 1.

Because the request for load assistance is a MOP "must transact" operation, the network bootstrap continues indefinitely until a volunteer is found. The REQ_PROGRAM message is sent out in bursts of eight at four second intervals, the first four in MOP Version four IEEE 802.3 format and the last four in MOP Version 3 Ethernet format. The backoff period between bursts doubles each cycle from an initial value of four seconds, to eight seconds,... up to a maximum of five minutes. However, to reduce the likelihood of many nodes posting requests in lock-step, a random "jitter" is applied to the backoff period. The actual backoff time is computed as (.75+(.5*RND(x)))*BACKOFF, where 0<=x<1.

4.7.3.3 Network "Listening"

While the CPU module is waiting for a load volunteer during bootstrap, it "listens" on the network for other maintenance messages directed to the node and periodically identifies itself at the end of each 8- to 12-minute interval before a bootstrap retry. In particular, this "listener" supplements the Maintenance Operation Protocol (MOP) functions of the VMB load requester typically found in bootstrap firmware and supports.

- A remote console server that generates COUNTERS messages in response to REQ_COUNTERS messages, unsolicited SYSTEM_ID messages every 8 to 12 minutes, and solicited SYSTEM_ID messages in response to REQUEST_ID messages, as well as recognition of BOOT messages.
- A loopback server that responds to Ethernet loopback messages by echoing the message to the requester.
- An IEEE 802.2 responder that replies to both XID and TEST messages.

During network bootstrap operation, the KA50/51/55/56 complies with the requirements defined in the "NI Node Architecture Specification" for a primitive node. The firmware listens only to MOP "Load/Dump", MOP "Remote Console", Ethernet "Loopback Assistance", and IEEE 802.3 XID/TEST messages (listed in Table 4–5) directed to the Ethernet physical address of the node. All other Ethernet protocols are filtered by the network device driver.

The MOP functions and message types, which are supported by the KA50/51 /55/56, are summarized in Tables 4–3 and 4–5.

Table 4–5 MOP Multicast Addresses and Protocol Specifiers

Function	Address	IEEE Prefix[1]	Protocol	Owner
Dump/Load	AB-00-00-01-00-00	08-00-2B	60-01	Digital
Remote console	AB-00-00-02-00-00	08-00-2B	60-02	Digital
Loopback assistance	CF-00-00-00-00-00[2]	08-00-2B	90-00	Digital

[1]MOP V4.0 only.
[2]Not used.

4.8 Operating System Restart

An operating system restart is the process of bringing up the operating system from a known initialization state following a processor halt. This procedure is often called *restart* or *warmstart*, and should not be confused with a processor restart which results in firmware entry.

On the KA50/51/55/56, a restart occurs if the conditions specified in Table G–1 are satisfied.

To restart a halted operating system, the firmware searches system memory for the Restart Parameter Block (RPB), a data structure constructed for this purpose by VMB. (Refer to Table C–2 in Appendix C for a detailed description of this data structure.) If a valid RPB is found, the firmware passes control to the operating system at an address specified in the RPB.

The firmware keeps a "restart in progress" (RIP) flag in CPMBX which it uses to avoid repeated attempts to restart a failing operating system. An additional "restart in progress" flag is maintained by the operating system in the RPB.

The firmware uses the following algorithm to restart the operating system:

1. Check CPMBX<3>(RIP). If it is set, restart fails.
2. Print the message "Restarting system software." on the console terminal.
3. Set CPMBX<3>(RIP).
4. Search for a valid RPB. If none is found, restart fails.
5. Check the operating system RPB$L_RSTRTFLG<0>(RIP) flag. If it is set, restart fails.
6. Write "0" on the diagnostic LEDs.

7. Dispatch to the restart address, RPB$L_RESTART, with:

SP	Physical address of the RPB plus 512
AP	Halt code
PSL	041F0000
PR$_MAPEN	0

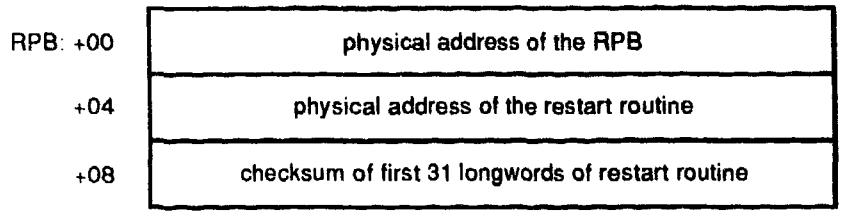
If the restart is successful, the operating system must clear CPMBX<3>(RIP).

If restart fails, the firmware prints "Restart failure." on the system console.

4.8.1 Locating the RPB

The RPB is a page-aligned control block which can be identified by the first three longwords. The format of the RPB "signature" is shown in Figure 4–6. (Refer to Table C–2 in Appendix C for a complete description of the RPB.)

Figure 4–6 Locating the Restart Parameter Block



The firmware uses the following algorithm to find a valid RPB:

1. Search for a page of memory that contains its address in the first longword. If none is found, the search for a valid RPB has failed.
2. Read the second longword in the page (the physical address of the restart routine). If it is not a valid physical address, or if it is zero, return to step 1. The check for zero is necessary to ensure that a page of zeros does not pass the test for a valid RPB.
3. Calculate the 32 bit twos-complement sum (ignoring overflows) of the first 31 longwords of the restart routine. If the sum does not match the third longword of the RPB, return to step 1.
4. A valid RPB has been found.

5

System Troubleshooting and Diagnostics

This chapter provides troubleshooting information for the two primary diagnostic methods: online, interpreting error logs to isolate the FRU; and offline, interpreting ROM-based diagnostic messages to isolate the FRU.

In addition, the chapter provides information on using MOP Ethernet functions to isolate errors, and interpreting UETP failures.

The chapter concludes with a section on running loopback tests to test the console port and embedded Ethernet ports.

Note

The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95, and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, may appear on the console and/or printouts from time to time.

5.1 Basic Troubleshooting Flow

Before troubleshooting any system problem, check the site maintenance log for the system's service history. Be sure to ask the system manager the following questions:

- Has the system been used before and did it work correctly?
- Have changes (changes to hardware, updates to firmware or software) been made to the system recently?
- What is the state of the system—is it on line or off line?

  If the system is off line and you are not able to bring it up, use the offline diagnostic tools, such as RBDs, MDM, and LEDs.

If the system is on line, use the online diagnostic tools, such as error logs, crash dumps, UETP, and other log files.

Four common problems occur when you make a change to the system:

- Incorrect cabling
- Module configuration errors (incorrect CSR addresses and interrupt vectors)
- Incorrect grant continuity
- Incorrect bus node ID plugs

In addition, check the following:

- If you have received error notification using VAXsimPLUS, check the mail messages and error logs as described in Section 5.2.
- If the operating system fails to boot (or appears to fail), check the console terminal screen for an error message. If the terminal displays an error message, see Section 5.3.
- Check the LEDs on the device you suspect is bad. If no errors are indicated by the device LEDs, run the ROM-based diagnostics described in this chapter.
- If the system boots successfully, but a device seems to fail or an intermittent failure occurs, check the error log ([SYSERR]ERRLOG.SYS) as described in Section 5.2.
- For fatal errors, check that the crash dump file exists for further analysis ([SYSEXE]SYSDUMP.DMP).
- Check other log files, such as OPERATOR.LOG, OPCOM.LOG, SETHOST.LOG, etc. Many of these can be found in the [SYSMGR] account. SETHOST.LOG is useful in comparing the console output with event logs and crash dumps in order to see what the system was doing at the time of the error.

  Use the following command to create SETHOST.LOG files, then log into the system account.

  ```
  $ SET HOST/LOG 0
  ```

  After logging out this file will reside in the [SYSMGR] account.

  If the system is failing in the boot or start-up phase, it may be useful to include the command SET VERIFY in the front of various start-up .COM files to obtain a trace of the start-up commands and procedures.

When troubleshooting, note the status of cables and connectors before you perform each step. Label cables before you disconnect them. This step saves you time and prevents you from introducing new problems.

Most communications modules use floating CSR addresses and interrupt vectors. If you remove a module from the system, you may have to change the addresses and vectors of other modules.

If you change the system configuration, run the CONFIGURE utility at the console I/O prompt (>>>) to determine the CSR addresses and interrupt vectors recommended by Digital.

5.2 Product Fault Management and Symptom-Directed Diagnosis

This section describes how errors are handled by the microcode and software, how the errors are logged, and how, through the Symptom-Directed Diagnosis (SDD) tool, VAXsimPLUS, errors are brought to the attention of the user. This section also provides the service theory used to interpret error logs to isolate the FRU. Interpreting error logs to isolate the FRU is the primary method of diagnosis.

5.2.1 General Exception and Interrupt Handling

This section describes the first step of error notification: the errors are first handled by the microcode and then are dispatched to the OpenVMS error handler.

The kernel uses the NVAX core chipset: NVAX CPU, NVAX Memory Controller (NMC), and NDAL to CDAL adapter (NCA).

Internal errors within the NVAX CPU result in machine check exceptions, through System Control Block (SCB) vector 004, or soft error interrupts at Interrupt Priority Level (IPL) 1A, SCB vector 054 hex.

External errors to the NVAX CPU, which are detected by the NMC or NDAL to CDAL adapter (NCA), usually result in these chips posting an error condition to the NVAX CPU. The NVAX CPU will then generate a machine check exception through SCB vector 004, hard error interrupt, IPL 1D, through SCB vector 060 (hex), or a soft error interrupt through SCB vector 054.

External errors to the NMC and NCA, which are detected by chips on the CDAL busses for transactions which originated by the NVAX CPU, are typically signaled back to the NCA adapter. The NCA adapter will post an error signal back to the NVAX CPU which generates a machine check or high level interrupt.

In the case of Direct Memory Access (DMA) transactions where the NCA or NMC detects the error, the errors are typically signaled back to the CDAL-Bus device, but not posted to the NVAX CPU. In these cases the CDAL-Bus device typically posts a device level interrupt to the NVAX CPU via the NCA. In almost all cases, error state is latched by the NMC and NCA. Although these errors will not result in a machine check exception or high level interrupt (i.e. results in device level IPL 14–17 versus error level IPL 1A, 1D), the OpenVMS machine check handler has a polling routine that will search for this state at one-second intervals. This will result in the host logging a polled error entry.

These conditions cover all of the cases that will eventually be handled by the OpenVMS error handler. The OpenVMS error handler will generate entries that correspond to the machine check exception, hard or soft error interrupt type, or polled error.

5.2.2 OpenVMS Error Handling

Upon detection of a machine check exception, hard error interrupt, soft error interrupt or polled error, the OpenVMS operating ststem will perform the following actions:

- Snapshot the state of the kernel.
- In most entry points, disable the caches.
- If it is a machine check and if the machine check is recoverable, determine if instruction retry is possible.

  Instruction retry is possible if one of the following conditions is true:

  - If PCSTS <10>PTE_ER = 0:

    Check that (ISTATE2 <07>VR = 1) or (PSL <27> FPD = 1)

    Otherwise crash the system or process depending on PSL <25:24> Current Mode.

  - If PCSTS <10>PTE_ER = 1:

    Check that (ISTATE2 <07>VR = 1) and (PSL <27>FPD = 0) and (PCSTS <09>PTE_ER_WR = 0)

    Otherwise crash the system.

  ISTATE2 is a longword in the machine check stack frame at offset (SP)+24; PSL is a longword in the machine check stack frame at offset (SP)+32; VR is the VAX Restart flag; and FPD is the First Part Done flag.
- Check to see if the threshold has been exceeded for various errors (typically the threshold is exceeded if 3 errors occur within a 10 minute interval).

- If the threshold has been exceeded for a particular type of cache error, mark a flag that will signify that this resource is to be disabled (the cache will be disabled in most, but not all, cases).
- Update the SYSTAT software register with results of error/fault handling.
- For memory uncorrectable Error Correction Code (ECC) errors:
  - If machine check, mark page bad and attempt to replace page.
  - Fill in MEMCON software register with memory configuration and error status for use in FRU isolation.
- For memory single-bit correctable ECC errors:
  - Fill in Corrected Read Data (CRD) entry FOOTPRINT with set, bank, and syndrome information for use in FRU isolation.
  - Update the CRD entry for time, address range, and count; fill the MEMCON software register with memory configuration information.
  - Scrub memory location for first occurrence of error within a particular footprint. If second or more occurrence within a footprint, mark page bad in hopes that page will be replaced later. Disable soft error logging for 10 minutes if threshold is exceeded.
  - Signify that CRD buffer be logged for the following events: system shutdown (operator shutdown or crash), hard single-cell address within footprint, multiple addresses within footprint, memory uncorrectable ECC error, or CRD buffer full.
- For ownership memory correctable ECC error, scrub location.
- Log error.
- Crash process or system, dependent upon PSL (Current Mode) with a fatal bugcheck for the following situations:
  - Retry is not possible.
  - Memory page could not be replaced for uncorrectable ECC memory error.
  - Uncorrectable tag store ECC errors present in writeback cache.
  - Uncorrectable data store ECC errors present in writeback cache for locations marked as OWNED.
  - Most INT60 errors.
  - Threshold is exceeded (except for cache errors).

  - A few other errors of the sort considered nonrecoverable are present.
- Disable cache(s) permanently if error threshold is exceeded.
- Flush and re-enable those caches which have been marked as good.
- Clear the error flags.
- Perform Return from Exception or Interrupt (REI) to recover and restart or continue the instruction stream for the following situations:
  - Most INT54 errors.
  - Those INT60 and INT54 errors which result in bad ECC written to a memory location. (These errors can provide clues that the problem is not memory related.)
  - Machine check conditions where instruction retry is possible.
  - Memory uncorrectable ECC error where page replacement is possible and instruction retry is possible.
  - Threshold exceeded (for cache errors only).
  - Return from Subroutine (RSB) and return from all polled errors.

Note

The results of the OpenVMS error handler may be preserved within the operating system session (for example, disabling a cache) but not across reboots.

Although the system can recover with cache disabled, the system performance will be degraded, since access time increases as available cache decreases.

5.2.3 OpenVMS Error Logging and Event Log Entry Format

The OpenVMS error handler for the kernel can generate six different entry types, as shown in Table 5–1. All error entry types, with the exception of correctable ECC memory errors, are logged immediately.

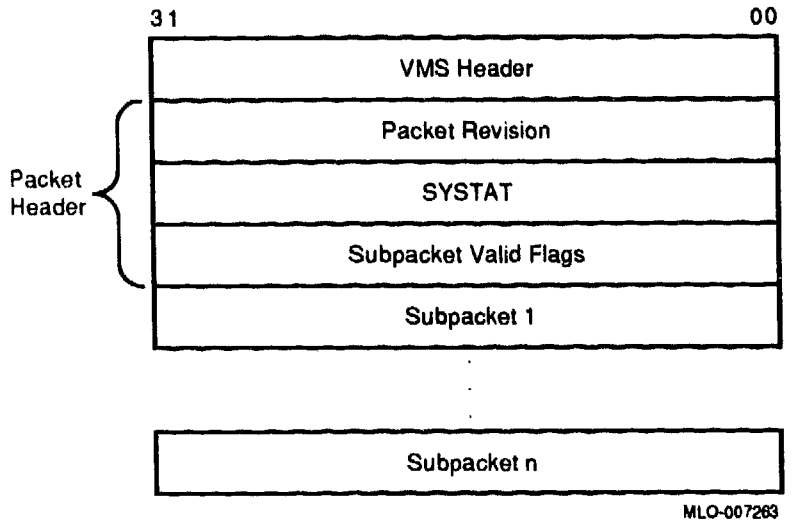
Table 5–1 OpenVMS Error Handler Entry Types

OpenVMS Entry Type	Code	Description
EMB$C_MC	(002.)	Machine Check Exception SCB Vector 4, IPL 1F
EMB$C_SE	(006.)	Soft Error Interrupt Correctable ECC Memory Error SCB Vector 54, IPL 1A
EMB$C_INT54	(026.)	Soft Error Interrupt SCB Vector 54, IPL 1A
EMB$C_INT60	(027.)	Hard Error Interrupt 60 SCB Vector 60, IPL 1D
EMB$C_POLLED	(044.)	Polled Errors No exception or interrupt generated by hardware.
EMB$C_BUGCHECK		Fatal bugcheck Bugcheck Types: MACHINECHK ASYNCWRTER BADMCKCOD INCONSTATE UNXINTEXC

Each entry consists of an OpenVMS operating system header, a packet header, and one or more subpackets (Figure 5–1). Entries can be of variable length based on the number of subpackets within the entry. The FLAGS software register in the packet header shows which subpackets are included within a given entry.

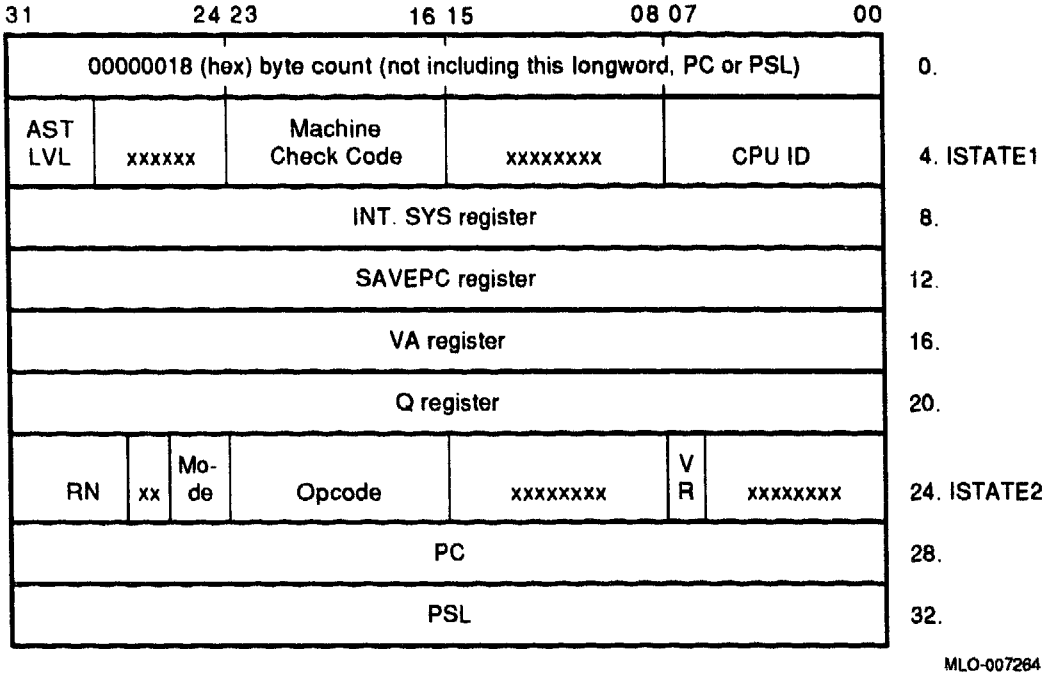
Refer to Section 5.2.4 for actual examples of the error and event logs described throughout this section.

Figure 5–1 Event Log Entry Format



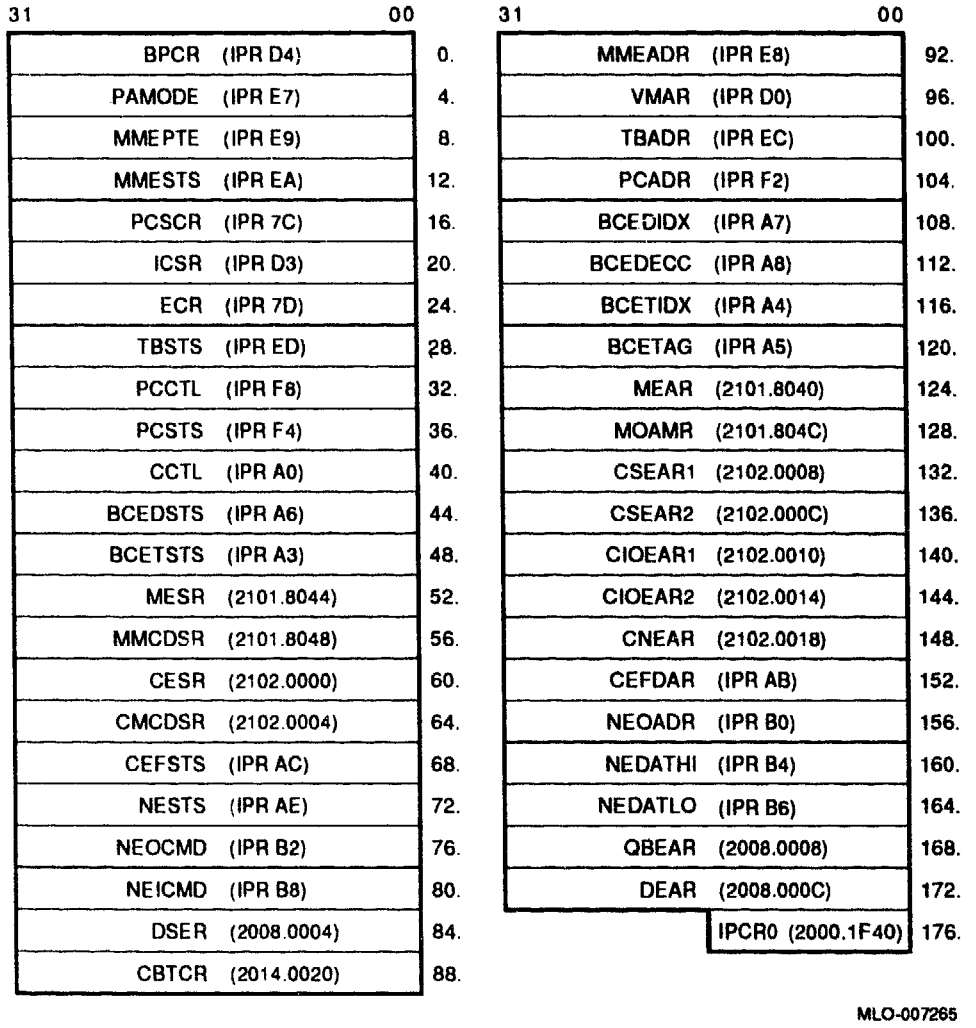
Machine check exception entries contain, at a minimum, a Machine Check Stack Frame subpacket (Figure 5–2).

Figure 5–2 Machine Check Stack Frame Subpacket



INT54, INT60, Polled, and some Machine Check entries contain a processor Register subpacket (Figure 5–3), which consists of some 40 plus hardware registers.

Figure 5–3 Processor Register Subpacket



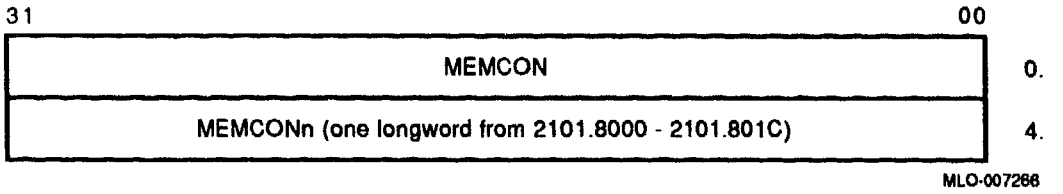
Note

The byte count, although part of the stack frame, is not included in the error log entry itself.

Bugcheck entries generated by the OpenVMS kernel error handler include the first 23 registers from the processor Register subpacket along with the Time of Day Register (TODR) and other software context states.

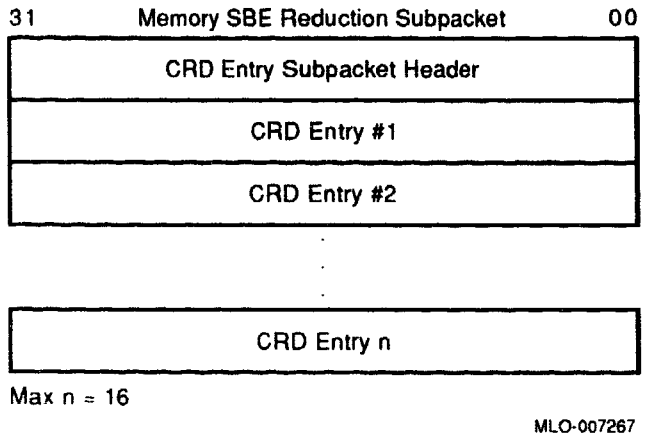
Uncorrectable ECC memory error entries include a Memory subpacket (Figure 5–4). The memory subpacket consists of MEMCON, which is a software register containing the memory configuration and error status used for FRU isolation, and MEMCONn, the hardware register that matched the error address in MEAR.

Figure 5–4 Memory Subpacket for ECC Memory Errors



Correctable Memory Error entries have a Memory (Single-Bit Error) SBE Reduction subpacket (Figure 5–5). This subpacket, unlike all others, is of variable length. It consists solely of software registers from state maintained by the error handler, as well as hardware state transformed into a more usable format.

Figure 5–5 Memory SBE Reduction Subpacket (Correctable Memory Errors)

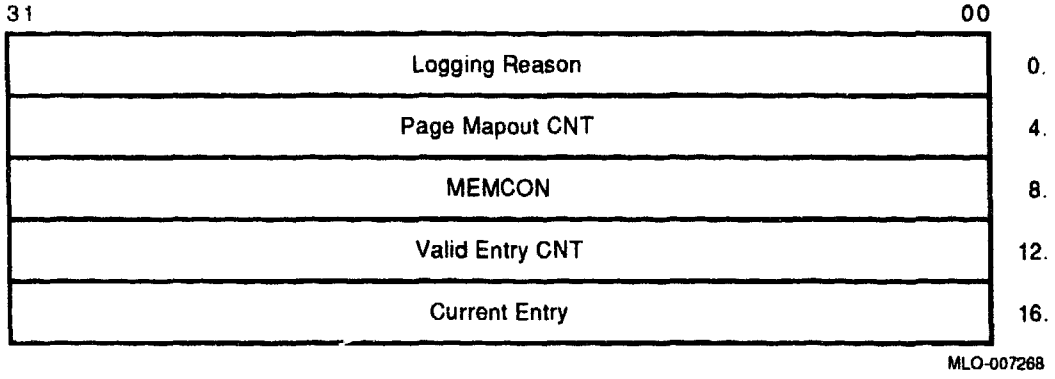


The OpenVMS error handler maintains a Correctable Read Data (CRD) buffer internally within memory that is flushed asynchronously for high-level events to the error log file. The CRD buffer and resultant error log entry are maintained and organized as follows.

- Each entry has a subpacket header (Figure 5–6) consisting of LOGGING REASON, PAGE MAPOUT CNT, MEMCON, VALID ENTRY CNT, and

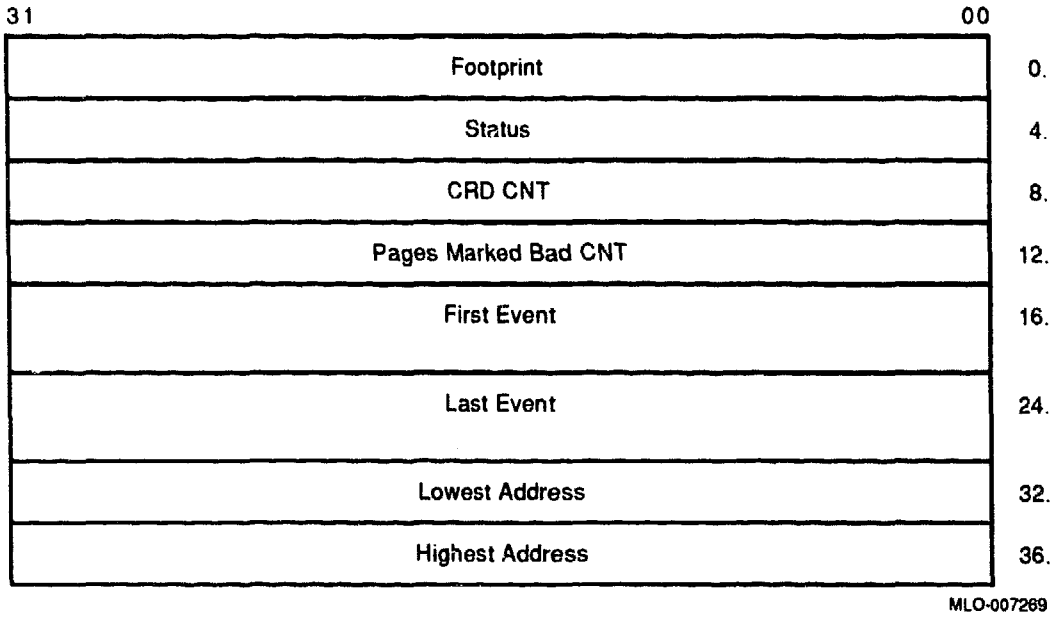
CURRENT ENTRY. MEMCON contains memory configuration information, but no error status as is done for the Memory subpacket.

Figure 5–6 CRD Entry Subpacket Header



- Following the subpacket header are 1 to 16 fixed-length Memory CRD Entries (Figure 5–7). The number of Memory CRD entries is shown in VALID ENTRY CNT. The entry which caused the report to be generated is in CURRENT ENTRY.

Figure 5–7 Correctable Read Data (CRD) Entry



Each Memory CRD Entry represents one unique DRAM within the memory subsystem. A unique set, bank, and syndrome are stored in footprint to construct a unique ID for the DRAM.

Rather than logging an error for each occurrence of a single symbol correctable ECC memory error, the OpenVMS error handler maintains the CRD buffer—it creates a Memory CRD Entry for new footprints and updates an existing Memory CRD Entry for errors that occur within the range specified by the ID in FOOTPRINT. This reduces the amount of data logged overall without losing important information—errors are logged per unique failure mode rather than on a per error basis.

Each Memory CRD entry consists of a FOOTPRINT, STATUS, CRD CNT, PAGE MAPOUT CNT, FIRST EVENT, LAST EVENT, LOWEST ADDRESS and HIGHEST ADDRESS.

FIRST EVENT, LAST EVENT, LOWEST ADDRESS and HIGHEST ADDRESS are updated to show the range of time and addresses of errors which have occurred for a DRAM. CRD CNT is simply the total count per footprint. PAGE MAPOUT CNT is the number of pages that have been marked bad for a particular DRAM.

STATUS contains a record of the failure mode status of a particular DRAM over time. This in turn determines whether or not the CRD buffer is logged. For the first occurrence of an error within a particular DRAM, the memory location will be scrubbed (corrected read data is read, then written back to the memory location) and CRD CNT will be set to 1. Since most memory single-bit errors are transient due to alpha particles, logging of the CRD buffer will not be done immediately for the first occurrence of an error within a DRAM. The CRD buffer will, however, be logged at the time of system shutdown (operator or crash induced), or when a more severe memory subsystem error occurs.

If the FOOTPRINT/DRAM experiences another error (CRD CNT > 1), the OpenVMS operating system will set HARD SINGLE ADDRESS or MULTIPLE ADDRESSES along with SCRUBBED in STATUS. Scrubbing is no longer performed; instead, pages are marked bad. In this case, the OpenVMS operating system will log the CRD buffer immediately. The CRD Buffer will also be logged immediately if PAGE MAPOUT THRESHOLD EXCEEDED is set in SYSTAT as a result of pages being marked bad. The threshold is reached if more than one page per Mbyte of system memory is marked bad.

Note

CURRENT ENTRY will be zero in the Memory SBE Reduction subpacket header if the CRD buffer was logged, not as a result of a HARD SINGLE ADDRESS or MULTIPLE ADDRESSES error in STATUS, but as a result of a memory uncorrectable ECC error shown as RELATED ERROR, or as a result of CRD BUFFER FULL or SYSTEM SHUTDOWN, all of which are shown under LOGGING REASON.

5.2.4 OpenVMS Event Record Translation

The kernel error log entries are translated from binary to ASCII using the ANALYZE/ERROR command. To invoke the error log utility, enter the DCL command ANALYZE/ERROR_LOG.

Format:

ANALYZE_ERROR_LOG [/qualifier(s)] [file-spec] [,...]

Example:

$ ANALYZE/ERROR_LOG/INCLUDE=(CPU,MEMORY)/SINCE=TODAY

The error log utility translates the entry into the traditional three-column format. The first column shows the register mnemonics, the second column depicts the data in hex, and the last column shows the actual English translations.

As in the above example, the OpenVMS error handler also provides support for the /INCLUDE qualifier, such that CPU and MEMORY error entries can be selectively translated.

Since most kernel errors are bounded to either the processor module/system board or memory modules, the individual error flags and fields are not covered by the service theory. Although these flags are generally not required to diagnose a system to the FRU (Field Replaceable Unit), this information can be useful for component isolation.

ERF bit to text translation highlights all error flags that are set, and other significant state—these are displayed in capital letters in the third column. Otherwise, nothing is shown in the translation column. The translation rules also have qualifiers such that if the setting of an error flag causes other registers to be latched, the other registers will be translated as well. For example, if a memory ECC error occurs, the syndrome and error address fields will be latched as well. If such a field is valid, the translation will be shown (e.g. MEMORY ERROR ADDRESS); otherwise, no translation is provided.

5.2.5 Interpreting CPU Faults Using ANALYZE/ERROR

If the following three conditions are satisfied, the most likely FRU is the CPU module. Example 5–1 shows an abbreviated error log with numbers to highlight the key registers.

❶ No memory subpacket is listed in the third column of the FLAGS register.

❷ CESR register bit <09>, CP2 IO Error, is equal to zero in the KA50/51/55 /56 Register Subpacket.

❸ DSER register bits <07>, Q22 Bus NXM, <05>, Q22 Bus Device Parity Error, or <02>, Q-22 Bus No Grant, are equal to zero in the KA50/51/55/56 Register Subpacket.

The FLAGS register is located in the packet header, which immediately follows the system identification header; the CESR and DSER registers are listed under the KA50/51/55/56 Register Subpacket.

CPU errors will increment an OpenVMS global counter, which can be viewed using the DCL command SHOW ERROR, as shown in Example 5–2.

To determine if any resources have been disabled, for example, if cache has been disabled for the duration of the OpenVMS session, examine the flags for the SYSTAT register in the packet header.

In Example 5–1, a translation buffer data parity error latched in the TBSTS register caused a machine check exception error.

Example 5–1 Error Log Entry Indicating CPU Error

```
V A X / V M S        SYSTEM ERROR REPORT        COMPILED 14-JAN-1992 18:55:52
                                                                 PAGE   1.
******************************** ENTRY      1. ********************************
ERROR SEQUENCE 11.                              LOGGED ON:        SID 13001401
DATE/TIME 27-SEP-1991 14:40:10.85                                 SYS_TYPE 03110A01
SYSTEM UPTIME: 0 DAYS 00:12:12
SCS NODE: OMEGA1                                                  VAX/OpenVMS V5.5-2

MACHINE CHECK KA50  CPU Microcode Rev # 1.  CONSOLE FW REV# 1.1
                    Standard Microcode Patch    Patch Rev # 10.

      REVISION        00000000
      SYSTAT          00000001
                                    ATTEMPTING RECOVERY
      FLAGS           00000003
                                    machine check stack frame
                                    KA50 subpacket

 STACK FRAME SUBPACKET

      ISTATE_1        80050000
                                    MACHINE CHECK FAULT CODE = 05(x)
                                    Current AST level  = 4(X)
                                    ASYNCHRONOUS HARDWARE ERROR
      .
      .
      .
      PSL             04140001
                                    c-bit
                                    executing on interrupt stack
                                    PSL previous mode = kernel
                                    PSL current mode = kernel
                                    first part done set

 KA50 REGISTER SUBPACKET
```

(continued on next page)

Example 5–1 (Cont.) Error Log Entry Indicating CPU Error

```
BPCR            ECC80024
.
.
.
TBSTS           800001D3
                                LOCK SET
                                TRANSLATION BUFFER DATA PARITY ERROR
                                em_latch invalid
                                s5_command  = 1D(X)
                                valid Ibox specifier ref. error stored
.
.
.
CESR            00000000 ●
.
.
.
DSER            00000000 ●
.
.
.
IPCR0           00000020
                                LOCAL MEMORY EXTERNAL ACCESS ENABLED
```

Note

Ownership (O-bit) memory correctable or fatal errors (MESR <04> or MESR <03> of the processor Register Subpacket set equal to 1) are processor module errors, NOT memory errors.

Example 5–2 SHOW ERROR Display Using the OpenVMS Operating System

```
$ SHOW ERROR
Device                          Error Count
CPU                                 1
MEMORY                              1
PAB0:                               1
PAA0:                               1
PTA0:                               1
RTA2:                               1
$
```

5.2.6 Interpreting Memory Faults Using ANALYZE/ERROR

If "memory subpacket" or "memory sbe reduction subpacket" is listed in the third column of the FLAGS register, there is a problem with one or more of the memory modules, CPU module, or backplane.

- The "memory subpacket" message indicates an uncorrectable ECC error. Refer to Section 5.2.6.1 for instructions in isolating uncorrectable ECC error problems.
- The "memory sbe reduction subpacket" message indicates correctable ECC errors. Refer to Section 5.2.6.2 for instructions in isolating correctable ECC error problems.

Note

The memory fault interpretation procedures work only if the memory modules have been properly installed and configured. For example, memory modules should start in backplane slot 4 (next to the processor module in slot 5) and proceed to slot 1 with no gaps.

Note

Although the OpenVMS error handler has built-in features to aid Services in memory repair, good judgment is needed by the Service Engineer. It is essential to understand that in many, if not most cases, correctable ECC errors are transient in nature. No amount of repair will fix them, as generally there is nothing to be fixed.

Memory modules can represent a great expense to the Corporation when they are sent back to Repair with no errors. If one disagrees with the strategy in this section or has questions or suggestions, please contact Corporate Support.

5.2.6.1 Uncorrectable ECC Errors

Refer to Example 5–3, which provides an abbreviated error log for uncorrectable ECC errors.

For uncorrectable ECC errors, a memory subpacket will be logged as indicated by "memory subpacket" listed in the third column of the FLAGS software register (❶). Also, the hardware register MESR <11> (❷) of the processor Register Subpacket will be set equal to 1, and MEAR will latch the error address (❺).

Examine the MEMCON software register (❸) under the memory subpacket. The MEMCON register provides memory configuration information.

The OpenVMS error handler will mark each page bad and attempt page replacement, indicated in SYSTAT (❶). The DCL command SHOW MEMORY (Example 5–4) will also indicate the result of OpenVMS page replacement.

Uncorrectable memory errors will increment the OpenVMS global counter, which can be viewed using the DCL command SHOW ERROR.

Note

If register MESR <11> was set equal to 1, but MESR <19:12> syndrome equals 07, no memory subpacket will be logged as a result of incorrect check bits written to memory because of an NDAL bus parity error detected by the NMC. In short, this indicates a problem with the CPU module, not memory. There should be a previous entry with MESR <22>, NDAL Data Parity Error set equal to 1.

Note

One type of uncorrectable ECC error, that due to a "disown write", will result in a CRD entry like those for correctable ECC errors. The FOOTPRINT longword for this entry contains the message "Uncorrectable ECC errors due to disown write". The failing module should be replaced for this error.

Example 5–3 Error Log Entry Indicating Uncorrectable ECC Error

```
V A X / V M S        SYSTEM ERROR REPORT        COMPILED  6-NOV-1991 10:16:49
                                                                  PAGE  25.
******************************** ENTRY      13. ********************************
ERROR SEQUENCE 2.                                LOGGED ON:        SID 13001401
DATE/TIME  4-OCT-1991 09:14:29.86                                  SYS_TYPE 03110A01
SYSTEM UPTIME: 0 DAYS 00:01:39
SCS NODE: OMEGA1                                                   VAX/OpenVMS V5.5-2

INT54 ERROR  KA50  CPU Microcode Rev # 1.  CONSOLE FW REV# 1.1
                   Standard Microcode Patch    Patch Rev # 10.

      REVISION        00000000
      SYSTAT          00000601
                                    ATTEMPTING RECOVERY
                                    PAGE MARKED BAD
                                    PAGE REPLACED      ❹
      FLAGS           00000006
                                    memory subpacket   ❶
                                    KA50 subpacket

KA50 REGISTER SUBPACKET

      BPCR            ECC80000
      .
      .
      .
      MESR            80006800
                                    UNCORRECTABLE MEMORY ECC ERROR   ❺
                                    ERROR SUMMARY
                                    MEMORY ERROR SYNDROME  = 06(X)
      .
      .
      .
      MEAR            02FFDC00
                                    main memory error address  = 0BFF7000   ❻
                                    ndal commander id  = 00(X)
      .
      .
      .
      IPCR0           00000020
                                    LOCAL MEMORY EXTERNAL ACCESS ENABLED

 MEMORY SUBPACKET

      MEMCON          000FFFF02  ❸
                                    MEMORY CONFIGURATION:
                                    MS44-AA SIM Memory Module 4 MB location 1E
                                    MS44-AA SIM Memory Module 4 MB location 1F
                                    MS44-AA SIM Memory Module 4 MB location 1G
                                    MS44-AA SIM Memory Module 4 MB location 1H
                                     total memory = 16MB
```

(continued on next page)

Example 5–3 (Cont.) Error Log Entry Indicating Uncorrectable ECC Error

```
MEMCON3          8B000003
                                  64 bit mode
                                  Base address valid
                                  RAM size = 1MB
                                  base address  = 0B(X)
```

Example 5–4 SHOW MEMORY Display Under the OpenVMS Operating System

```
$ SHOW MEMORY
            System Memory Resources on 21-FEB-1992 05:58:52.58

Physical Memory Usage (pages):     Total       Free      In Use    Modified
  Main Memory (128.00Mb)          262144     224527       28759        8858

  Bad Pages                        Total    Dynamic  I/O Errors      Static
                                       1          1           0           0

Slot Usage (slots):                Total       Free    Resident     Swapped
  Process Entry Slots                360        347          13           0
  Balance Set Slots                  324        313          11           0

Fixed-Size Pool Areas (packets):   Total       Free      In Use        Size
  Small Packet (SRP) List           3067       2724         343         128
  I/O Request Packet (IRP) List     2263       2070         193         176
  Large Packet (LRP) List             87         61          26        1856

Dynamic Memory Usage (bytes):      Total       Free      In Use     Largest
  Nonpaged Dynamic Memory        1037824     503920      533904      473184
  Paged Dynamic Memory           1468416     561584      906832      560624

Paging File Usage (pages):                     Free  Reservable       Total
  DISK$VMS054-0:[SYS0.SYSEXE]PAGEFILE.SYS    300000      266070      300000

Of the physical pages in use, 24120 pages are permanently allocated to OpenVMS.
$
```

Using the OpenVMS command ANALYZE/SYSTEM, you can associate a page that had been replaced (Bad Pages in SHOW MEMORY display) with the physical address in memory.

In Example 5–5, 5ffb8 (under the Page Frame Number (PFN) column) is identified as the single page that has been replaced. The command EVAL 5ffb8 * 200 converts the PFN to a physical page address. The result is 0bff7000, which is the MEAR address translated in Example 5–3. (Bits <8:0> of the addresses may differ since the page address from EVAL always shows bits <8:0> as 0.)

Example 5–5 Using ANALYZE/SYSTEM to Check the Physical Address in Memory for a Replaced Page

```
$ ANALYZE/SYSTEM

VAX/OpenVMS System analyzer

SDA> SHOW PFN /BAD

Bad page list
-------------
Count:                  1
Lolimit:               -1
High limit:     1073741824

  PFN     PTE ADDRESS    BAK     REFCNT   FLINK    BLINK      TYPE       STATE
  ----    -----------  --------  ------   -----    -----    ----------  ----------
0005FFB8    00000000   00000000      0   00000000 00000000  20 PROCESS  02 BADLIST

SDA> EVAL 5ffb8 * 200
Hex = 0BFF7000   Decimal = 201289728
SDA> EXIT
$
```

5.2.6.2 Correctable ECC Errors

Refer to Example 5–6, which provides an error log showing correctable ECC errors.

For correctable ECC errors, a Single-Bit Error (SBE) Memory Subpacket will be logged as indicated by "memory sbe reduction subpacket" listed in the third column of the FLAGS software register (❶).

The Memory SBE Reduction Subpacket header contains a CURRENT ENTRY register (❷) that displays the number of the Memory CRD Entry that caused the error notification. If CURRENT ENTRY > 0, examine which bits are set in the STATUS register (❸) for this entry—GENERATE REPORT should be set.

Note

If CURRENT ENTRY = 0, then the entry was logged for something other than a single-bit memory correctable error Footprint. You will need to examine all of the Memory CRD Entries and Footprints to try to determine the likely FRU.

Check for the following:

- SCRUBBED (❹)—If SCRUBBED is the only bit set in the STATUS register, memory modules should NOT generally be replaced.

The kernel performs memory scrubbing of DRAM memory cells that may flip due to transient alpha particles. Scrubbing simply reads the corrected data and writes it back to the memory location. Returning memory modules that only have SCRUBBED set in STATUS will cost the corporation money, since the repair centers will generally not find a problem.

Unlike uncorrectable ECC errors, the error handling code cannot indicate if the page has been replaced. To get some idea, use DCL command, SHOW MEMORY. If the page mapout threshold has not been reached ("PAGE MAPOUT THRESHOLD EXCEEDED" is not set in SYSTAT packet header register (❺)), the system should be restarted at a convenient time to allow the power-up self-test and ROM-based diagnostics to map out these pages. This can be done by entering TEST 0 at the console prompt, running an extended script TEST A9, or by powering down then powering up the system. In all cases, the diagnostic code will mark the page bad for hard single address errors, as well as any uncorrectable ECC error by default.

If there are many locations affected by hard single-cell errors, on the order of one or more pages per MB of system memory, the memory module should be replaced. The console command SHOW MEMORY will indicate the number of bad pages per module. For example, if the system contains 64 MB of main memory and there are 64 or more bad pages, the affected memory should be replaced.

Note

Under the OpenVMS operating system, the page mapout threshold is calculated automatically. If "PAGE MAPOUT THRESHOLD EXCEEDED" is set in SYSTAT (❺), the failing memory module should be replaced.

In cases of a new memory module used for repair or as part of system installation, one may elect to replace the module rather than having diagnostics map them out, even if the threshold has not been reached for hard single-address errors.

- MULTIPLE ADDRESSES (❻)—If the second occurrence of an error within a footprint is at a different address (LOWEST ADDRESS not equal to HIGHEST ADDRESS (❼), MULTIPLE ADDRESSES will be set in STATUS along with SCRUBBED. Scrubbing will not be attempted for this situation. In most cases, the failing memory module should be replaced regardless of the page mapout threshold.

If CRD BUFFER FULL is set in LOGGING REASON (❽) (located in the subpacket header) or PAGE MAPOUT THRESHOLD EXCEEDED is set in SYSTAT (❼), the failing memory module should be replaced regardless of any thresholds.

For all cases (except when SCRUBBED is the only flag set in STATUS) isolate the offending memory by examining the translation in FOOTPRINT called MEMORY ERROR STATUS (❾): The memory module is identified by its backplane position. In Example 5–6, SIMM memory modules in locations 0A and 0B are identified as failing.

The Memory SBE Reduction Subpacket header translates the MEMCON register (❿) for memory subsystem configuration information.

Unlike uncorrectable memory and CPU errors, the OpenVMS global counter, as shown by the DCL command SHOW ERROR, is not incremented for correctable ECC errors unless it results in an error log entry for reasons other than system shutdown.

Note

If footprints are being generated for more than one memory module, especially if they all have the same bit in error, the processor module, backplane, or other component may be the cause.

Note

One type of uncorrectable ECC error, that due to a "disown write", will result in a CRD entry like those for correctable ECC errors. The FOOTPRINT longword for this entry contains the message "Uncorrectable ECC errors due to disown write". The failing module should be replaced for this error.

Example 5–6 Error Log Entry Indicating Correctable ECC Error

```
V A X / V M S        SYSTEM ERROR REPORT        COMPILED 21-NOV-1991 16:55:58
                                                                  PAGE   1.
******************************* ENTRY      1. ******************************
ERROR SEQUENCE 2.                              LOGGED ON:        SID 13001401
DATE/TIME 27-SEP-1991 09:51:13.98                          SYS_TYPE 03110A01
SYSTEM UPTIME: 0 DAYS 00:05:06
SCS NODE: OMEGA1                                          VAX/OpenVMS V5.5-2

CORRECTABLE MEMORY ERROR  KA50  CPU Microcode Rev # 1.  CONSOLE FW REV# 1.1
                    Standard Microcode Patch    Patch Rev # 10.

      REVISION        00000000
      SYSTAT          00000040 ❺
      FLAGS           00000008

                                  ❶
                                      memory sbe reduction subpacket

 MEMORY SBE REDUCTION SUBPACKET

      LOGGING REASON  00000004 ❻
                                      shutdown
      PAGE MAPOUT CNT 00000000
      MEMCON          000FFD01 ❼
                                      MEMORY CONFIGURATION:
                                      MS44-AA SIM Memory Module (4MB) Loc 0A
                                      MS44-AA SIM Memory Module (4MB) Loc 0B
                                      MS44-AA SIM Memory Module (4MB) Loc 0C
                                      MS44-AA SIM Memory Module (4MB) Loc 0D
                                       _Total memory = 16MB
                                       _sets enabled = 000000001
                                ❾  MEMORY ERROR STATUS:
                                       SIMM MEMORY MODULES: LOCATIONS 0A & 0B
                                       Set  = 0(X)
                                       Bank = A
      VALID ENTRY CNT 00000001
                                      1.
      CURRENT ENTRY   00000000
                                      0. ❷

 MEMORY CRD ENTRY 1.

      FOOTPRINT       00000073
```

(continued on next page)

Example 5–6 (Cont.) Error Log Entry Indicating Correctable ECC Error

```
                                MEMORY ERROR STATUS:
                                _SIMM MEMORY MODULE:  LOCATION 0A
                                _set = 0
                                _bank = 0.
                                ECC SYNDROME = 73(X)
                                _CORRECTED DATA BIT = 0.
                                _
STATUS ❹  00000010
                                scrubbed ❶
CRD CNT          00000001
                                1.
PAGE MAPOUT CNT  00000000
                                0.
FIRST EVENT      16B0F640
                 009622CB
                                16-OCT-1992 11:03:36.10
LAST EVENT       16B0F640
                 009622CB
                                16-OCT-1992 11:03:36.10
LOWEST ADDRESS   0BFF4000
HIGHEST ADDRESS  0BFF4000  ❻ ❼
```

Note

Ownership (O-bit) memory correctable or fatal errors (MESR <04> or MESR <03> of the processor Register Subpacket set equal to 1) are processor module errors, NOT memory errors.

5.2.7 Interpreting System Bus Faults Using ANALYZE/ERROR

If hardware register CESR <09> (❶) and/or CQBIC hardware register DSER <07>, <05>, or <02> (❷) is set equal to 1, there may be a problem with the Q–bus or Q–bus option.

When CESR <09> is set equal to 1, examine the hardware register CIOEAR2 (❸) to determine the address of the offending option.

Example 5–7 provides an error log showing a faulty Q–bus option. The CIOEAR2 error register indicates the first UQSSP controller as the offending address.

Example 5–7 Error Log Entry Indicating Q-Bus Error

```
V A X / V M S        SYSTEM ERROR REPORT         COMPILED 20-NOV-1991 14:28:13
                                                                  PAGE   1.
******************************** ENTRY      75. ********************************
ERROR SEQUENCE 1852.                              LOGGED ON:        SID 13001401
DATE/TIME 20-NOV-1991 14:26:11.14                                 SYS_TYPE 00310A01
SYSTEM UPTIME: 12 DAYS 20:04:19
SCS NODE:                                                         VAX/OpenVMS V5.5-2

MACHINE CHECK  KA50  CPU Microcode Rev # 1.  CONSOLE FW REV# 1.1
                     Standard Microcode Patch     Patch Rev # 10.

      REVISION        00000000
      SYSTAT          00000001
                                    ATTEMPTING RECOVERY
      FLAGS           00000003
                                    machine check stack frame
                                    KA50 subpacket

 STACK FRAME SUBPACKET

      ISTATE_1        80060000
      .
      .
      .
      PSL             03C00000
                                    PSL previous mode = user
                                    PSL current mode = user
                                    first part done set

KA50  REGISTER SUBPACKET
```

(continued on next page)

Example 5–7 (Cont.) Error Log Entry Indicating Q-Bus Error

```
      BPCR            ECC80024
      .
      .
      .
      CESR            80000200 ❶
                                    CP2 IO ERROR
                                    ERROR SUMMARY
      .
      .
      .
      DSER            00000080 ❷
                                    Q-22 BUS NXM
      .
      .
      .
      CIOEAR2         00001468                    ❸
                                    cp2 IO error address  = 20001468
                                    NDAL commander id (cp2 transac)  = 0(X)
      .
      .
      .
      IPCR0           00000020
                                    LOCAL MEMORY EXTERNAL ACCESS ENABLED
ANAL/ERR/OUT=QBUS QBUS.ZPD
```

5.2.8 Interpreting DMA ⇔ Host Transaction Faults Using ANALYZE/ERROR

Some kernel errors may result in two or more entries being logged. If the SGEC Ethernet controller or other CDAL device (residing on the processor module) encounter host main memory uncorrectable ECC errors, main memory NXMs or CDAL parity errors or timeouts, more than one entry results. Usually there will be one Polled Error entry logged by the host, and one or more Device Attention and other assorted entries logged by the device drivers.

In these cases the processor module or one of the four memory modules are the most likely cause of the errors. Therefore, it is essential to analyze Polled Error entries, since a polled entry usually represents the source of the error versus other entries, which are simply aftereffects of the original error.

Example 5–8 provides an abbreviated error log for a polled error. Example 5–9 provides an example of a device attention entry.

Example 5–8 Error Log Entry Indicating Polled Error

```
V A X / V M S        SYSTEM ERROR REPORT         COMPILED 17-FEB-1992 05:32:21
                                                          PAGE   1.
******************************** ENTRY      2. ********************************
ERROR SEQUENCE 15.                              LOGGED ON:        SID 13001401
DATE/TIME 17-FEB-1992 05:22:00.90                                 SYS_TYPE 00310A01
SYSTEM UPTIME: 0 DAYS 00:27:48
SCS NODE:                                                         VAX/OpenVMS V5.5-2

POLLED ERROR  KA50  CPU Microcode Rev # 1.  CONSOLE FW REV# 1.1
                    Standard Microcode Patch    Patch Rev # 10.

      REVISION        00000000
      SYSTAT          00000001
                                    ATTEMPTING RECOVERY
      FLAGS           00000006
                                    memory subpacket
                    KA50 subpacket

KA50 REGISTER SUBPACKET

      BPCR            ECC80024
      .
      .
      .
      MESR            8001B800
                                    UNCORRECTABLE MEMORY ECC ERROR
                                    ERROR SUMMARY
                                    MEMORY ERROR SYNDROME  = 1B(X)
      .
      .
      .
      MEAR            50000410
                                    main memory error address  = 00001040
                                    ndal commander id  = 05(X)
      .
      .
      .
      IPCR0           00000020
                                    LOCAL MEMORY EXTERNAL ACCESS ENABLED

MEMORY SUBPACKET

      MEMCON          000FFFF02
                                    MEMORY CONFIGURATION:
                                    MS44-AA SIM Memory Module 4 MB location 1E
                                    MS44-AA SIM Memory Module 4 MB location 1F
                                    MS44-AA SIM Memory Module 4 MB location 1G
                                    MS44-AA SIM Memory Module 4 MB location 1H
                                     total memory = 16MB
```

(continued on next page)

Example 5–8 (Cont.) Error Log Entry Indicating Polled Error

```
    MEMCONO          80000003
                                    64 bit mode
                                    Base address valid
                                    RAM size = 1MB
                                    base address  = 00(X)

ANAL/ERR/OUT=TB1 TB1.ZPD
```

Example 5–9 Device Attention Entry

```
V A X / V M S        SYSTEM ERROR REPORT        COMPILED 17-FEB-1992 05:32:21
                                                          PAGE   1.
********************************* ENTRY      2. *********************************
ERROR SEQUENCE 15.                              LOGGED ON:        SID 13001401
DATE/TIME 17-FEB-1992 05:22:00.90                                 SYS_TYPE 00310A01
SYSTEM UPTIME: 0 DAYS 00:27:48
SCS NODE:                                                        VAX/OpenVMS V5.5-2

DEVICE ATTENTION  KA50  CPU Microcode Rev # 1.  CONSOLE FW REV# 1.1
                  Standard Microcode Patch    Patch Rev # 10.

DSSI SUB-SYSTEM, PAB0: - PORT WILL BE RE-STARTED

      PORT TIMEOUT, DRIVER RESETTING PORT

      CNF             03060022
                                    MAINTENANCE ID = 0022(X)
                                    FIRMWARE REVISION = 06(X)
                                    HARDWARE REVISION = 03(X)
      PMCSR           00000000
      PSR             80010000
                                    MAINTENANCE ERROR
                                    SHARED HOST MEMORY ERROR
      PFAR            40001044
                                    APPROX HOST ADDR 40001044(X)
      PESR            00010000
                                    CPDAL BUS ERROR
      PPR             00000000
                                    NODE #0.
                                    0. BYTE INTERNAL BUFFER
                                    16. NODES MAXIMUM

      UCB$B_ERTCNT          2C
                                    44. RETRIES REMAINING
      UCB$B_ERTMAX          32
                                    50. RETRIES ALLOWABLE
      UCB$L_CHAR      0C450000
                                    SHARABLE
                                    AVAILABLE
                                    ERROR LOGGING
                                    CAPABLE OF INPUT
                                    CAPABLE OF OUTPUT
      UCB$W_STS           0010
                                    ONLINE
      UCB$W_ERRCNT        0007
                                    7. ERRORS THIS UNIT
ANAL/ERR/ENTRY=(ST:2,END:3)/OUT=POLL_SHM
```

5.2.9 VAXsimPLUS and System-Initiated Call Logging (SICL) Support

Symptom-Directed Diagnostic (SDD) toolkit support for KA50/51/55/56 kernels is provided in version 2.0 of the toolkit. If version 2.0 is not available, you should install the previous version, as it provides support for many existing options.

MicroVAX 3100 systems use Symptom-Directed Diagnosis tools primarily for notification. The VAX System Integrity Monitor Plus (VAXsimPLUS) interactive reporting tool triggers notification for high-level events recorded in SYSTAT and LOGGING REASON.

The VAXsimPLUS monitor simply parses for a handful of SYSTAT flags and LOGGING reason codes. The VAXsimPLUS monitor display is updated and triggering occurs if the threshold has been reached. Some flags have a threshold of one; for example, SYSTAT <08> ERROR THRESHOLD EXCEEDED will trigger VAXsimPLUS upon the first occurrence, since at least three errors would have already occurred and been handled by the OpenVMS operating system.

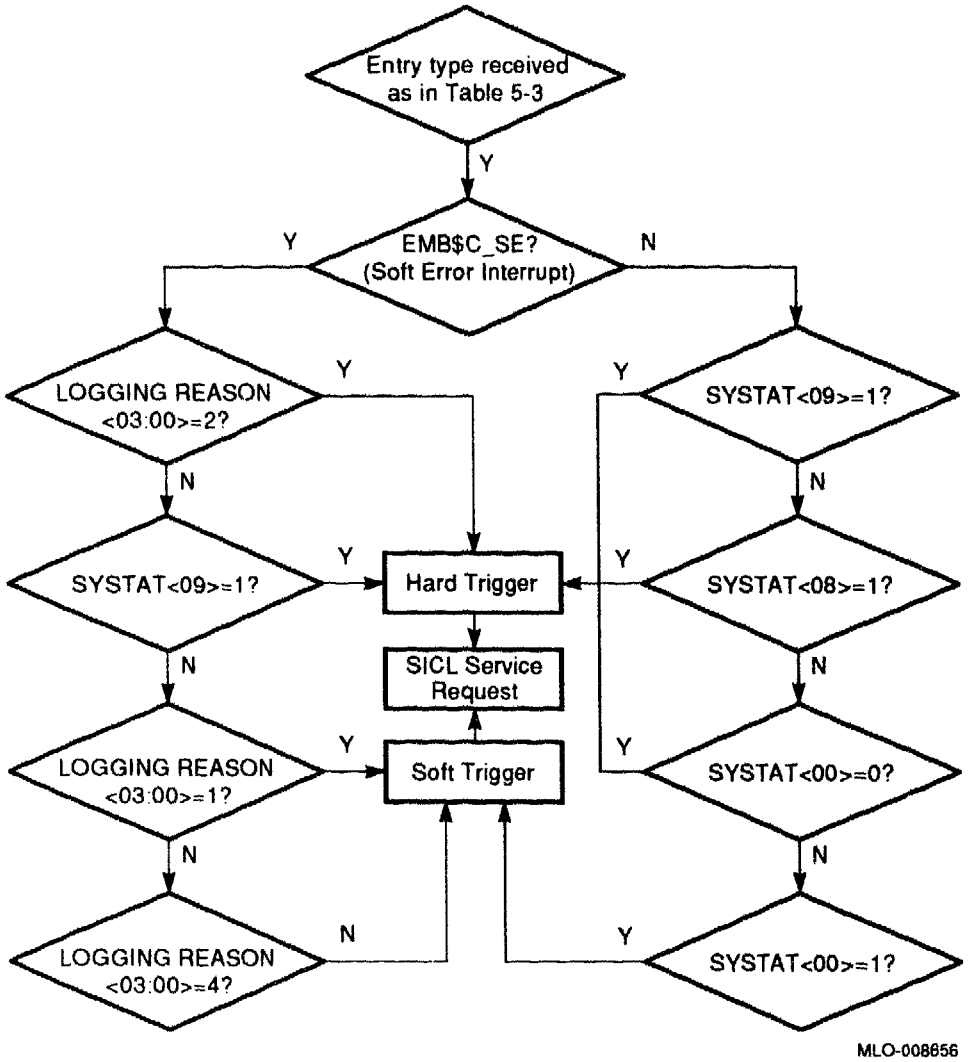
All lower level errors will ultimately set one of the conditions shown in Table 5–2. VAXsimPLUS will examine the conditions within a 24-hour period—thresholds are typically one or two flags or logging reason codes within that period.

Table 5–2 lists the conditions that will trigger VAXsimPLUS notification and updating. Figure 5–8 shows the flow for the VAXsimPLUS monitor trigger (for decision blocks with only one branch, the alternative is treated as an ignore condition). The entries ultimately are classified as either hard or soft. Errors that require corrective maintenance are classified as hard; while errors potentially requiring corrective maintenance are classified as soft.

Table 5–2 Conditions That Trigger VAXsimPLUS Notification and Updating

Condition	Description
SYSTAT <00> = 1	"Attempting recovery"
SYSTAT <00> = 0	"Full recovery or retry not possible"
SYSTAT <08> = 1	"Error threshold exceeded"
SYSTAT <09> = 1	"Page marked bad for uncorrectable ECC error in main memory"
SYSTAT <11> = 1	"Page mapout threshold for single bit ECC errors in main memory exceeded"
LOGGING REASON <3:0> = 1	"Memory CRD buffer full"
LOGGING REASON <3:0> = 2	"Generate report as a result of hard single address or multiple address DRAM memory fault"
LOGGING REASON <3:0> = 0, 3, 5–F	"Illegal LOGGING REASON"

Figure 5–8 Trigger Flow for the VAXsimPLUS Monitor



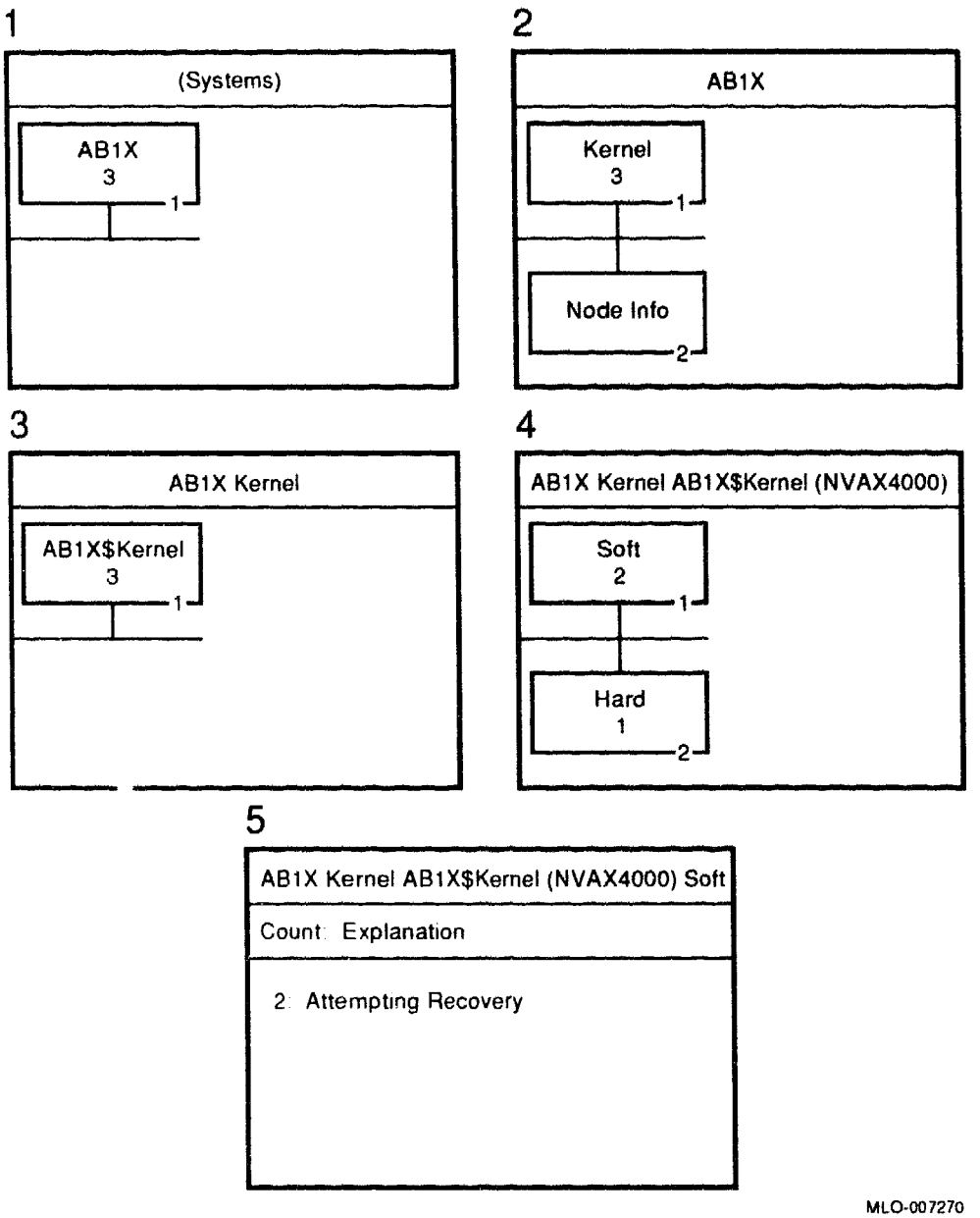
VAXsimPLUS triggering notifies the customer and Services using three message types: HARD, SOFT, and SICL Service Request. Each message contains the single STARS article theory number, as well as the SYSTAT or LOGGING REASON state. In addition, the SICL Service Request will have a Merged Error Log (MEL) datafile appended. Both hard and soft triggers will generate SICL Service Request messages.

Figure 5–9 shows the five VAXsimPLUS monitor screen displays. Table 5–3 provides a brief explanation of the five levels of screen displays.

Table 5–3 Five-Level VAXsimPLUS Monitor Screen Displays

Level	Explanation
1. System	The system level screen provides one box for each system being analyzed (in Figure 5–9 a single system is being analyzed). As with each screen level, the number of reported errors is displayed in the box. The boxes blink when the hard error thresholds are reached; the boxes are highlighted when the soft error thresholds are reached.
2. Subsystem	The subsystem level screen provides separate boxes for the kernel and node information. Other boxes that may be displayed are bus, disk, tape, etc.
3. Unit	The unit level screen provides a box for the kernel. If the subsystem has more than one unit or device with errors, those will be displayed as well.
4. Error Class	The error class level screen provides a box for both hard and soft errors.
5. Error Detail	Two error detail level screens (hard and soft) provide the number of reported errors along with a brief error description.

Figure 5–9 Five-Level VAXsimPLUS Monitor Display



Once notification occurs, the service engineer should examine the error log file (after using the ANALYZE/ERROR command) or read the appended Merged Error Log (MEL) file in the SICL service request message. (The MEL file is encrypted, refer to Section 5.2.9.1 for instructions in converting these files.)

Using the theory of interpretation provided in the previous sections, you can manually interpret the error logs.

Note

The interpretation theory provided in this manual is also a STARS article and can be accessed via the Decoder Kit. (Theory 30B01.xxx reproduces in full, Section 5.2 of this manual).

In summary, a service engineer should use VAXsimPLUS notification as follows:

1. Make sure all four message types are sent to the Field and System accounts.
2. Log into the Field or System account.
3. Read mail (look for the SICL service request message with its appended MEL file).
4. Convert the encrypted MEL file and use the theory provided in this manual to interpret the error log file.

5.2.9.1 Converting the SICL Service Request MEL File

Use the following procedure to convert the encrypted MEL file that is appended to the SICL service request message (MEL files can be converted on site or at a support center). Example 5–10 shows a sample SICL service request message and appended MEL file.

1. Extract the SICL mail message from mail.
2. Edit the extracted file to obtain the appended MEL file. The MEL file is the encrypted code that appears between the rows of asterisks and includes the words "SICL" and "end."
3. Convert the encrypted code to a binary file using the VAXsimPLUS decode command file as follows:

   $ MCR SDD$EXE:FMGR$SICL_DECODE [MEL filename] [binary filename]

4. Use the ANALYZE/ERROR command to produce an error log entry.

$ ANALYZE/ERROR [binary filename]

Example 5–10 SICL Service Request with Appended MEL File

```
From: AB1X::SDD$MANAGER      "VAXsimPLUS Message" 15-APR-1992 10:29:21.05
To: SYSTEM
CC:
Subj: SDD T2.0 Service Request - Analysis:[30B01.200]
********************************************************************************

                  VAXsimPLUS Notification Message

   VAXsimPLUS has detected that the following device needs attention:

       DEVICE:                 AB1X$KERNEL (NVAX4000)
       NODE:                   AB1X
       SYSTEM SERIAL NUMBER:   KA136H1520
       SYSTEM TYPE:            VAX 4000-600

                  VAXsimPLUS Diagnosis Information

Attn:          Field Service

Device:        AB1X$KERNEL (NVAX4000)
Count:         1.

Theory:        [30B01.200]

Evidence:  Urgent action required - AB1X$KERNEL Hard error(s):

          SYSTAT <9> = 1 - Page Marked Bad For Uncorrectable ECC Error In
                           Main Memory
********************************************************************************
%% SDD$PROFILE is defined to be NONE, no Customer Profile included in message %%
********************************************************************************
SICL
 134
M @( $_O  0=# 0        A$24U)3$\@(            %  @ G!::G+Y*5
M  @ 034N-2U-,2 7    &0\          @  !P   !@       :  0   "<<
M  !\F>] "    (  %/,%SP    #  RO R.P     \%31!03  !P  @ !
M (H   #0 0"  /S_S$X_\A  !            F 6  /CA"0     (P0
M"A(        %\   [P          0    '@U@. '        @%.^!0
M    (S+<]P ,12  P P                      , "          S ,13
FO@4          \         (     ?Y#P(%  " !_D/ @4  (
end
********************************************************************************
```

5.2.9.2 VAXsimPLUS Installation Tips

When installing VAXsimPLUS, the system will prompt you for information. You will need to know the serial number and system model number for the system on which you are installing VAXsimPLUS. The serial number is located on the front of the chassis at the bottom and to the left (the front door must be open). The system model number is attached to the outside of the door.

Also, if the system does not have dialout capability, you should answer no when asked if you want to enable SICL—if you enter yes, the system will attempt to send mail via DSNLink resulting in error messages. After VAXsimPLUS is installed you can activate SICL and customize the VAXsimPLUS mailing lists so that SICL messages are sent to an appropriate destination(s) on site. This way, SICL messages are received onsite without incurring error messages regarding remote link failures.

5.2.9.3 VAXsimPLUS Post-Installation Tips

Once VAXsimPLUS is installed, you can set up mailing lists to direct VAXsimPLUS messages to the appropriate destinations. If the system has no dialout capability, SICL messages should be directed to the System and/or Field account—this is good practice for systems with dialout and service center support as well.

In the example that follows, the four types of mailing lists are displayed and System and Field accounts are added to all four mailing lists using VAXSIM /FAULT_MANAGER commands.

Note

The commands can be abbreviated.

DSN%SICL appears under the SICL mailing list if you enabled SICL during installation.

```
$ VAXSIM/FAULT SHOW MAIL
-- FSE mailing list --

    FIELD

-- CUSTOMER mailing list --

    SYSTEM

-- MONITOR mailing list is empty --

-- SICL mailing list --

    DSN%SICL

$ VAXSIM/FAULT ADD SYSTEM ALL
$ VAXSIM/FAULT ADD FIELD ALL
$ VAXSIM/FAULT SHOW MAIL
-- FSE mailing list --

    FIELD
    SYSTEM

-- CUSTOMER mailing list --

    FIELD
    SYSTEM

-- MONITOR mailing list --

    FIELD
    SYSTEM

-- SICL mailing list --

    DSN%SICL
    FIELD
    SYSTEM
```

To activate SICL after installation, use the following command:

$ VAXSIM/FAULT SET SICL ON

VAXsimPLUS customer notification messages should display a phone number for the customer to call in the event the system needs service. Use the following commands to examine and set the phone number parameter:

```
$ VAXSIM/FAULT SHOW PARAMETER

(SET parameter)          (Parameter settings)

PHONE_NUMBER    Customer Service Phone Number is unknown
COPY            Automatic copying is OFF
SICL            System Initiated Call Logging is ON

SYSTEM_INFO  System info for AB1X
             Serial number   KA136H1520
             System type     VAX 4000-600
```

```
$ VAXSIM/FAULT SET PHONE 1-800-DIGITAL
```

Finally, the VAXSIMPLUS/MERGE command is useful in examining how a device is functioning in a cluster. The merge command collects the messages that are being sent to the other CPUs in the cluster.

5.2.10 Repair Data for Returning FRUs

When sending back an FRU for repair, include as much of the error log information as possible. If one or more error flags are set in a particular entry, record the mnemonic(s) of the register(s), the hex data, and error flag translation(s) on the repair tag. If an error address is valid, include the mnemonic, hex data, and translation on the repair tag as well. For memory and cache errors, include the syndrome and corrected-bit/bit-in-error information, along with the register mnemonic and hex data. Other registers which should be recorded for any entry type are SYSTAT, MEMCON and FOOTPRINT.

5.3 Power-On Self-Test (POST) and ROM-Based Diagnostic (RBD) Failures

If any of the tests fail, the test code displays on the console LED and, if specified in the firmware script, a diagnostic console printout displays in the format shown in Example 5–11.

Example 5–11 Sample Output with Errors

```
                ❶ ❷          ❸          ❹          ❺
? Test_Subtest_40_06  Loop_Subtest=00  Err_Type=FF   DE_Memory_count_pages.lis
  ❻          ❼
Vec=0000  Prev Errs=0004   P1=00000001   P2=00000002   P3=00000001  P4=00000000
P5=00000020  P6=00008000   P7=00000020   P8=00000000   P9=00000000 P10=00FCD44B
r0=00FF4008  r1=00000007   r2=00000000   r3=FFFFFFFF   r4=00000068  r5=00000000
r6=00000000  r7=00000002   r8=00FF4000   r9=20140758  r10=FFFFFFFE r11=FFFFFFFF
 dser=0000 cesr=00000000 intmsk=00 icsr=01 pcsts=FC00 pcadr=FFFFFFF8 pcctl=FC13
 cctl=00000021 bcetsts=0000 bcedsts=0000 cefsts=00000200 nests=00
 mmcdsr=01111000 mesr=00080000

>>>
```

Several lines are printed in the error display. The first line has eight column headings:

❶ *Test* identifies the diagnostic test, test ?40 in Example 5–11. Using Table 5–4, you can use the test number to point to possible problems in field replaceable units (FRUs).

❷ *Subtest log* is two hex digits identifying, usually within 10 instructions, where in the diagnostic the error occurred.

❸ *Loop_subtest_log* is an additional log generated out of the current test specified by the current test number and subtestlog. Usually these logs occur in common subroutines called from a diagnostic test.

❶ *Error_type* (diagnostic executive error) signals the diagnostic's state and any illegal behavior. This field indicates a condition that the diagnostic expects on detecting a failure. FE or EF in this field means that an unexpected exception or interrupt was detected. FF indicates an error as a result of normal testing, such as a miscompare. The possible codes are:

Error Code	Description
FF	Normal error exit from diagnostic
FE	Unanticipated interrupt
FD	Interrupt in cleanup routine
FC	Interrupt in interrupt handler
FB	Script requirements not met
FA	No such diagnostic
EF	Unanticipated exception in executive

❺ *ASCII messages* Shows the name of the listing file that contains the failed diagnostic.

❻ *Vec* identifies the SCB vector through which the unexpected exception or interrupt trapped, when the de_error field detects an unexpected exception or interrupt (FE or EF).

❼ *Prev_errs* is four hex digits showing the number of previous errors that have occurred (four in Example 5–11).

Lines 2 and 3 of the error printout are parameters 1 through 10. When the diagnostics are running normally, these parameters are the same parameters listed in Example 4–3.

When returning a module for repair, always record the the test number, subtest, and Err_type from line 1 of the printout. Also record the Vec from line 2. If possible, record additional information. If the error can be saved onto a printer, then enclose the full printout with the failing module.

Note

Do not confuse the countdown pattern of powerup tests with the test number. In the following the last countdown was 58; this number should not be reported! The test number was 31.

The countdown pattern is used to indicate progress in the power-up tests. The actual true test number associated with a countdown value can change from one release of the ROM code to another. For example:

```
KA50-A T1.2-156, VMB 2.14
Performing normal system tests.
72..71..70..69..68..67..66..65..64..63..62..61..60..59..58..

? Test_Subtest_31_06  Loop_Subtest=05  Err_Type=FF   DE_Memory_Setup_CSRs.lis
Vec=0000  Prev_Errs=0000   P1=C94AC94A   P2=01000000   P3=00000002  P4=00000000

Minimum recording for this error is:

 Test = 31
 Subtest = 6
 Loop_subtest = 5
 Err_type = FF
 Vec = 0.
```

Table 5-4 lists the hex LED display, the default action on errors, and the most likely unit that needs replacing reading from left to right. Example, 1,4 indicates 1 is most likely, then 4. The Default on Error column refers to the action taken by the diagnostic executive when the test fails in the script.

Memory tests are usually treated differently; when an error occurs, the memory tests usually try to continue and mark the bitmap. Test 40 reports failing pages in the bitmap.

When any memory test fails, always do a SHOW MEMORY to help identify the FRU. SHOW MEMORY will identify the FRU to a SET of SIMMs or to an individual SIMM if possible.

If a single set of SIMMs is present, and replacing a suspected bad SIMM or set does not fix the problem, assume that the system board is bad. Always check the seating of SIMMs before replacing. If nonvolatile data is lost after powerup or you always get a request to select a language at powerup, the battery may be bad.

Table 5–4 shows the various LED values and console terminal displays as they point to problems in field-replaceable units (FRUs).

Table 5–4 KA50/51/55/56 Console Displays as Pointers to FRUs

On Error Hex LED	Normal Console Display	Default Action on Error	Failing Test Number	Test Description	FRU[1]
Power-Up Tests (Script A1)					
F	None	Loop	None	Power up	1, 4
E	None	None	None	ROM code execution begun	1, 4
D	None	Loop	None	Wait for power	1, 4
B	72	Cont	9D	Utility	1, 4
B	71	Cont	42	Chk_for_interrupts	1, 3
9	70	Cont	35	B_Cache_diag_mode	1
B	69	Cont	33	NMC_powerup	1
B	68	Cont	32	NMC_registers	1
B	67	Cont	D0	V_Cache_diag_mode	1
B	66	Cont	D2	O_bit_Diag_mode	1
B	65	Cont	DF	O-bit_debug	1
B	64	Cont	46	P_cache_diag_mode	1
9	63	Cont	35	B_cache_diag_mode	1
9	62	Cont	DE	B_Cache_tag_debug	1
9	61	Cont	DD	B_Cache_data_debug	1
9	60	Cont	DA	PB_Flush_cache	1
8	59	Halt	DC	NO_Memory_present	2, 1
8	58	Cont	31	Memory_Setup_CSRs	2, 1
8	57	Halt	30	Memory_Init_Bitmap	2, 1
7	56	Cont	91	CQBIC_powerup	1, 3

[1]Field-replaceable unit key:

1 = KA50
2 = MS44
3 = Q22–bus option
4 = System power supply
5 = SCSI device or 2 devices with same target id
6 = ASYNC option board
7 = COMM option board (SYNC)
8 = SHAC option board

(continued on next page)

Table 5–4 (Cont.) KA50/51/55/56 Console Displays as Pointers to FRUs

On Error Hex LED	Normal Console Display	Default Action on Error	Failing Test Number	Test Description	FRU[1]
Power-Up Tests (Script A1)					
7	55	Cont	90	CQBIC_registers	1
C	54	Cont	C6	SSC_powerup	1
C	53	Cont	52	SSC_Prog_timers	1
C	52	Cont	52	SSC_Prog_timers	1
C	51	Cont	53	SSC_TOY_Clock	1
C	50	Cont	C1	SSC_RAM_Data	1
C	49	Cont	34	SSC_ROM	1
C	48	Cont	C5	SSC_registers	1
B	47	Cont	55	Interval_Timer	1
8	46	Cont	4F	Memory_Data	2, 1
8	45	Cont	4E	Memory_Byte	2, 1
8	44	Cont	4B	Memory_Byte_Errors	2, 1
8	43	Cont	4A	Memory_ECC_SBEs	2, 1
8	42	Cont	4C	Memory_ECC_Logic	2, 1
8	41	Cont	48	Memory_Addr_shorts	2, 1
8	40	Cont	48	Memory_addr_shorts	2, 1
8	39	Cont	48	Memory_addr_shorts	2, 1
8	38	Cont	48	Memory_addr_shorts	2, 1
8	37	Cont	48	Memory_addr_shorts	2, 1
8	36	Cont	48	Memory_addr_shorts	2, 1

[1]Field-replaceable unit key:

1 = KA50
2 = MS44
3 = Q22–bus option
4 = System power supply
5 = SCSI device or 2 devices with same target id
6 = ASYNC option board
7 = COMM option board (SYNC)
8 = SHAC option board

(continued on next page)

Table 5–4 (Cont.) KA50/51/55/56 Console Displays as Pointers to FRUs

On Error Hex LED	Normal Console Display	Default Action on Error	Failing Test Number	Test Description	FRU[1]
Power-Up Tests (Script A1)					
8	35	Cont	48	Memory_addr_shorts	2, 1
8	34	Cont	48	Memory_addr_shorts	2, 1
8	33	Cont	4D	Memory_address	2, 1
8	32	Cont	47	Memory_Refresh	2, 1
8	31	Halt	40	Memory_count_pages	2, 1
8	30	Cont	40	Memory_count_pages	2, 1
6	29	Cont	E4	DZ	1
B	28	Cont	54	Virtual_Mode	1
9	27	Cont	37	Cache_W_memory	1, 2
C	26	Cont	C2	SSC_RAM_Data_Addr	1
7	25	Cont	80	CQBIC_memory	1, 2
9	24	Cont	37	Cache_w_memory	1, 2
A	23	Cont	51	FPA	1
5	22	Cont	E2	SCSI_MAP	1
5	21	Cont	E0	SCSI	1, 5
4	20	Cont	5F	SGEC	1
5	19	Cont	5C	SHAC	8, 1
B	18	Cont	9A	INTERACTION	1
7	17	Cont	83	QZA_Intlpbck1	3
7	16	Cont	84	QZA_Intlpbck2	3

[1]Field-replaceable unit key:

1 = KA50
2 = MS44
3 = Q22-bus option
4 = System power supply
5 = SCSI device or 2 devices with same target id
6 = ASYNC option board
7 = COMM option board (SYNC)
8 = SHAC option board

(continued on next page)

Table 5–4 (Cont.) KA50/51/55/56 Console Displays as Pointers to FRUs

On Error Hex LED	Normal Console Display	Default Action on Error	Failing Test Number	Test Description	FRU[1]
Power-Up Tests (Script A1)					
7	15	Cont	85	QZA_memory	3
7	14	Cont	86	QZA_DMA	3
7	13	Cont	63	QDSS_any	3
7	12	Cont	63	QDSS_any	3
B	11	Cont	DB	Speed	1
7	10	Cont	EC	ASYNC	6, 1
7	09	Cont	E8	SYNC	7, 1
C	08	Cont	52	SSC_Prog_timers	1
C	07	Cont	52	SSC_Prog_timers	1
C	06	Cont	53	SSC_TOY_Clock	1
C	05	Cont	C1	SSC_RAM_Data	1
B	04	Cont	55	Interval_Timer	1
B	03	Cont	41	Board_Reset	1, 3

[1]Field-replaceable unit key:

1 = KA50
2 = MS44
3 = Q22–bus option
4 = System power supply
5 = SCSI device or 2 devices with same target id
6 = ASYNC option board
7 = COMM option board (SYNC)
8 = SHAC option board

5.3.1 FE Utility

In addition to the diagnostic console display and the LED code, the FE utility dumps the diagnostic state to the console (Example 5–12). This state indicates the major and minor test code of the test that failed, the 10 parameters associated with the test, and additional diagnostic state information.

Example 5–12 FE Utility Example

```
>>>T FE

Bitmap=00FF3000, Length=00001000, Checksum=807F, Busmap=00FF8000
Test_number=00, Subtest=00, Loop_Subtest=00, Error_type=00
Error_vector=0060, Severity=02, Last_exception_PC=20057C37
Total_error_count=0004, Led_display=08, Console_display=81, save_mchk_code=00
parameter_1=00000082  2=00000000  3=2000146A  4=00000000  5=20051400
parameter_6=00000001  7=00000000  8=00000020  9=00000000 10=00000000
previous_errors, Test  Subtest  Loop_Subtest  Error_Type
  Test_81_02_00_FE   Test_40_06_00_FF   Test_E8_03_00_FF   Test_E4_02_00_FF
Flags=FFFF FFFCFC 0408443E   BCache_Disable=06  KA50  128KB BC  14.0 ns
Return_stack=201406CC, Subtest_pc=2005D7FF, Timeout=000007D0

>>>
```

5.3.2 Overriding Halt Protection

The ROM diagnostics are run in halt-protected space during execution after power-up of the system. During this time they cannot normally be halted with the BREAK key or the HALT button. After power-up is complete, all diagnostics including the power-up script (A1) or (0) are run with halts enabled allowing a user to stop a script or test. The preferred method to stop scripts is to use CONTROL C first.

5.3.3 Isolating Memory Failures

This section describes procedures for isolating memory subsystem failures.

Memory tests numbers are DC, 31, 30, 4F, 4E, 4B, 4A, 4C, 48, 4D, 47 and 40. All of these tests are run during power-up.

Normally, if one or more of these tests fail during power-up at the end of power-up the diagnostic executive will execute the SHOW MEMORY command automatically to help identify the memory failure. In all cases of a memory failure, the primary means to isolate to the FRU is to use the SHOW MEMORY command.

Example 5–13 shows a memory failure due to a missing SIMM. In this case only one 16-MB set (4 SIMMs of 4 MB each) is present, and one of these is missing. Because of this, test DC fails and the power-up script is halted because no usable memory is present. At the end, SHOW MEMORY is automatically executed before the test is halted. In this example, SIMM set 1 (1E,1F,1G,1H) is present but SIMM 1F is either missing or not correctly installed in its socket.

Example 5–13 Failure Due to a Missing SIMM (One 16 Mbyte Set)

```
KA50-A V1.2, VMB 2.14
Performing normal system tests.
72..71..70..69..68..67..66..65..64..63..62..61..60..59..

? Test_Subtest_DC_88  Loop_Subtest=05  Err_Type=FF   DE_NO_Memory_present.lis
Vec=0000  Prev_Errs=0000   P1=C90AC90A   P2=00000000   P3=00000000  P4=00001006
P5=00000000  P6=7F7F7F33   P7=00000000   P8=00000000   P9=FFFF0000 P10=200636E4
r0=00000008  r1=21018000   r2=C90AC90A   r3=80000000   r4=01000000  r5=04000000
r6=00000002  r7=00000000   r8=00000000   r9=20140758  r10=FFFFFFFE r11=FFFFFFFF
 dser=0000 cesr=00000000 intmsk=00 icsr=01 pcsts=FA00 pcadr=FFFFFFF8 pcctl=FE13
 cctl=00000006 bcetsts=03E0 bcedsts=0F00 cefsts=0001EC20 nests=00
 mmcdsr=01FFFE40 mesr=00000000

Error:  SIMM Set 1 (1E,1F,1G,1H), SSR = C90A
 SIMM_1E = 16MB      SIMM_1F = 00MB ??   SIMM_1G = 16MB      SIMM_1H = 16MB

Total of 0MB, 0 good pages, 0 bad pages, 0 reserved pages
Normal operation not possible.
>>>
```

Note

The value listed by each SIMM is either 16 MB or 64 MB which indicates the full size of the set of SIMMs if all are present.

ACTION:

- If SIMM 1F is missing, install a SIMM.
- If SIMM 1F is present in socket, reseat the SIMM.
- If reseating SIMM 1F does not fix the problem, replace the SIMM with a new SIMM.
- At this point the system board is probably bad. If no new system board is available, try moving the SIMMs to the other set of sockets.

Example 5–14 shows a memory failure due to a missing SIMM. In this case two 16-MB sets (4 SIMMs of 4 MB each) are present with one SIMM missing in Set 1. Since one set of memory is fully usable, all testing is completed. At the end SHOW MEMORY is automatically executed as before. SIMM 1H is missing or not installed correctly. The system is usable but with only 16 MB of memory instead of 32 MB.

Example 5–14 Failure Due to a Missing SIMM (Two 16 Mbyte Sets)

```
KA50-A T1.2-156, VMB 2.14
Performing normal system tests.
72..71..70..69..68..67..66..65..64..63..62..61..60..59..58..

? Test_Subtest_31_06  Loop_Subtest=05  Err_Type=FF   DE_Memory_Setup_CSRs.lis
Vec=0000  Prev_Errs=0000   P1=C94AC94A   P2=01000000   P3=00000002  P4=00000000
P5=25800000  P6=FFFFFFFF   P7=00000000   P8=00000000   P9=0000C94A P10=C94AC14A
r0=00000008  r1=21018000   r2=C94AC94A   r3=81000000   r4=01000000  r5=04000000
r6=00000002  r7=21018048   r8=00000000   r9=20140758  r10=FFFFFFFE r11=FFFFFFFF
 dser=0000 cesr=00000000 intmsk=00 icsr=01 pcsts=FA00 pcadr=FFFFFFF8 pcctl=FE13
 cctl=00000006 bcetsts=0360 bcedsts=0F00 cefsts=00206E20 nests=00
 mmcdsr=01FFFE00 mesr=00000000

57..56..55..54..53..52..51..50..49..48..47..46..45..44..43..42..
41..40..39..38..37..36..35..34..33..32..31..30..29..28..27..26..
25..24..23..22..21..20..19..18..17..16..15..14..13..12..11..10..
09..08..07..06..05..04..03..

16 MB RAM, SIMM Set (0A,0B,0C,0D) present
Memory Set 0: 00000000 to 00FFFFFF, 16MB, 32768 good pages, 0 bad pages

Error:  SIMM Set 1 (1E,1F,1G,1H), SSR = C94A
 SIMM_1E = 16MB      SIMM_1F = 16MB      SIMM_1G = 16MB      SIMM_1H = 00MB ??

Total of 16MB, 32768 good pages, 0 bad pages, 104 reserved pages

Normal operation not possible.

>>>
```

ACTION:

- If SIMM 1H is missing, install a SIMM.
- If SIMM 1H is present in socket, reseat the SIMM.
- If reseating SIMM 1H does not fix the problem then replace the SIMM with a new SIMM.
- At this point the system board is probably bad.

Example 5–15 shows a memory failure due to a bad SIMM. In this case two 16-MB sets (4 SIMMs of 4 MB each) are present with one bad SIMM. SIMM 1H is marked as being bad.

Example 5–15 Failure Due to a Bad SIMM

```
KA50-A V1.2, VMB 2.14
Performing normal system tests.
72..71..70..69..68..67..66..65..64..63..62..61..60..59..58..57..
56..55..54..53..52..51..50..49..48..47..46..45..44..43..42..41..
40..39..38..37..36..35..34..33..32..31..30..

? Test_Subtest_40_06  Loop_Subtest=00  Err_Type=FF   DE_Memory_count_pages.lis

29..28..27..26..25..24..23..22..21..20..19..18..17..16..15..14..
13..12..11..10..09..08..07..06..05..04..03..

16 MB RAM, SIMM Set (0A,0B,0C,0D) present
Memory Set 0: 00000000 to 00FFFFFF, 16MB, 32768 good pages, 0 bad pages

Error:  SIMM Set 1 (1E,1F,1G,1H), SSR = C14A
 SIMM_1E = 16MB      SIMM_1F = 16MB      SIMM_1G = 16MB      SIMM_1H = 16MB ??
Memory Set 1: 01000000 to 01FFFFFF, 16MB, 0 good pages, 32768 bad pages

Total of 32MB, 32768 good pages, 32768 bad pages, 112 reserved pages

>>>
```

ACTION

- Reseat the SIMM 1H.
- If reseating SIMM 1H does not fix the problem then replace the SIMM with a new SIMM.
- At this point the system board is probably bad.

Example 5–16 indicates that a large SIMM is mixed in with a set of small SIMMs. If a full set of SIMMs is present and one or more is the incorrect size then the diagnostic code will configure the set as a small set and run the tests. In this example, SIMM 1G is the wrong size SIMM. Because the set is configured as a small set, it is usable as a 16-MB set.

Example 5–16 SIMM Wrong Size

```
Error:  SIMM Set 1 (1E,1F,1G,1H), SSR = C14A
 SIMM_1E = 16MB       SIMM_1F = 16MB       SIMM_1G = 64MB ??   SIMM_1H = 16MB
Memory Set 1: 01000000 to 01FFFFFF, 16MB, 32768  good pages, 0 bad pages
```

ACTION:

Replace SIMM 1G with one of the correct size.

The diagnostics cannot always determine which SIMM caused a failure. If this occurs and more than one set is present, usually the failing set can be identified by using the **SHOW MEMORY** command.

```
>>>SHOW MEMORY

16 MB RAM, SIMM Set (0A,0B,0C,0D) present
Memory Set 0: 00000000 to 00FFFFFF, 16MB, 32768 good pages, 0 bad pages

16 MB RAM, SIMM Set (1E,1F,1G,1H) present
Memory Set 1: 01000000 to 01FFFFFF, 16MB, 0 good pages, 32768 bad pages

Total of 32MB, 32768 good pages, 32768 bad pages, 112 reserved pages

>>>
```

ACTION:

Replace SIMM set 1 (1E,1F,1G,1H).

After installing a new set of SIMMs and successfully running power-up tests, run memory test script A8.

>>>T A8

Note

Script A9 is another memory test script. This script will stop on the first occurrence of any error. It will also stop on a soft error. If a failure occurs in A9 and if A9 then runs successfully 10 times and script A8 runs without error the problem is a soft error and does not require action.

Note

If a memory failure is marked in the bitmap, it will not be erased until either the system is powered up or the bitmap placing test is run with

parameter P4 set to 0 to rebuild the bitmap.

To force rebuilding the bitmap to all good memory, enter the following commands:

```
T 30 0 0 0 0   ; T 30 will not work by itself.
T 0            ; rerun powerup script
```

5.4 Using MOP Ethernet Functions to Isolate Failures

The console requester can receive LOOPED_DATA messages from the server by sending out a LOOP_DATA message using NCP to set this up. An example follows.

Identify the Ethernet adapter address for the system under test (system 1) and attempt to boot over the network.

```
***system 1 (system under test)***

>>>SHOW ETHERNET
Ethernet Adapter
-EZA0 (08-00-2B-28-18-2C)
>>>BOOT EZA0
(BOOT/R5:2 EZA0)

  2..
-EZA0
Retrying network bootstrap.
```

Unless the system is able to boot, the "Retrying network bootstrap" message will display every 8–12 minutes.

Identify the system's Ethernet circuit and circuit state, enter the SHOW KNOWN CIRCUITS command from the system conducting the test (system 2).

```
***system 2 (system conducting test)***

$ MCR NCP
NCP>SHOW KNOWN CIRCUITS

Known Circuit Volatile Summary as of 14-NOV-1991 16:01:53

   Circuit          State                 Loopback     Adjacent
                                            Name      Routing Node

  ISA-0             on                                25.1023 (LAR25)
```

```
NCP>SET CIRCUIT ISA-0 STATE OFF
NCP>SET CIRCUIT ISA-0 SERVICE ENABLED
NCP>SET CIRCUIT ISA-0 STATE ON
NCP>LOOP CIRCUIT ISA-0 PHYSICAL ADDRESS 08-00-2B-28-18-2C
WITH ZEROES
NCP>EXIT
$
```

If the loopback message was received successfully, the NCP prompt will reappear with no messages.

The following two examples show how to perform the Loopback Assist Function using another node on the network as an assistant (system 3) and the system under test as the destination. Both the assistant and the system under test are attempting to boot from the network. We will also need the physical address of the assistant node.

```
***system #3 (loopback assistant)***

>>>SHOW ETHERNET
Ethernet Adapter
-EZA0 (08-00-2B-1E-76-9E)
>>>b eza0
(BOOT/R5:2 EZA0)

  2..
-EZA0
Retrying network bootstrap.

***system 2***

NCP>LOOP CIRCUIT ISA-0 PHYSICAL ADDRESS 08-00-2b-28-18-2C ASSISTANT PHYSICAL
ADDRESS 08-00-2B-1E-76-9E WITH MIXED COUNT 20 LENGTH 200 HELP FULL
NCP>
```

Instead of using the physical address, you could use the assistant node's area address. When using the area address, system 3 is running the OpenVMS operating system.

```
***system 3***
$MCR NCP

NCP>SHOW NODE KLATCH

Node Volatile Summary as of 27-FEB-1992 21:04:11

Executor node = 25.900 (KLATCH)

State                     = on
Identification            = DECnet-VAX V5.4-1,  OpenVMS V5.4-2
Active links              = 2

NCP>SHOW KNOWN LINES CHARACTERISTICS

Known Line Volatile Characteristics as of 27-FEB-1992 11:20:50

Line = ISA-0
```

```
Receive buffers          = 6
Controller               = normal
Protocol                 = Ethernet
Service timer            = 4000
Hardware address         = 08-00-2B-1E-76-9E
Device buffer size       = 1498

NCP>SET CIRCUIT ISA-0 STATE OFF
NCP>SET CIRCUIT ISA-0 SERVICE ENABLED
NCP>SET CIRCUIT ISA-0 STATE ON
NCP>EXIT
$

***system 2***
$ MCR NCP
NCP>LOOP CIRCUIT ISA-0 PHYSICAL ADDRESS 08-00-2B-28-18-2C ASSISTANT MODE 25.900
WITH MIXED COUNT 20 LENGTH 200 HELP FULL
NCP>EXIT
$
```

Note

The kernel's Ethernet buffer is 1024 bytes deep for the LOOP functions and will not support the maximum 1500-byte transfer length.

In order to verify that the address is reaching this node, a remote node can examine the status of the periodic SYSTEM_IDs sent by the KA50/51/55/56 Ethernet server. The SYSTEM_ID is sent every 8–12 minutes using NCP as in the following example:

```
***system 2***

$ MCR NCP
NCP>SET MODULE CONFIGURATOR CIRCUIT ISA-0 SURVEILLANCE ENABLED
NCP>SHOW MODULE CONFIGURATOR KNOWN CIRCUITS STATUS TO ETHER.LIS
NCP>EXIT
$ TYPE ETHER.LIS

Circuit name             = ISA-0
Surveillance flag        = enabled
Elapsed time             = 00:09:37
Physical address         = 08-00-2B-28-18-2C
Time of last report      = 27-Feb 11:50:34
Maintenance version      = V4.0.0
Function list            = Loop, Multi-block loader, Boot, Data link counters
Hardware address         = 08-00-2B-28-18-2C
Device type              = ISA
```

Depending on your network, the file used to receive the output from the SHOW MODULE CONFIGURATOR command may contain many entries, most of which do not apply to the system you are testing. It is helpful to use an editor to search the file for the Ethernet hardware address of the system under test.

Existence of the hardware address verifies that you are able to receive the address from the system under test.

5.5 Interpreting User Environmental Test Package (UETP) OpenVMS Failures

When UETP encounters an error, it reacts like a user program. It either returns an error message and continues, or it reports a fatal error and terminates the image or phase. In either case, UETP assumes the hardware is operating properly and it does not attempt to diagnose the error.

If the cause of an error is not readily apparent, use the following methods to diagnose the error:

- *OpenVMS Error Log Utility*—Run the Error Log Utility to obtain a detailed report of hardware and system errors. Error log reports provide information about the state of the hardware device and I/O request at the time of each error. For information about running the Error Log Utility, refer to the *OpenVMS Error Log Utility Manual* and Section 5.2 of this manual.
- *Diagnostic facilities*—Use the diagnostic facilities to test exhaustively a device or medium to isolate the source of the error.

5.5.1 Interpreting UETP Output

You can monitor the progress of UETP tests at the terminal from which they were started. This terminal always displays status information, such as messages that announce the beginning and end of each phase and messages that signal an error.

The tests send other types of output to various log files, depending on how you started the tests. The log files contain output generated by the test procedures. Even if UETP completes successfully, with no errors displayed at the terminal, it is good practice to check these log files for errors. Furthermore, when errors are displayed at the terminal, check the log files for more information about their origin and nature.

5.5.1.1 UETP Log Files

UETP stores all information generated by all UETP tests and phases from its current run in one or more UETP.LOG files, and it stores the information from the previous run in one or more OLDUETP.LOG files. If a run of UETP involves multiple passes, there will be one UETP.LOG or one OLDUETP.LOG file for each pass.

At the beginning of a run, UETP deletes all OLDUETP.LOG files, and renames any UETP.LOG files to OLDUETP.LOG. Then UETP creates a new UETP.LOG file and stores the information from the current pass in the new file. Subsequent passes of UETP create higher versions of UETP.LOG. Thus, at the end of a run of UETP that involves multiple passes, there is one UETP.LOG file for each pass. In producing the files UETP.LOG and OLDUETP.LOG, UETP provides the output from the two most recent runs.

If the run involves multiple passes, UETP.LOG contains information from all the passes. However, only information from the latest run is stored in this file. Information from the previous run is stored in a file named OLDUETP.LOG. Using these two files, UETP provides the output from its tests and phases from the two most recent runs.

The cluster test creates a NETSERVER.LOG file in SYS$TEST for each pass on each system included in the run. If the test is unable to report errors (for example, if the connection to another node is lost), the NETSERVER.LOG file on that node contains the result of the test run on that node. UETP does not purge or delete NETSERVER.LOG files; therefore, you must delete them occasionally to recover disk space.

If a UETP run does not complete normally, SYS$TEST might contain other log files. Ordinarily these log files are concatenated and placed within UETP.LOG. You can use any log files that appear on the system disk for error checking, but you must delete these log files before you run any new tests. You may delete these log files yourself or rerun the entire UETP, which checks for old UETP.LOG files and deletes them.

5.5.1.2 Possible UETP Errors

This section is intended to help you identify problems you might encounter running UETP.

The following are the most common failures encountered while running UETP:

- Wrong quotas, privileges, or account
- UETINIT01 failure
- Ethernet device allocated or in use by another application
- Insufficient disk space
- Incorrect VAXcluster setup
- Problems during the load test
- DECnet–VAX error
- Lack of default access for the FAL object

- Errors logged but not displayed
- No PCB or swap slots
- Hangs
- Bug checks and machine checks

For more information refer to the *VAX 3520, 3540 OpenVMS Installation and Operations (ZKS166)* manual.

5.6 Using Loopback Tests to Isolate Failures

You can use external loopback tests to isolate problems with the console port, and Ethernet controller (SGEC chip).

5.6.1 Testing the Console Port

To test the console port at power-up, set the Power-Up Mode switch on the console module to the Loop Back Test Mode position (bottom) and install an H3103 loopback connector into the MMJ. The H3103 connects the console port transmit and receive lines. At power-up, the SLU_EXT_LOOPBACK test then runs a continuous loopback test.

While the test is running, the LED display on the console module should alternate between 6 and 3. A value of 6 latched in the display indicates a test failure. If the test fails, one of the following parts is faulty: the KA50/51/55/56 or the cabling.

To test out to the end of the console terminal cable:

1. Plug the MMJ end of the console terminal cable into the back BA42B.
2. Disconnect the other end of the cable from the terminal.
3. Place an H8572 adapter into the disconnected end of the cable.
4. Connect the H3103 to the H8572.
5. Cycle power and observe the LED.

5.6.2 Embedded Ethernet Loopback Testing

Note

Before running Ethernet loopback tests, check that the problem is not due to a missing terminator on a ThinWire T-connector.

Test 5F is the internal loopback test for SGEC (Ethernet controller).

```
>>>T 5F
```

For an external SGEC loopback, enter "1".

```
>>>T 5F 1
```

Before running test 5F on the ThinWire Ethernet port, connect an H8223 T-connector with two H8225 terminators.

Before running test 5F on the standard Ethernet port, you must have a 12–22196–02 loopback connector installed.

Note

Make sure the Ethernet Connector Switch is set for the correct Ethernet port.

T 59 polls other nodes on Ethernet to verify SGEC functionality. The Ethernet cable must be connected to a functioning Ethernet. A series of MOP messages are generated; look for response messages from other nodes.

```
>>>T 59

Reply received from node: AA-00-04-00-FC-64
Total responses: 1
Reply received from node: AA-00-04-00-47-16
Total responses: 2
Reply received from node: 08-00-2B-15-48-70
Total responses: 3
   .
   .
   .
Reply received from node: AA-00-04-00-17-14
Total responses: 25
>>>
```

Table 5–5 Loopback Connectors for Common Devices

Device	Module Loopback	Cable Loopback
CXA16/CXB16	H3103 + H8572[1]	–
CXY08	H3046 (50-pin)	H3197 (25-pin)
DIV32	H3072	–
DPV11	12–15336–10 or H325	H329 (12–27351–01)
DRQB3	–	17–01481–01 (from port 1 to port 2)
DRV1W	70–24767–01	–
DZQ11	12–15336–10 or H325	H329 (12–27351–01)
Ethernet[2]	–	–
IBQ01	IBQ01–TA	–
IEQ11	17–01988–01	–
KMV1A	H3255	H3251
KZQSA	12–30552–01	–
LPV11	12–15336–11	–

[1]Use the appropriate cable to connect transmit-to-receive lines. H3101 and H3103 are double-ended cable connectors.

[2]For ThinWire, use H8223–00 plus two H8225–00 terminators. For standard Ethernet, use 12–22196–02.

6

FEPROM Firmware Update

Note

The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95, and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, will appear on the console and/or printouts from time to time.

KA50/51/55/56 firmware is located on four chips, each 128 K by 8 bits of FLASH programmable EPROMs, for a total of 512 Kbytes of ROM. (A FLASH EPROM (FEPROM) is a programmable read-only memory that uses electrical (bulk) erasure rather than ultraviolet erasure.)

FEPROMs provide nonvolatile storage of the CPU power-up diagnostics, console interface, and operating system primary bootstrap (VMB). An advantage of this technology is that the entire image in the FEPROMs may be erased, reprogrammed, and verified in place without removing the CPU module or replacing components.

A slight disadvantage to the FEPROM technology is that the entire part must be erased before reprogramming. Hence, there is a small "window of vulnerability" when the CPU has inoperable firmware. Normally, this window is less than 30 seconds. Nonetheless, an update should be allowed to execute undisturbed.

Firmware updates are provided through a package called the Firmware Update Utility. A Firmware Update Utility contains a bootable image, which can be booted from tape or Ethernet, that performs the FEPROM update. Firmware update packages, like software, are distributed through Digital's SSB. Service engineers are notified of updates through a service blitz or Engineering Change Order (ECO)/Field Change Order (FCO) notification.

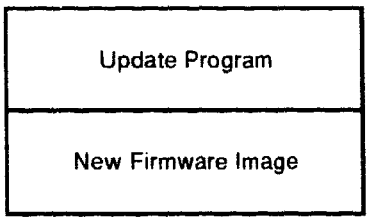
Note

The NVAX CPU chip has an area called the Patchable Control Store (PCS), which can be used to update the microcode for the CPU chip.

Updates to the PCS require a new version of the firmware.

A Firmware Update Utility image consists of two parts, the update program and the new firmware, as shown in Figure 6–1. The update program uniformly programs, erases, reprograms, and verifies the entire FEPROM.

Figure 6–1 Firmware Update Utility Layout



MLO-007271

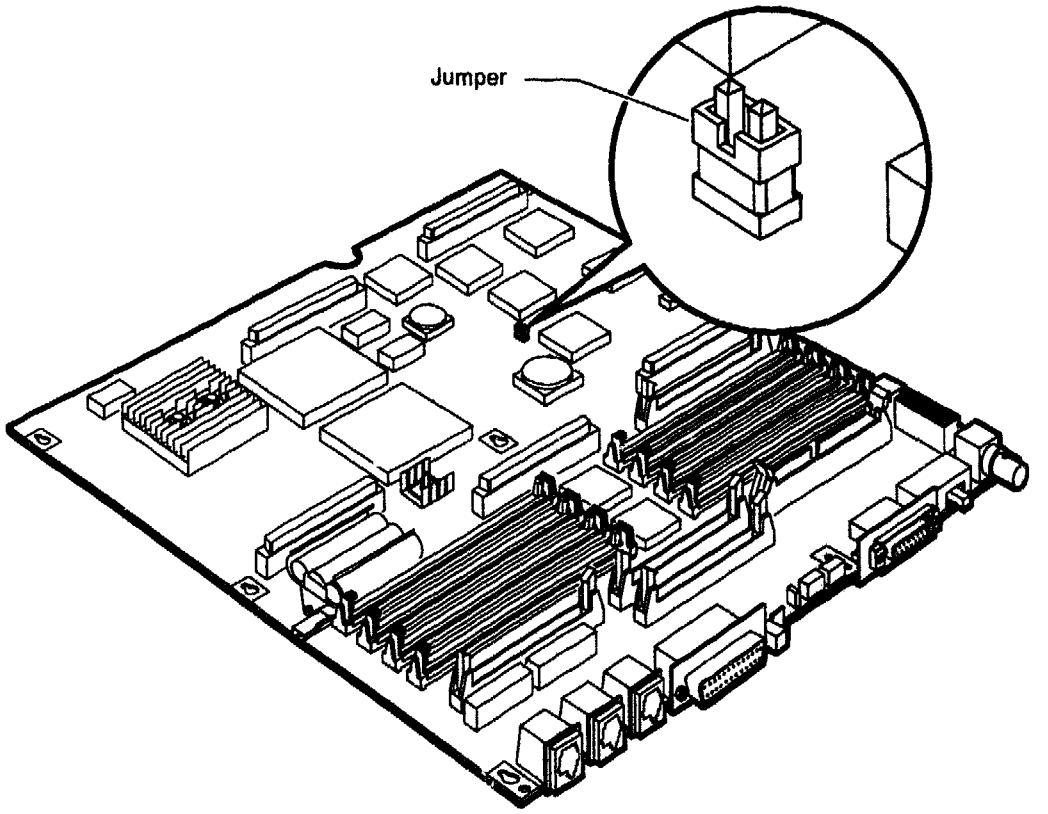
Once the update has completed successfully, normal operation of the system may continue. The operator may then either halt or reset the system and reboot the operating system.

6.1 Preparing the Processor for a FEPROM Update

Complete the following steps to prepare the processor for a FEPROM update:

1. The system manager should perform operating system shutdown.
2. Enter console mode by pressing the Halt button once to halt the system. If the Break Enable/Disable switch on the console module is set to enable (indicated by 1), you can halt the system by pressing the Break key on the console terminal.

Figure 6–2 W4 Jumper Setting for Updating Firmware



6.2 Updating Firmware via Ethernet

To update firmware via the Ethernet, the "client" system (the target system to be updated) and the "server" system (the system that serves boot requests) must be on the same Ethernet segment. The Maintenance Operation Protocol (MOP) is the transport used to copy the network image.

Use the following procedure to update firmware via the Ethernet:

1. Enable the server system's NCP circuit using the following OpenVMS commands:

```
$ MCR NCP
NCP>SET CIRCUIT <circuit> STATE OFF
NCP>SET CIRCUIT <circuit> SERVICE ENABLED
NCP>SET CIRCUIT <circuit> STATE ON
```

Where <circuit> is the system Ethernet circuit. Use the SHOW KNOWN CIRCUITS command to find the name of the circuit.

Note

The SET CIRCUIT STATE OFF command will bring down the system's network.

2. Copy the file containing the updated code to the MOM$LOAD area on the server (this procedure may require system privileges). Refer to the Firmware Update Utility Release Notes for the Ethernet bootable filename. Use the following command to copy the file:

   $ COPY <filename>.SYS MOM$LOAD:*.*

   Where <filename> is the Ethernet bootable filename provided in the release notes.

3. On the client system, enter the command BOOT/100 EZ at the console prompt (>>>).

   The system then prompts you for the name of the file.

Note

Do NOT type the ".SYS" suffix when entering the Ethernet bootfile name. The MOP load protocol only supports 15 character filenames.

4. After the FEPROM upgrade program is loaded, simply type "Y" at the prompt to start the FEPROM blast. Example 6–1 provides a console display of the FEPROM update program.

Caution

Once you enter the bootfile name, do not interrupt the FEPROM blasting program, as this can damage the CPU module. The program takes several minutes to complete.

Example 6–1 FEPROM Update via Ethernet

```
***** On Server System *****

$ MCR NCP
NCP>SET CIRCUIT ISA-0 STATE OFF
NCP>SET CIRCUIT ISA-0 SERVICE ENABLED
NCP>SET CIRCUIT ISA-0 STATE ON
NCP>EXIT
$
$ COPY KA50_V41_EE.SYS MOM$LOAD:*.*
$

***** On Client System *****

>>>b/100 eza0
(BOOT/R5:100 EZA0)

  2..

Bootfile: ka50_v12

-EZA0
  1..0..

 FEPROM update program

                        ---CAUTION---

 --- Executing this program will change your current FEPROM ---

Do you want to continue [Y/N] ? : y

Blasting in V1.2-41.   The program will take at most several minutes.

DO NOT ATTEMPT TO INTERRUPT PROGRAM EXECUTION
Doing so may result in loss of operable state !!!

+--------------------------------------+
10...9...8...7...6...5...4...3...2...1...0

FEPROM Programming successful

?06 HLT INST
        PC = 00008E24
>>>
```

Note

If the update does not work, check to be sure the "write enable" on-board jumper is installed (see Figure 6–2).

5. Recycle power or enter "T 0" at the console prompt (>>>).
6. If the customer requires, return the jumper on the module to the "write disable mode" setting.

6.3 Updating Firmware via Tape

To update firmware via tape, the system must have a TZ30, TF85, TK70, TK50 or TLZ04 tape drive.

If you need to make a bootable tape, copy the bootable image file to a tape as shown in the following example. Refer to the release notes for the name of the file.

```
$ INIT MKA500:"VOLUME_NAME"
$ MOUNT/BLOCK_SIZE = 512  MKA500:"VOLUME_NAME"
$ COPY/CONTIG <file_name> MkA500:<file_name>
$ DISMOUNT MKA500
$
```

Use the following procedure to update firmware via tape:

1. Be sure the on board jumper is in the correct ("write enable mode") position (Section 6.1).
2. At the console prompt (>>>), enter the BOOT/100 command for the tape device, for example: BOOT/100 MKA500.

   Use the SHOW DEVICE command if you are not sure of the device name for the tape drive.

   The system prompts you for the name of the file. Enter the bootfile name.
3. After the FEPROM upgrade program is loaded, simply type "Y" at the prompt to start the FEPROM blast. Example 6–2 provides a console display of the FEPROM update program.

   Caution

   Once you enter the bootfile name, do not interrupt the FEPROM blasting program, as this can damage the CPU module. The program takes several minutes to complete.

4. Press the Restart button on the SCP or enter "T 0" at the console prompt (>>>).
5. If the customer requires, return the jumper on the CPU module to the "write disable mode" setting.

Example 6–2 FEPROM Update via Tape

```
>>> BOOT/100 MKA500
(BOOT/R5:100 MKA500)

  2..
Bootfile: KA50_V41_BZ
-MKA500

  1..0..

 FEPROM update program

                         ---CAUTION---

 --- Executing this program will change your current FEPROM ---

Do you want to continue [Y/N] ? : y

Blasting in V1.2-41.   The program will take at most several minutes.

DO NOT ATTEMPT TO INTERRUPT PROGRAM EXECUTION
Doing so may result in loss of operable state !!!

+--------------------------------------+
10...9...8...7...6...5...4...3...2...1...0

FEPROM Programming successful

?06 HLT INST
        PC = 00008E24
>>>
```

6.4 FEPROM Update Error Messages

The following is a list of error messages generated by the FEPROM update program and actions to take if the errors occur.

MESSAGE:
?? ERROR update enable jumper is disconnected
unable to blast ROMs...
ACTION:
Reposition update enable jumper (Section 6.1).

MESSAGE:
?? ERROR, FEPROM programming failed
ACTION:
Turn off the system, then turn it on. If you see the banner message as expected, reenter console mode and try booting the update program again. If you do not see the usual banner message, replace the CPU module.

Patchable Control Store (PCS) Loading Error Messages

The following is a list of error messages that may appear if there is a problem with the PCS. The PCS is loaded as part of the power-up stream (before ROM-based diagnostics are executed).

MESSAGE:
CPU is not an NVAX
COMMENT:
CPU_TYPE as read in NVAX SID is not = 19 (decimal), as is should be for an NVAX processor.

MESSAGE:
Microcode patch/CPU rev mismatch
COMMENT:
Header in microcode patch does not match MICROCODE_REV as read in NVAX SID.

MESSAGE:
PCS Diagnostic failed
COMMENT:
Something is wrong with the PCS. Replace the NVAX chip (or CPU module).

A

Address Assignments

Note

The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95 and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, will appear on the console and/or printouts from time to time.

A.1 KA50/51/55/56 General Local Address Space Map

Address Assignments

A.1 KA50/51/55/56 General Local Address Space Map

VAX Memory Space

Address Range	Contents
0000 0000 - 1FFF FFFF	Local Memory Space (512MB)

VAX I/O Space

Address Range	Contents
2000 0000 - 2000 1FFF	Local Q22-Bus I/O Space (8KB)
2000 2000 - 2003 FFFF	Reserved Local I/O Space (248KB)
2008 0000 - 201F FFFF	Local Register I/O Space (1.5MB)
2020 0000 - 23FF FFFF	Reserved Local I/O Space (62.5MB)
2400 0000 - 27FF FFFF	Reserved Local I/O Space (64MB)
2008 0000 - 2BFF FFFF	Reserved Local I/O Space (64MB)
2C08 0000 - 2FFF FFFF	Reserved Local I/O Space (64MB)
3000 0000 - 303F FFFF	Local Q22-Bus Memory Space (4MB)
3040 0000 - 33FF FFFF	Reserved Local I/O Space (60MB)
3400 0000 - 37FF FFFF	Reserved Local I/O Space (64MB)
3800 0000 - 3BFF FFFF	Reserved Local I/O Space (64MB)
3C00 0000 - 3FFF FFFF	Reserved Local I/O Space (64MB)
E004 0000 - E007 FFFF	Local ROM Space

A.2 KA50/51/55/56 Detailed Local Address Space Map

Local Memory Space (up to 128MB)	0000 0000 - 7FF FFFF
Q22-bus Map - top 32KB of Main Memory	
VAX I/O Space	
Local Q22-bus I/O Space	2000 0000 - 2000 1FFF
Reserved Q22-bus I/O Space	2000 0000 - 2000 0007
Q22-bus Floating Address Space	2000 0008 - 2000 07FF
User Reserved Q22-bus I/O Space	2000 0800 - 2000 0FFF
Reserved Q22-bus I/O Space	2000 1000 - 2000 1F3F
Interprocessor Comm Reg	2000 1F40
Reserved Q22-bus I/O Space	2000 1F44 - 2000 1FFF
Local Register I/O Space	2000 2000 - 2003 FFFF
Reserved Local Register I/O Space	2000 4000 - 2000 422F
Reserved Local Register I/O Space	2000 42B0 - 2000 7FFF
Reserved Local Register I/O Space	2000 40B0 - 2000 422F
NICSR0 - Vector Add, IPL, Sync/Async	2000 8000
NICSR1 - Polling Demand Register	2000 8004
NICSR2 - Reserved	2000 8008
NICSR3 - Receiver List Address	2000 800C
NICSR4 - Transmitter List Address	2000 8010
NICSR5 - Status Register	2000 8014
NICSR6 - Command and Mode Register	2000 8018
NICSR7 - System Base Address	2000 801C
NICSR8 - Reserved	2000 8020*
NICSR9 - Watchdog Timers	2000 8024*
NICSR10- Reserved	2000 8028*
NICSR11- Rev Num & Missed Frame Count	2000 802C*
NICSR12- Reserved	2000 8030*
NICSR13- Breakpoint Address	2000 8034*
NICSR14- Reserved	2000 8038*
NICSR15- Diagnostic Mode & Status	2000 803C
Reserved Local Register I/O Space	2000 8040 - 2003 FFFF

A.2 KA50/51/55/56 Detailed Local Address Space Map

KA50/51/55/56 DETAILED LOCAL ADDRESS SPACE MAP (Cont.)

```
**********************************************************************
*
*  Q-22 Bus Local Register I/O Space          2008 0000 - 201F FFFF
*     DMA System Configuration Register       2008 0000
*     DMA System Error Register               2008 0004
*     DMA Master Error Address Register       2008 0008
*     DMA Slave Error Address Register        2008 000C
*     Q22-bus Map Base Register               2008 0010
*     Reserved Local Register I/O Space       2008 0014 - 2008 00FF
*
**********************************************************************

    Reserved Local Register I/O Space         2008 0194 - 2008 3FFF

    Boot and Diagnostic Reg (32 Copies)       2008 4000 - 2008 407C

    Reserved Local Register I/O Space         2008 4080 - 2008 7FFF

**********************************************************************
*
*   Q22-bus Map Registers                     2008 8000 - 2008 FFFF
*   Reserved Local Register I/O Space         2009 0000 - 2013 FFFF
*
**********************************************************************

SSC CSRs

    SSC Base Address Register                 2014 0000
    SSC Configuration Register                2014 0010
    CP Bus Timeout Control Register           2014 0020
    Diagnostic LED Register                   2014 0030
    Reserved Local Register I/O Space         2014 0034 - 2014 006B
```

KA50/51/55/56 DETAILED LOCAL ADDRESS SPACE MAP (Cont.)

VAX IPRs implemented by NCA	
Interval Clock Control Status Reg	2100 0060
Next Interval Count Register	2100 0064
Interval Count Register	2100 0068
NMC CSRs	
O-bit Data Registers	2101 0000 - 2101 7FFF
Main Memory Configuration Reg 0	2101 8000
Main Memory Configuration Reg 1	2101 8004
Main Memory Signature Register 0	2101 8020
Main Memory Signature Register 1	2101 8024
Main Memory Error Address Register	2101 8040
Main Memory Error Status Register	2101 8044
Main Memory Mode Control and Diagnostic Register	2101 8048
O-bit Address and Mode Register	2101 804C
NCA CSRs	
Error Status Register	2102 0000
Mode Control and Diagnostic Reg	2102 0004
CP1 Slave Error Address Register	2102 0006
CP2 Slave Error Address Register	2102 000C
CP1 IO Error Address Register	2102 0010
CP2 IO Error Address Register	2102 0014
NDAL Error Address Register	2102 0018

```
KA50/51/55/56 DETAILED LOCAL ADDRESS SPACE MAP (Cont.)

*****************************************************************
* OPTIONAL KZDDA SCSI CONTROLLER
*
*   SCSI  DMA address register                21C00000
*   SCSI  DMA direction register              21C00004
*   Interrupt mask register                   21C00008
*   Interrupt pending register                21C0000C
*   SCSI Controller (53C94) registers         22000080 - 220000B0
*      (13 byte regs (0:9,A,B,C) on LW boundary)
*                              scsicsr0       22000080
*                              scsicsr1       22000084
*                              scsicsr2       22000088
*                              scsicsr3       2200008C
*                              scsicsr4       22000090
*                              scsicsr5       22000094
*                              scsicsr6       22000098
*                              scsicsr7       2200009C
*                              scsicsr8       220000A0
*                              scsicsr9       220000A4
*                              scsicsra       220000A8
*                              scsicsrb       220000AC
*                              scsicsrc       220000B0
*   SCSI DMA Map registers                    23000000 - 23007FFF
*   (8,192 32 bit registers)
*****************************************************************

EDAL BUS DEVICES

*****************************************************************
* OPTIONAL SYNC COMMUNICATION DEVICE
*
*   Register sets of the SYNC ports       2400 0000 - 24FF FFFF
*   Option ROM Space          ???? ???? - ???? ????
*
*****************************************************************
    QUART (DC7085) Registers          2500 0000 - 2500 0007
    SCSI DMA Address Register        25C0 0000
    SCSI DMA Direction Register       25C0 0004
    Interrupt Mask Register         25C0 0008
    Interrupt Pending Register        25C0 000C
    SCSI Controller (53C94) Registers         2600 0080 - 2600 00BF
    SCSI DMA Map Registers         2700 0000 - 2700 7FFF
```

```
KA50/51/55/56 DETAILED LOCAL ADDRESS SPACE MAP (Cont.)
****************************************************************
* OPTIONAL ASYNC COMMUNICATION DEVICE
*
*   Register sets of the ASYNC ports       3E00 0000 - 3E00 000E
*   Option ROM Space          3E01 0000 - 3E02 FFFF
*
****************************************************************

Local FEPROM Space                                 E004 0000 - E007 FFFF
    VAX System Type Register (In ROM)              E004 0004
    Local FEPROM - (Halt Protected)                E004 0000 - E007 FFFF

************************************************************************
The following addresses allow those KA50/51/55/56 Internal Processor
Registers that are implemented in the SSC chip (External, Internal
Processor Registers) to be accessed via the local I/O page. These
addresses are documented for diagnostic purposes only and should
not be used by non-diagnostic programs.

    Time Of Year Register                          2014 006C

    Console Storage Receiver Status                2014 0070*
    Console Storage Receiver Data                  2014 0074*
    Console Storage Transmitter Status             2014 0078*
    Console Storage Transmitter Data               2014 007C*
    Console Receiver Control/Status                2014 0080
    Console Receiver Data Buffer                   2014 0084
    Console Transmitter Control/Status             2014 0088
    Console Transmitter Data Buffer                2014 008C
    Reserved Local Register I/O Space              2014 0090 - 2014 00DB

    I/O Bus Reset Register                         2014 00DC
    Reserved Local Register I/O Space              2014 00E0

    Reserved Local Register I/O Space              2014 00FC - 2014 00FF

  * These registers are not fully implemented, accesses yield
    UNPREDICTABLE results.

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

KA50/51/55/56 DETAILED LOCAL ADDRESS SPACE MAP (Cont.)

Local Register I/O Space (Cont.)	
Timer 0 Control Register	2014 0100
Timer 0 Interval Register	2014 0104
Timer 0 Next Interval Register	2014 0108
Timer 0 Interrupt Vector	2014 010C
Timer 1 Control Register	2014 0110
Timer 1 Interval Register	2014 0114
Timer 1 Next Interval Register	2014 0118
Timer 1 Interrupt Vector	2014 011C
Reserved Local Register I/O Space	2014 0120 - 2014 012F
BDR Address Decode Match Register	2014 0140
BDR Address Decode Mask Register	2014 0144
Reserved Local Register I/O Space	2014 0138 - 2014 03FF
Battery Backed-Up RAM	2014 0400 - 2014 07FF
Reserved Local Register I/O Space	2014 0800 - 201F FFFF
Reserved Local I/O Space	2020 0000 - 2FFF FFFF
Local Q22-bus Memory Space	3000 0000 - 303F FFFF
Reserved Local Register I/O Space	3040 0000 - 3FFF FFFF

A.3 External, Internal Processor Registers

Several of the Internal Processor Registers (IPR's) on the KA50/51/55/56 are implemented in the NCA or SSC chip rather than the CPU chip. These registers are referred to as External Internal Processor Registers and are listed below.

IPR #	Register Name	Abbrev.
27	Time of Year Register	TOY
28	Console Storage Receiver Status	CSRS*
29	Console Storage Receiver Data	CSRD*
30	Console Storage Transmitter Status	CSTS*
31	Console Storage Transmitter Data	CSDB*
32	Console Receiver Control/Status	RXCS
33	Console Receiver Data Buffer	RXDB
34	Console Transmitter Control/Status	TXCS
35	Console Transmitter Data Buffer	TXDB
55	I/O System Reset Register	IORESET

* These registers are not fully implemented, accesses yield UNPREDICTABLE results.

A.4 Global Q22–bus Address Space Map

```
Q22-bus Memory Space
--------------------

Q22-bus Memory Space  (Octal)                  0000 0000 - 1777 7777

Q22-bus I/O Space  (BBS7 Asserted)
----------------------------------

Q22-bus I/O Space (Octal)                      1776 0000 - 1777 7777
    Reserved Q22-bus I/O Space                 1776 0000 - 1776 0007
    Q22-bus Floating Address Space             1776 0010 - 1776 3777
    User Reserved Q22-bus I/O Space            1776 4000 - 1776 7777
    Reserved Q22-bus I/O Space                 1777 0000 - 1777 7477
    Interprocessor Comm Reg                    1777 7500
    Reserved Q22-bus I/O Space                 1777 7502 - 1777 7777
```

A.5 Processor Registers

Table A–1 Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Kernel Stack Pointer	KSP	0	0	RW	NVAX	1-1	
Executive Stack Pointer	ESP	1	1	RW	NVAX	1-1	
Supervisor Stack Pointer	SSP	2	2	RW	NVAX	1-1	
User Stack Pointer	USP	3	3	RW	NVAX	1-1	
Interrupt Stack Pointer	ISP	4	4	RW	NVAX	1-1	
Reserved		5–7	5			3	E1000014

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
P0 Base Register	P0BR	8	8	RW	NVAX	1-2	
P0 Length Register	P0LR	9	9	RW	NVAX	1-2	
P1 Base Register	P1BR	10	A	RW	NVAX	1-2	
P1 Length Register	P1LR	11	B	RW	NVAX	1-2	
System Base Register	SBR	12	C	RW	NVAX	1-2	
System Length Register	SLR	13	D	RW	NVAX	1-2	
CPU Identification	CPUID	14	E	RW	NVAX	2-1	
Reserved		15	F			3	E100003C
Process Control Block Base	PCBB	16	10	RW	NVAX	1-1	
System Control Block Base	SCBB	17	11	RW	NVAX	1-1	
Interrupt Priority Level[1]	IPL	18	12	RW	NVAX	1-1	
AST Level[1]	ASTLVL	19	13	RW	NVAX	1-1	
Software Interrupt Request Register	SIRR	20	14	W	NVAX	1-1	

[1] Initialized on reset

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Software Interrupt Summary Register[1]	SISR	21	15	RW	NVAX	1-1	
Reserved		22–23	16			3	E1000058
Interval Counter Control /Status	ICCS	24	18	RW	NCA	2-7	E1000060
Next Interval Count	NICR	25	19	RW	NCA	3-7	E1000064
Interval Count	ICR	26	1A	RW	NCA	3-7	E1000068
Time of Year Register	TODR	27	1B	RW	SSC	2-3	E100006C
Console Storage Receiver Status	CSRS	28	1C	RW	SSC	2-3	E1000070
Console Storage Receiver Data	CSRD	29	1D	R	SSC	2-3	E1000074
Console Storage Transmitter Status	CSTS	30	1E	RW	SSC	2-3	E1000078
Console Storage Transmitter Data	CSTD	31	1F	W	SSC	2-3	E100007C

[1]Initialized on reset

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Console Receiver Control /Status	RXCS	32	20	RW	SSC	2-3	E1000080
Console Receiver Data Buffer	RXDB	33	21	R	SSC	2-3	E1000084
Console Transmitter Control /Status	TXCS	34	22	RW	SSC	2-3	E1000088
Console Transmitter Data Buffer	TXDB	35	23	W	SSC	2-3	E100008C
Reserved		36	24			3	E1000090
Reserved		37	25			3	E1000094
Machine Check Error Register	MCESR	38	26	W	NVAX	2-1	
Reserved		39	27			3	E100009C
Reserved		40	28			3	E10000A0
Reserved		41	29			3	E10000A4
Console Saved PC	SAVPC	42	2A	R	NVAX	2-1	
Console Saved PSL	SAVPSL	43	2B	R	NVAX	2-1	
Reserved		44–54	2C			3	E10000B0
I/O System Reset Register	IORESET	55	37	W	SSC	2-3	E10000DC

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Memory Management Enable[1,2]	MAPEN	56	38	RW	NVAX	1-2	
Translation Buffer Invalidate All[2]	TBIA	57	39	W	NVAX	1-1	
Translation Buffer Invalidate Single[2]	TBIS	58	3A	W	NVAX	1-1	
Reserved		59	3B			3	E10000EC
Reserved		60	3C			3	E10000F0
System Identification	SID	62	3E	R	NVAX	1-1	
Translation Buffer Check	TBCHK	63	3F	W	NVAX	1-1	
IPL 14 Interrupt ACK[3]	IAK14	64	40	R	SSC	2-3	E1000100
IPL 15 Interrupt ACK[3]	IAK15	65	41	R	SSC	2-3	E1000104
IPL 16 Interrupt ACK[3]	IAK16	66	42	R	SSC	2-3	E1000108
IPL 17 Interrupt ACK[3]	IAK17	67	43	R	SSC	2-3	E100010C
Clear Write Buffer[3]	CWB	68	44	RW	SSC	2-3	E1000110

[1]Initialized on reset

[2]Change broadcast to vector unit if present

[3]Testability and diagnostic use only; not for software use in normal operation

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Reserved		69–99	45			3	E1000114
Reserved for VM		100	64			3	E1000190
Reserved for VM		101	65			3	E1000194
Reserved for VM		102	66			3	E1000198
Reserved		103–121	67			3	E100019C
Interrupt System Status Register	INTSYS	122	7A	RW	NVAX	2-1	
Performance Monitoring Facility Count	PMFCNT	123	7B	RW	NVAX	2-1	
Patchable Control Store Control Register	PCSCR	124	7C	WO	NVAX	2-1	
Ebox Control Register	ECR	125	7D	RW	NVAX	2-1	
Mbox TB Tag Fill[3]	MTBTAG	126	7E	W	NVAX	2-1	
Mbox TB PTE Fill[3]	MTBPTE	127	7F	W	NVAX	2-1	
Cbox Control Register	CCTL	160	A0	RW	NVAX	2-5	

[3]Testability and diagnostic use only; not for software use in normal operation

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Reserved		161	A1		NVAX	2-6	
Bcache Data ECC	BCDECC	162	A2	W	NVAX	2-5	
Bcache Error Tag Status	BCETSTS	163	A3	RW	NVAX	2-5	
Bcache Error Tag Index	BCETIDX	164	A4	R	NVAX	2-5	
Bcache Error Tag	BCETAG	165	A5	R	NVAX	2-5	
Bcache Error Data Status	BCEDSTS	166	A6	RW	NVAX	2-5	
Bcache Error Data Index	BCEDIDX	167	A7	R	NVAX	2-5	
Bcache Error ECC	BCEDECC	168	A8	R	NVAX	2-5	
Reserved		169	A9		NVAX	2-6	
Reserved		170	AA		NVAX	2-6	
Fill Error Address	CEFADR	171	AB	R	NVAX	2-5	
Fill Error Status	CEFSTS	172	AC	RW	NVAX	2-5	
Reserved		173	AD		NVAX	2-6	
NDAL Error Status	NESTS	174	AE	RW	NVAX	2-5	
Reserved		175	AF		NVAX	2-6	

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
NDAL Error Output Address	NEOADR	176	B0	R	NVAX	2-5	
Reserved		177	B1		NVAX	2-6	
NDAL Error Output Command	NEOCMD	178	B2	R	NVAX	2-5	
Reserved		179	B3		NVAX	2-6	
NDAL Error Data High	NEDATHI	180	B4	R	NVAX	2-5	
Reserved		181	B5		NVAX	2-6	
NDAL Error Data Low	NEDATLO	182	B6	R	NVAX	2-5	
Reserved		183	B7		NVAX	2-6	
NDAL Error Input Command	NEICMD	184	B8	R	NVAX	2-5	
Reserved		185–207	B9		NVAX	2-6	
VIC Memory Address Register	VMAR	208	D0	RW	NVAX	2-5	
VIC Tag Register	VTAG	209	D1	RW	NVAX	2-5	
VIC Data Register	VDATA	210	D2	RW	NVAX	2-5	

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Ibox Control and Status Register	ICSR	211	D3	RW	NVAX	2-5	
Ibox Branch Prediction Control Register[3]	BPCR	212	D4	RW	NVAX	2-5	
Reserved		213	D5		NVAX	2-6	
Ibox Backup PC[3]	BPC	214	D6	R	NVAX	2-5	
Ibox Backup PC with RLOG Unwind[3]	BPCUNW	215	D7	R	NVAX	2-5	
Reserved		216–223	D8		NVAX	2-6	
Mbox P0 Base Register[3]	MP0BR	224	E0	RW	NVAX	2-5	
Mbox P0 Length Register[3]	MP0LR	225	E1	RW	NVAX	2-5	
Mbox P1 Base Register[3]	MP1BR	226	E2	RW	NVAX	2-5	
Mbox P1 Length Register[3]	MP1LR	227	E3	RW	NVAX	2-5	
Mbox System Base Register[3]	MSBR	228	E4	RW	NVAX	2-5	

[3]Testability and diagnostic use only; not for software use in normal operation

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Mbox System Length Register[3]	MSLR	229	E5	RW	NVAX	2-5	
Mbox Memory Management Enable[3]	MMAPEN	230	E6	RW	NVAX	2-5	
Mbox Physical Address Mode	PAMODE	231	E7	RW	NVAX	2-5	
Mbox MME Address	MMEADR	232	E8	R	NVAX	2-5	
Mbox MME PTE Address	MMEPTE	233	E9	R	NVAX	2-5	
Mbox MME Status	MMESTS	234	EA	R	NVAX	2-5	
Reserved		235	EB		NVAX	2-6	
Mbox TB Parity Address	TBADR	236	EC	R	NVAX	2-5	
Mbox TB Parity Status	TBSTS	237	ED	RW	NVAX	2-5	
Reserved		238	EE		NVAX	2-6	
Reserved		239	EF		NVAX	2-6	
Reserved		240	F0		NVAX	2-6	
Reserved		241	F1		NVAX	2-6	

[3]Testability and diagnostic use only; not for software use in normal operation

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
Mbox Pcache Parity Address	PCADR	242	F2	R	NVAX	2-5	
Reserved		243	F3		NVAX	2-6	
Mbox Pcache Status	PCSTS	241	F4	RW	NVAX	2-5	
Reserved		245	F5		NVAX	2-6	
Reserved		246	F6		NVAX	2-6	
Reserved		247	F7		NVAX	2-6	
Mbox Pcache Control	PCCTL	248	F8	RW	NVAX	2-5	
Reserved		249	F9		NVAX	2-6	
Reserved		250	FA		NVAX	2-6	
Reserved		251	FB		NVAX	2-6	
Reserved		252	FC		NVAX	2-6	
Reserved		253	FD		NVAX	2-6	
Reserved		254	FE		NVAX	2-6	
Reserved		255	FF		NVAX	2-6	
Unimplemented			100–00FFFFFF			3	

(continued on next page)

Table A–1 (Cont.) Processor Registers

Register Name	Mnemonic	Number (Dec)	Number (Hex)	Type	Impl	Cat	I/O Address
See Table A–2			01000000–FFFFFFFF			2	

Type:

R = Read-only register
RW = Read-write register
W = Write-only register

Impl(emented):

NVAX = Implemented in the NVAX CPU chip
System = Implemented in the system environment
Vector = Implemented in the optional vector unit or its NDAL interface

Cat(egory), *class-subclass*, where:
class is one of:

1 = Implemented as per DEC standard 032
2 = NVAX-specific implementation which is unique or different from the DEC standard 032 implementation
3 = Not implemented internally; converted to I/O space read or write and passed to system environment

subclass is one of:

1 = Processed as appropriate by Ebox microcode
2 = Converted to Mbox IPR number and processed via internal IPR command
3 = Processed by internal IPR command, then converted to I/O space read or write and passed to system environment
4 = If virtual machine option is implemented, processed as in 1, otherwise as in 3
5 = Processed by internal IPR command
6 = May be block decoded; reference causes UNDEFINED behavior
7 = Full interval timer may be implemented in the system environment. Subset ICCS is implemented in NVAX CPU chip
8 = Converted to MFVP MSYNC

A.6 IPR Address Space Decoding

Table A–2 IPR Address Space Decoding

IPR Group	Mnemonic[2]	IPR Address Range (hex)	Contents
Normal		00000000..000000FF[1]	256 individual IPRs.
Bcache Tag	BCTAG	01000000..011FFFE0[1]	64k Bcache tag IPRs, each separated by 20(hex) from the previous one.
Bcache Deallocate	BCFLUSH	01400000..015FFFE0[1]	64k Bcache tag deallocate IPRs, each separated by 20(hex) from the previous one.
Pcache Tag	PCTAG	01800000..01801FE0[1]	256 Pcache tag IPRs, 128 for each Pcache set, each separated by 20(hex) from the previous one.
Pcache Data Parity	PCDAP	01C00000..01C01FF8[1]	1024 Pcache data parity IPRs, 512 for each Pcache set, each separated by 8(hex) from the previous one.

[1]Unused fields in the IPR addresses for these groups should be zero. Neither hardware nor microcode detects and faults on an address in which these bits are nonzero. Although noncontiguous address ranges are shown for these groups, the entire IPR address space maps into one of these groups. If these fields are nonzero, the operation of the CPU is UNDEFINED.

[2]The mnemonic is for the first IPR in the block.

Processor registers in all groups except the normal group are processed entirely by the NVAX CPU chip and will never appear on the NDAL. This is also true for a number of the IPRs in the normal group. IPRs in the normal group that are not processed by the NVAX CPU chip are converted into I/O space references and passed to the system environment via a read or write command on the NDAL.

Each of the 256 possible IPRs in the normal group are of longword length, so a 1-KB block of I/O space is required to convert each possible IPR to a unique I/O space longword. This block starts at address E1000000 (hex). Conversion of an IPR address to an I/O space address in this block is done by shifting the IPR address left into bits <9:2>, filling bits <1:0> with zeros, and merging in the base address of the block. This can be expressed by the equation:

$$IO\ ADDRESS = E1000000 + (IPR\ NUMBER * 4)$$

B

ROM Partitioning

Note

The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95, and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, will appear on the console and/or printouts from time to time.

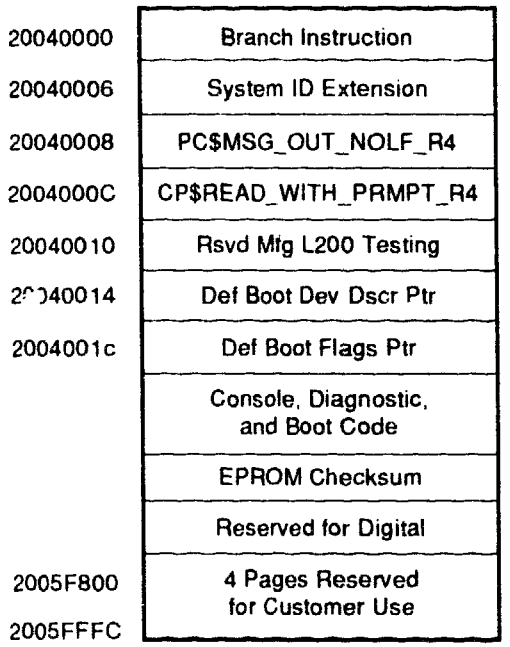
This section describes ROM partitioning and subroutine entry points that are public and are guaranteed to be compatible over future versions of the firmware. An entry point is the address at which any subroutine or subprogram will start execution.

B.1 Firmware EPROM Layout

The KA50/51/55/56 has 512 Kbytes of FEPROM. Unlike previous Q22–bus based processors, there is no duplicate decoding of the FEPROM into halt-protected and halt-unprotected spaces. The entire FEPROM is halt-protected. See Figure B–1 for the KA50/51/55/56 FEPROM layout.

ROM Partitioning
B.1 Firmware EPROM Layout

Figure B–1 KA50/51/55/56 FEPROM Layout



MLO-007698

The first instruction executed on halts is a branch around the System ID Extension (SIE) and the callback entry points. This allows these public data structures to reside in fixed locations in the FEPROM.

The callback area entry points provide a simple interface to the currently defined console for VMB and secondary bootstraps. This is documented further in the next section.

The fixed area checksum is the sum of longwords from 20040000 to the checksum, inclusive. This checksum is distinct from the checksum that the rest of the console uses.

The console, diagnostic and boot code constitute the bulk of the firmware. This code is field upgradable. The console checksum is from 20044000 to the checksum, inclusive.

The memory between the console checksum and the user area at the end of the FEPROM is reserved for Digital for future expansion of the firmware. The contents of this area is set to FF.

The last 4096 bytes of FEPROM are reserved for customer use and are not included in the console checksum. During a PROM bootstrap with PRB0 as the selected boot device, this block is tested for a PROM "signature block".

B.1.1 System Identification Registers

The firmware and operating system software reference two registers to determine the processor on which they are running. The first, the System Identification register (SID), is a NVAX internal processor register. The second, the System Identification Extension register (SIE), is a firmware register located in the FEPROM.

B.1.1.1 PR$_SID (IPR 62)

The SID longword can be read from IPR 62 using the MFPR instruction. This longword value is processor specific, however, the layout of this register is shown in Figure B–2. A description of each field is provided in Table B–1.

Figure B–2 SID : System Identification Register

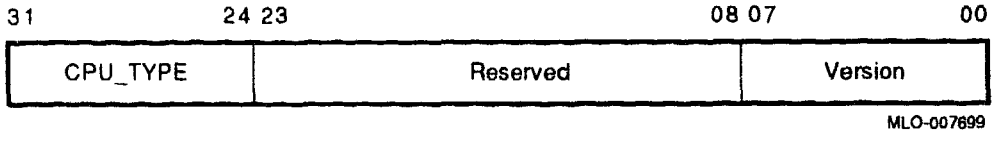


Table B–1 System Identification Register

Field	Name	RW	Description
31:24	CPU_TYPE	ro	CPU type is the processor specific identification code. *0A : CVAX* *0B : RIGEL* **13 : NVAX** *14 : SOC*
24:8	Reserved	ro	Reserved for future use.
7:0	VERSION	ro	Version of the microcode.

B.1.1.2 SIE (20040004)

The System Identification Extension register is an extension of the SID and is used to further differentiate between hardware configurations. The SID identifies which CPU and microcode are executing, and the SIE identifies which module and firmware revision are present. Note, the fields in this register are dependent on SID<31:24>(CPU_TYPE).

ROM Partitioning
B.1 Firmware EPROM Layout

By convention, all MicroVAX 3100 systems implement a longword at physical location 20040004 in the firmware FEPROM for the SIE. The layout of the SIE is shown in Figure B–3. A description of each field is provided in Table B–2.

Figure B–3 SIE : System Identification Extension (20040004)

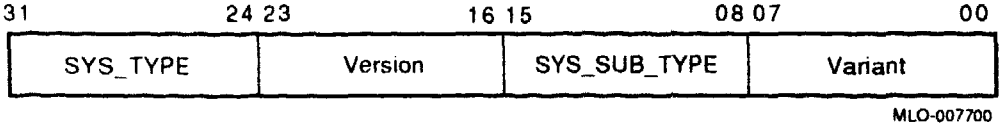


Table B–2 System Identification Extension

Field	Name	RW	Description
31:24	SYS_TYPE	ro	This field identifies the type of system for a specific processor. *03 : Bounded system.*
23:16	VERSION	ro	This field indentifies the resident version of the firmware encoded as two hexadecimal digits. For example, if the banner displays V5.0, then this field is 50 (hex).
15:8	SYS_SUB_ TYPE	ro	This field indentifies the particular system subtype. *08 : KA50/KA55* *09 : KA51/KA56* *0A : KA52* *0B : KA53*
7:0	VARIANT	ro	This field indentifies the particular system variant.

B.1.2 Call-Back Entry Points

The firmware provides several entry points that facilitate I/O to the designated console device. Users of these entry points do not need to be aware of the console device type, be it a video terminal or workstation.

The primary intent of these routines is to provide a simple console device to VMB and secondary bootstraps, before operating systems load their own terminal drivers.

These are JSB (subroutine as opposed to procedure) entry points located in fixed locations in the firmware. These locations branch to code that in turn calls the appropriate routines.

All of the entry points are designed to run at IPL 31 on the interrupt stack in physcial mode. Virtual mode is not supported. Due to internal firmware architectural restrictions, users are encouraged to only call into the halt-protected entry points. These entry points are listed in Table B–3.

Table B–3 Call-Back Entry Points

CP$GET_CHAR_R4	20040008
CP$MSG_OUT_NOLF_R4	2004000C
CP$READ_WTH_ PRMPT_R4	20040010

B.1.2.1 CP$GETCHAR_R4

This routine returns the next character entered by the operator in R0. A timeout interval can be specified. If the timeout interval is zero, no timeout is generated. If a timeout is specified and if timeout occurs, a value of 18 (CAN) is returned instead of normal input.

Registers R0,R1,R2,R3 and R4 are modified by this routine, all others are preserved.

```
;-------------------------------------------------------------
; Usage with timeout:

movl    #timeout_in_tenths_of_second,r0 ; Specify timeout.
jsb     @#CP$GET_CHAR_R4                ; Call routine.
cmpb    r0,#^x18                        ; Check for timeout.
beql    timeout_handler                 ; Branch if timeout.
; Input is in R0.

;-------------------------------------------------------------
; Usage without timeout:

clrl    r0                              ; Specify no timeout.
jsb     @#CP$GET_CHAR_R4                ; Call routine.
; Input is in R0.

;-------------------------------------------------------------
```

B.1.2.2 CP$MSG_OUT_NOLF_R4

This routine outputs a message to the console. The message is specified either by a message code or a string descriptor. The routine distinguishes between message codes and descriptors by requiring that any descriptor be located outside of the first page of memory. Hence, message codes are restricted to values between 0 and 511.

Registers R0,R1,R2,R3 and R4 are modified by this routine, all others are preserved.

```
;-------------------------------------------------------------
; Usage with message code:

movzbl  #console_message_code,r0        ; Specify message code.
jsb     @#CP$MSG_OUT_NOLF_R4            ; Call routine.

;-------------------------------------------------------------
; Usage with a message descriptor (position dependent).

movaq   5$,r0                           ; Specify address of desc.
jsb     @#CP$MSG_OUT_NOLF_R4            ; Call routine.
.
.
5$:     .ascid  /This is a message/     ; Message with descriptor.

;-------------------------------------------------------------
; Usage with a message descriptor (position independent).

pushab  5$                              ; Generate message desc.
pushl   #10$-5$                         ; on stack.
movl    sp,r0                           ; Pass desc. addr. in R0.
jsb     @#CP$MSG_OUT_NOLF_R4            ; Call routine.
clrq    (sp)+                           ; Purge desc. from stack.
.
.
5$:     .ascii  /This is a message/     ; Message.
10$:                                    ;

;-------------------------------------------------------------
```

B.1.2.3 CP$READ_WTH_PRMPT_R4

This routine outputs a prompt message and then inputs a character string from the console. When the input is accepted, DELETE, CONTROL-U and CONTROL-R functions are supported.

As with CP$MSG_OUT_NOLF_R4, either a message code or the address of a string descriptor is passed in R0 to specify the prompt string. A value of zero results in no prompt. A time-out value in 10-millisecond ticks may be passed in R1. If R1 is zero, the prompt will not timeout.

A descriptor of the input string is returned in R0 and R1. R0 contains the length of the string and R1 contains the address. This routine inputs the string into the console program string buffer and therefore the caller need not provide an input buffer. Successive calls however destroy the previous contents of the input buffer.

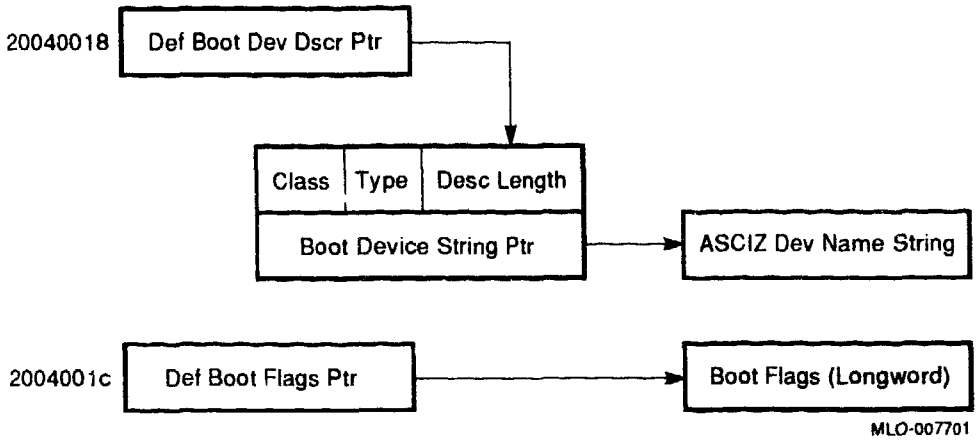
Registers R0 and R1 are modified by this routine, all others are preserved.

```
;-------------------------------------------------------------
; Usage with a message descriptor (position independent).

pushab  5$                              ; Generate prompt desc.
pushl   #10$-5$                         ; on stack.
movl    sp,r0                           ; Pass desc. addr. in R0.
clrl    r1                              ; Specify no time-out.
jsb     @#CP$READ_WTH_PRMPT_R4          ; Call routine.
clrq    (sp)+                           ; Purge prompt desc.
.                                       ; Input desc in R0 and R1.
.

5$:     .ascii  /Prompt> /              ; Prompt string.
10$:

;-------------------------------------------------------------
```

B.1.3 Boot Information Pointers

Two longwords located in FEPROM are used as pointers to the default boot device descriptor and the default boot flags (Figure B–4), because the actual location of this data may change in successive versions of the firmware. Any software that uses these pointers should reference them at the addresses in halt-protected space.

Figure B–4 Boot Information Pointers



The following macro defines the boot device descriptor format.

```
;-----------------------------------------------------------
; Default Boot Device Descriptor
;
boot_device_descriptor::
        base = .
        . = base + dsc$w_length
        .word   nvr$s_boot_device

        . = base + dsc$b_dtype
        .byte   dsc$k_dtype_z

        . = base + dsc$b_class
        .byte   dsc$k_class_z

        . = base + dsc$a_pointer
        .long   nvr_base + nvr$b_boot_device

        . = base + dsc$s_dscdef1
;-----------------------------------------------------------
```

C

Data Structures and Memory Layout

This appendix contains definitions of the key global data structures used by the CPU firmware.

Note

The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95 and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, will appear on the console and/or printouts from time to time.

C.1 Halt Dispatch State Machine

The CPU halt dispatcher determines what actions the firmware will take on halt entry based on the machine state. The dispatcher is implemented as a state machine, which uses a single bitmap control word and the transition (see Table C–1) to process all halts. The transition table is sequentially searched for matches with the current state and control word. If there is a match, a transition occurs to the next state.

The control word comprises the following information:

- **Halt Type,** used for resolving external halts. Valid only if Halt Code is 00.

  000 : power-up state
  001 : halt in progress
  010 : negation of Q22–bus DCOK
  011 : console BREAK condition detected
  100 : Q22–bus BHALT
  101 : SGEC BOOT_L asserted (trigger boot)

- **Halt Code,** compressed form of SAVPSL<13:8>(RESTART_CODE).

  00 : RESTART_CODE = 2, external halt
  01 : RESTART_CODE = 3, power-up/reset
  10 : RESTART_CODE = 6, halt instruction
  11 : RESTART_CODE = any other, error halts

- **Mailbox Action,** passed by an operating system in CPMBX<1:0>(HALT_ACTION).

  00 : restart, boot, halt
  01 : restart, halt
  10 : boot, halt
  11 : halt

- **User Action,** specified with the SET HALT console command.

  000 : default
  001 : restart, halt
  010 : boot, halt
  011 : halt
  100 : restart, boot, halt

- **HEN,** Break (halt) Enable/Disable switch, BDR<07>
- **ERR,** error status
- **TIP,** trace in progress
- **DIP,** diagnostics in progress
- **BIP,** bootstrap in progress CPMBX<2>
- **RIP,** restart in progress CPMBX<3>

A transition to a "next state" occurs if a match is found between the control word and a "current state" entry in the table. The firmware does a linear search through the table for a match. Therefore, the order of the entries in the transition table is important. The control longword is reassembled before each transition from the current machine state. The state machine transitions are shown in Table C--1.

Table C–1 Firmware State Transition Table

Current State	Next State	Halt Type	Halt Code	Mailbx Action	User Action	HEN-ERR-TIP-DIP-BIP-RIP
		Perform conditional initialization [1]				
ENTRY	->RESET INIT	xxx	01	xx	xxx	x - x - x - x - x - x
ENTRY	->BREAK INIT	011	00	xx	xxx	x - x - x - x - x - x
ENTRY	->TRACE INIT	xxx	10	xx	xxx	x - 0 - 1 - x - x - x
ENTRY	->OTHER INIT	xxx	xx	xx	xxx	x - x - x - x - x - x
		Perform common initialization [2]				
RESET INIT	->INIT	xxx	xx	xx	xxx	x - x - x - x - x - x
BREAK INIT	->INIT	xxx	xx	xx	xxx	x - x - x - x - x - x
TRACE INIT	->INIT	xxx	xx	xx	xxx	x - x - x - x - x - x
OTHER INIT	->INIT	xxx	xx	xx	xxx	x - x - x - x - x - x
		Check for external halts [3]				
INIT	->BOOTSTRAP	101	00	xx	xxx	x - x - x - x - x - x
INIT	->HALT	xxx	00	xx	xxx	x - x - x - x - x - x
		Check for pending (NEXT) trace [4]				
INIT	->TRACE	xxx	10	xx	xxx	x - x - 1 - x - x - x

[1] Perform a unique initialization routine on entry. In particular, power-ups, BREAKs, and TRACEs require special initialization. Any other halt entry performs a default initialization.

[2] After performing conditional initialization, complete common initialization.

[3] Halt on all external halts, except:

if DCOK (unlikely) and halts are disabled, bootstrap
if SGEC remote trigger, bootstrap

[4] Unconditionally enter the TRACE state, if the TIP flag is set and the halt was due to a HALT instruction. From the TRACE state the firmware exits, if TIP is set and ERR is clear; otherwise it halts.

(continued on next page)

Table C–1 (Cont.) Firmware State Transition Table

Current State	Next State	Halt Type	Halt Code	Mailbx Action	User Action	HEN-ERR-TIP-DIP-BIP-RIP
		Check for pending (NEXT) trace [4]				
TRACE	->EXIT	xxx	10	xx	xxx	x - 0 - 1 - x - x - x
TRACE	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
		Check for bootstrap conditions [5]				
INIT	->BOOTSTRAP	xxx	01	xx	xxx	0 - 0 - 0 - 0 - 0 - 0
INIT	->BOOTSTRAP	xxx	01	xx	010	1 - 0 - 0 - 0 - 0 - 0
INIT	->BOOTSTRAP	xxx	01	xx	100	1 - 0 - 0 - 0 - 0 - 0
INIT	->BOOTSTRAP	xxx	1x	10	xxx	x - 0 - 0 - 0 - 0 - 0
INIT	->BOOTSTRAP	xxx	1x	00	010	x - 0 - 0 - 0 - 0 - 0
INIT	->BOOTSTRAP	xxx	1x	00	100	x - 0 - 0 - 0 - 0 - 1
INIT	->BOOTSTRAP	xxx	1x	00	100	x - 1 - 0 - 0 - 0 - x
INIT	->BOOTSTRAP	xxx	1x	00	000	0 - 0 - 0 - 0 - 0 - 1
RESTART	->BOOTSTRAP	xxx	1x	00	000	0 - 1 - 0 - 0 - 0 - x
		Check for restart conditions [6]				
INIT	->RESTART	xxx	1x	01	xxx	x - 0 - 0 - 0 - 0 - 0
INIT	->RESTART	xxx	1x	00	001	x - 0 - 0 - 0 - 0 - 0
INIT	->RESTART	xxx	1x	00	100	x - 0 - 0 - 0 - 0 - 0

[4] Unconditionally enter the TRACE state, if the TIP flag is set and the halt was due to a HALT instruction. From the TRACE state the firmware exits, if TIP is set and ERR is clear; otherwise it halts.

[5] Bootstrap,

if power-up and halts are disabled.
if power-up and halts are enabled and user action is 2 or 4.
if not power-up and mailbox is 2.
if not power-up and mailbox is 0 and user action is 2.
if not power-up and restart failed and mailbox is 0 and user action is 0 or 4.

[6] Restart the operating system if not power-up and

if mailbox is 1.
if mailbox is 0 and user action is 1 or 4.
if mailbox is 0 and user action is 0 and halts are disabled.

(continued on next page)

Table C–1 (Cont.) Firmware State Transition Table

Current State	Next State	Halt Type	Halt Code	Mailbx Action	User Action	HEN-ERR-TIP-DIP-BIP-RIP
		Check for restart conditions [6]				
INIT	->RESTART	xxx	1x	00	000	0 - 0 - 0 - 0 - 0 - 0
		Perform common exit processing, if no errors [7]				
BOOTSTRAP	->EXIT	xxx	xx	xx	xxx	x - 0 - x - x - x - x
RESTART	->EXIT	xxx	xx	xx	xxx	x - 0 - x - x - x - x
HALT	->EXIT	xxx	xx	xx	xxx	x - 0 - x - x - x - x
		Exception transitions, just halt [8]				
INIT	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
BOOT	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
REST	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
HALT	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
TRACE	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
EXIT	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x

[6] Restart the operating system if not power-up and

if mailbox is 1.
if mailbox is 0 and user action is 1 or 4.
if mailbox is 0 and user action is 0 and halts are disabled.

[7] Exit after halts, bootstrap or restart. The exit state transitions to program I/O mode.

[8] Guard block that catches all exception conditions. In all cases, just halt.

C.2 Restart Parameter Block

VMB typically utilizes the low portion of memory unless there are bad pages in the first 128K bytes. The first page in its block is used for the Restart Parameter Block (RPB), through which it communicates to the operating system. Usually, this is page 0.

C.2 Restart Parameter Block

VMB will initialize the Restart Parameter Block (RPB) as shown in Table C–2.

Table C–2 Restart Parameter Block Fields

(R11)+	Field Name	Description
00:	RPB$L_BASE	Physical address of base of RPB.
04:	RPB$L_RESTART	Cleared.
08:	RPB$L_CHKSUM	-1
0C:	RPB$L_RSTRTFLG	Cleared.
10:	RPB$L_HALTPC	R10 on entry to VMB (HALT PC).
10:	RPB$L_HALTPSL	PR$_SAVPSL on entry to VMB (HALT PSL).
18:	RPB$L_HALTCODE	AP on entry to VMB (HALT CODE).
1C:	RPB$L_BOOTR0	R0 on entry to VMB.

Note

The field RPB$W_R0UBVEC, which overlaps the high-order word of RPB$L_BOOTR0, is set by the boot device drivers to the SCB offset (in the second page of the SCB) of the interrupt vector for the boot device.

20:	RPB$L_BOOTR1	VMB version number. The high-order word of the version is the major ID and the low-order word is the minor ID.
24:	RPB$L_BOOTR2	R2 on entry to VMB.
28:	RPB$L_BOOTR3	R3 on entry to VMB.
2C:	RPB$L_BOOTR4	R4 on entry to VMB.

Note

The 48-bit booting node address is stored in RPB$L_BOOTR3 and RPB$L_BOOTR4 for compatibility with ELN VX.X. (This field is only initialized this way when performing a network boot.)

30:	RPB$L_BOOTR5	R5 on entry to VMB.
34:	RPB$L_IOVEC	Physical address of boot driver's I/O vector of transfer addresses.

(continued on next page)

Table C–2 (Cont.) Restart Parameter Block Fields

(R11)+	Field Name	Description
38:	RPB$L_IOVECSZ	Size of BOOT QIO routine.
3C:	RPB$L_FILLBN	LBN of secondary bootstrap image.
40:	RPB$L_FILSIZ	Size of secondary bootstrap image in blocks.
44:	RPB$Q_PFNMAP	The PFN bitmap is an array of bits, where each bit has the value "1" if the corresponding page of memory is valid, or has the value "0" if the corresponding page of memory contains a memory error. Through use of the PFNMAP, the operating system can avoid memory errors by avoiding known bad pages altogether. The memory bitmap is always page-aligned, and describes all the pages of memory from physical page #0 to the high end of memory, but excluding the PFN bitmap itself and the Q–bus map registers. If the high byte of the bitmap spans some pages available to the operating system and some pages of the PFN bitmap itself, the pages corresponding to the bitmap itself will be marked as bad pages. The first longword of the PFNMAP descriptor contains the number of bytes in the PFNMAP; the second longword contains the physical address of the bitmap.
4C:	RPB$L_PFNCNT	Count of "good" pages of physical memory, but not including the pages allocated to the Q22–bus scatter/gather map, the console scratch area, and the PFN bitmap at the top of memory.
50:	RPB$L_SVASPT	0.
54:	RPB$L_CSRPHY	Physical address of CSR for boot device.
58:	RPB$L_CSRVIR	0.
5C:	RPB$L_ADPPHY	Physical address of ADP (really the address of QMRs - ^x800 to look like a UBA adapter).
60:	RPB$L_ADPVIR	0.
64:	RPB$W_UNIT	Unit number of boot device.
66:	RPB$B_DEVTYP	Device type code of boot device.
67:	RPB$B_SLAVE	Slave number of boot device.

(continued on next page)

Table C–2 (Cont.) Restart Parameter Block Fields

(R11)+	Field Name	Description
68:	RPB$T_FILE	Name of secondary bootstrap image (defaults to [SYS0.SYSEXE]SYSBOOT.EXE). This field (up to 40 bytes) is overwritten with the input string on a "solicit" boot.

Note

1. For VAX/OpenVMS, the RPB$T_FILE must contain the root directory string "SYSn." on a non-network bootstrap. This string is parsed by SYSBOOT (SYSBOOT does not use the high nibble of BOOTR5).
2. The RPB$T_FILE is overwritten to contain the boot node name for compatibility with ELN VX.X (this field is only initialized this way when performing a network boot).

90:	RPB$B_CONFREG	Array (16 bytes) of adapter types (NDT$_UB0 - UNIBUS).
A0:	RPB$B_HDRPGCNT	Count of header pages.
A1:	RPB$W_BOOTNDT	Boot adapter nexus device type. Used by SYSBOOT and INIADP (OF SYSLOA) to configure the adapter of the boot device (changed from a byte to a word field in Version 12 of VMB).
B0:	RPB$L_SCBB	Physical address of SCB.
BC:	RPB$L_MEMDSC	Count of pages in physical memory including both good and bad pages. The high 8 bits of this longword contain the TR #, which is always 0 for KA52.
C0:	RPB$L_MEMDSC+4	PFN of the first page of memory. This field is always 0 for KA50/51/55/56, even if page #0 is a bad page.

Note

No other memory descriptors are used.

104:	RPB$L_BADPGS	Count of "bad" pages of physical memory.

(continued on next page)

Table C–2 (Cont.) Restart Parameter Block Fields

(R11)+	Field Name	Description
108:	RPB$B_CTRLLTR	Boot device controller number biased by 1. In VAX/OpenVMS, this field is used by INIT (in SYS) to construct the boot device's controller letter. A 0 implies this field has not been initialized, else if initialized, A=1, B=2, etc. (this field was added in Version 13 of VMB).
nn:		The rest of the RPB is zeroed.

C.3 VMB Argument List

The VMB code will also initialize an argument list as shown in Table C–3 (the address of the argument list is passed in the AP).

Table C–3 VMB Argument List

(AP)+	Field Name	Description
04:	VMB$L_FILECACHE	Quadword filename.
0C:	VMB$L_LO_PFN	PFN of first page of physical memory (always 0, regardless of where 128 Kbytes of "good" memory starts).
10:	VMB$L_HI_PFN	PFN of last page of physical memory.
14:	VMB$Q_PFNMAP	Descriptor of PFN bitmap. First longword contains count of bytes in bitmap. Second longword contains physical address of bitmap. (Same rules as for RPB$Q_PFNMAP listed above.)
1C:	VMB$Q_UCODE	Quadword.
24:	VMB$B_SYSTEMID	48-bit (actually a quadword is allocated) booting node address which is initialized when performing a network boot. This field is copied from the Target System Address parameter of the parameters message. (The DECnet HIORD value is added if the field was two bytes.)
2C:	VMB$L_FLAGS	Set as needed.
30:	VMB$L_CI_HIPFN	Cluster interface high PFN.

(continued on next page)

Table C–3 (Cont.) VMB Argument List

(AP)+	Field Name	Description
34:	VMB$Q_NODENAME	Boot node name which is initialized when performing a network boot. This field is copied from the Target System Name parameter of the parameters message.
3C:	VMB$Q_HOSTADDR	Host node address (this value is only initialized when booting over the network). This field is copied from the Host System Address parameter of the parameters message.
44:	VMB$Q_HOSTNAME	Host node name (this value is only initialized when performing a network boot). This field is copied from the Host System Name parameter of the parameters message.
4C:	VMB$Q_TOD	Time of day (this value is only initialized when performing a network boot). The time of day is copied from the first eight bytes of the Host System Time parameter of the parameters message. (The time differential values are NOT copied.)
54:	VMB$L_XPARAM	Pointer to data retrieved from request of the parameter file.
58:		The rest of the argument list is zeroed.

D

Configurable Machine State

The KA50/51/55/56 CPU modules have many control registers that need to be configured for proper operation of the module. The following list shows the normal state of all configurable bits in the CPU module as they are left after the successful completion of power-up ROM diagnostics.

Note

The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95 and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, will appear on the console and/or printouts from time to time.

```
Configuration Register Bit Settings(* = reset state)

NCA:
-----
NCA_CSR1:       Mode Control and Diagnostic Status Register  (2102 0004)
                15:14:  CP2 MT Timer Prescaler
                        11 = 144000 cycles*  - needed for CQBIC 10ms No Grant
                                timeout

                13:12:  CP1 MT Timer Prescaler
                        00 = 144 cycles - minimum for passive releases, no
                                cycle should take longer than this.

                11:10:  NDAL Timeout Prescaler
                        00 = 3200 cycles* - this is longer than both NCA and
                                NMC transactions timeouts, preserves timeout
                                order.

                9:      CQBIC mode
                        0 =  CQBIC not present* - this is to avoid the
                                QBUS_TRANS deadlock.
```

```
                8:      IO2 ID enable
                        1 = enabled
                7:      Force wrong CP2 bus parity - off - diagnostic use only
                6:      Force wrong CP1 bus parity - off - diagnostic use only
                5:      Force wrong NDAL master parity - off - diagnostic use
                          only
                4:      Force wrong NDAL slave parity - off - diagnostic use
                          only
                3:      Enable prefetch
                        1 = enable CP bus prefetch on DMA reads
                2:      Force write buffer hit - off - diagnostic use only
                1:      Force CP2 bus owner - diagnostic use only
                        0 = disabled
                0:      Force CP1 bus owner - diagnostic use only
                        0 = disabled

ICCS:           Interval Clock Control and Status Register  (2100 0060)

                NOTE: OpenVMS sets ICCS, NICR to proper values.

                6:      Interrupt enable
                        0 = disabled*
                5:      Single step - off
                4:      Transfer
                        0 = disabled*
                0:      Run - increment every 1usec - off

NICR:           Next Interval Count Register  (2100 0064)
                31:0    Initial count value for ICR (FFFFD8F0* (10ms))

NMC:
-----
NMC_CSR0-7:     Memory Configuration Registers  (2101 8000 thru 2101 801C)

                NOTE: Diagnostics set these registers based on available memory

                31:     Base Address Valid
                        0 = not valid*
                        1 = valid
                28:24:  Base Address (0 on reset)
                        1MB RAM - all address bits used
                        4MB RAM - only <28:26> used
                2:1     RAM size
                        01 = 1MB RAM
                        10 = 4MB RAM
                        11 = non-existent bank
                0:      Mode
                        1 = 64-bit mode

NMC_CSR18:      Mode Control and Diagnostic Status Register  (2101 8048)
                31:     Fast Diagnostic Mode (FDM)
                        0 = disabled* - diagnostic use only

                30:     FDM Second pass
                        0 = disabled* - diagnostic use only
```

```
                29:     Diagnostic Checkbit mode
                        0 = disabled* - diagnostic use only

                28:     QBus on IO1
                        0 = QBus on IO2*

                27:     Enable soft error log (NDAL & memory related)
                        0 = disabled* - OpenVMS enables this

                26:     Flush BCache
                        0 = don't flush*

                24:17:  Memory diagnostic check bits (0*) - may not be read
                                                            as 0

                8:7:    NDAL Timeout Scaler
                        00 = 2600 cycles* - maximum to preserve timeout order

                6:      Disable memory error
                        0 = memory errors deteted and corrected*

                5:      Refresh interval timer select
                        0 = 328 cycles*

                4:2:    Force wrong parity on NDAL transactions - off
                              - diagnostic use only

                1:      Disable memory refresh
                        0 = memory refreshed*

                0:      Force refresh
                        0 = normal refresh*

NMC_CSR19:      6-bit Address and Mode Register  (2101 804C)
                16:     Ignore 6-bit mode
                        0 = 6-bits checked*
                15:     Disable 6-bit error
                        0 = 6-bit errors detected*
                14:6:   6-bit segment address (0*) - not used in normal
                                  operation
                5:3:    6-bit mask (0*) - not used in normal operation
                2:0:    6-bit operation mode
                        X00 = reconstruction mode* - not used in normal
                                  operation

NMC_OSCR:       6-bit Data Registers  (2101 0000 thru 2101 7FFF)
                23:12:  6-bit field 1 (0 at reset)
                11:0:   6-bit field 0 (0 at reset)

NVAX:
-----
CPUID:          CPU ID Register  (IPR E)
                7:0:    CPU identifcation = 0 (for single processor config.)

SID:            System Identification Register  (IPR 3E)
                NOTE: this register may only be written by microcode
```

Configurable Machine State

```
                31:24:  CPU type - 13hex (NVAX code)
                13:8:   Patch revision
                7:0:    Microcode revision

ICSR:           IBox Control and Status Register (IPR D3)
                0:      VIC enable
                        1 = enabled

ECR:            EBox Control Register  (IPR 7D)
                13:     FBox test enable
                        0 = disable* - diagnostic use only

                7:      Interval time mode
                        1 = full CPU implemented interval timer

                5:      S3 stall timeout
                        0 = counts cycles w/ timeout_enable asserted (~3 sec)*

                3:      FBox stage 4 bypass
                        1 = enabled - improves FBox latency

                2:      S3 external time base timeout
                        0 = disabled* - use internal time base

                1:      FBox enable
                        1 = enabled

                0:      Vector present
                        0 = no* - no vector option available at this time

MMAPEN:         Memory Map Enable Register  (IPR E6)
                0:      Memory map enable
                        0 = disabled* - OpenVMS enables this

PAMODE:         Physical Address Mode Register  (IPR E7)
                0:      Physical address mode
                        0 = 30-bit physical address space*

PCCTL:          PCache Control Register  (IPR F8)
                8:      PCache Electrical disable
                        0 = PCache enabled*

                7:5     MBox performance monitor mode (0*) - diagnostic use
                        only

                4:      PCache error enable
                        1 = enables PCache error detection

                3:      Bank select during force hit mode
                        0 = left bank selected if force hit mode enabled*
                                - diagnostic use only

                2:      Force hit
                        0 = disabled* - diagnostic use only

                1:      I_enable
                        1 = enable PCache for IREAD, INVAL, I_CF commands
```

```
                0:      D_enable
                        1 = enable PCache for INVAL, D-stream read/write/fill

CCTL:           CBox Control Register  (IPR A0)
                30:     Software ETM
                        0 = disabled* - diagnostic use only

                16:     Force NDAL parity error - off - diagnostic use only

                15:11:  Performance monitoring bits (0*) - diagnostic use only

                10:     Disable CBox write packer
                        0 = write packer enabled* - improves write latency

                9:      Read timeout time base
                        0 = external time base

                8:      Software ECC
                        0 = use correct ECC*

                7:      Disable BCache errors
                        0 = BCache errors detected*

                6:      Force Hit
                        0 = disabled* - diagnostic use only

                5:4:    BCache size
                        00 = 128 KB* (KA50/52/55)
                        10 = 512 KB  (KA51/53/54/56)

                3:2:    Data store speed
                        00 = 2 cycle read, 3 cycle write* (KA51/53/54/56)
                        01 = 3 cycle read, 4 cycle write  (KA50/52)
                        10 = 4 cycle read, 5 cycle write  (KA55)
                1:      Tag store speed
                        0 = 3 cycle read, 3 cycle write* (KA51/53/54/56)
                        1 = 4 cycle read, 4 cycle write  (KA50/52/55)

                0:      Enable BCache
                        1 = enabled

CQBIC:
------
SCR:            System Configuration Register  (2008 0000)
                14:     Halt enable
                        1 = BHALT to CQBIC HALTIN pin to cause halts

                12:     Page prefetch disable
                        1 = map prefetch disabled - historical latency reasons

                7:      Restart enable
                        0 = QBus restart causes ARB power-up reset*

                3:1:    ICR offset address select bits
                        0 = (AUX mode not supported)*
```

Configurable Machine State

```
ICR:            Interprocessor Communication Register  (2000 1F40)
                8:      AUX Halt
                        0 = no halt - AUX mode not supported

                6:      ICR interrupt enable
                        0 = interprocessor interrupts disabled - only
                                uniprocessor config. allowed

                5:      Local memory external access enable
                        0 = external access disabled* - OpenVMS configures map

QBMBR:          Q-Bus Map Base Address Register (2008 0010)
                28:15:  address where 8K QBus mapping register are located
                        (undefined at reset)

SHAC:
-----
NOTE:  all SHAC registers are set up by OpenVMS driver

PQBBR:          Port Queue Block Base Register  (2000 4248)
                20:0:   upper bits of physical address of base of Port Queue
                        block.  Contains HW version, FW version, shared host
                        memory version and CI port maintenance ID at power-up.

PPR:            Port Parameter Register  (2000 4258)
                31:29:  Cluster size.  For SHAC value = 0.
                28:16:  Internal buffer length = 0* (For SHAC value = 1010 hex)
                7:0:    Port number.  Same as SHAC's DSSI ID.

PMCSR:          Port Maintenance Control and Status Register  (2000 425C)
                2:      Interrupt enable
                        0 = disabled*
                1:      Maintenance timer disable
                        0 = enabled*

SGEC:
-----
NOTE:  all SGEC registers are set up by OpenVMS driver

NICSR0:         Vector Address, IPL, Synch/Asynch Register  (2000 8000)
                31:30:  Interrupt priority
                        00 = 14*
                29:     Synch/Asynch bus master operating mode
                        0 = asynchronous*
                15:0:   Interrupt vector = 0003hex*
```

```
NICSR6:     Command and Mode Register  (2000 8018)
            30:     Interrupt enable
                    0 = disabled*
            28:25:  Burst limit mode
                    maximum number of longwords transferred in a single
                    DMA burst.  1*,2,4,8 when NICSR<19>is clear; 1*,4
                    when set.
            20:     Boot message enable mode
                    0 = disabled*
            19:     Single cycle enable mode
                    0 = disabled*
            11:     Start/Stop transmission command
                    0 = SGEC transmission process in stopped state*
            10:     Start/Stop reception command
                    0 = SGEC reception process in stopped state*
            9:8:    Operating mode
                    00 = normal mode*
            7:      Disable data chaining mode
                    0 = frames too long for current receive buffer
                        will be transferred to the next buffer(s)
                        in receive list*
            6:      Force collision mode (internal loopback mode only)
                    0 = no collision*
            3:      Pass bad frames mode
                    0 = bad frames discarded*
            2:1:    Address filtering mode
                    00 = normal mode*

NICSR7:     System Base Register  (2000 801C)
            29:0:   System base address - physical starting address of
                    the VAX system page table (unpredictable after reset)

NICSR9:     Watchdog Timers Register  (2000 8024)
            31:16:  Receive watchdog timeout
                    0 = never timeout*
                    default = 1250 = 2 ms
                    range = 72 µs (45) to 100 ms
            15:0:   Transmit watchdog timeout
                    0 = never timeout*
                    default = 1250 = 2 ms
                    range = 72 µs (45) to 100 ms

SSC:
----
SSCBAR:     SSC Base Address Register  (2014 0000)
            29:0    Base Address (reset value = 20140000)

SSCCR:      SSC Configuration Register  (2014 0010)
            27:     Interrupt vector disable
                    0 = interrupt vector enabled*
```

Configurable Machine State

```
                25:24:  IPL Level
                        00 = 14*

                23:     ROM access time
                        0 = 350 ns*

                22:20:  ROM size
                        110 = 512KB

                18:16:  Halt protected space
                        110 = 20040000 - 200BFFFF (historical)

                15:     n/a

                14:12:  n/a

                6:      Programmable address strobe 1 ready enable (for BDR)
                        1 = ready asserted after address strobe

                5:4:    Programmable address strobe 1 enable (for BDR)
                        11 = read enabled, write enabled

                2:      Programmable address strobe 0 ready enable
                        0 = no ready after address strobe*  Used for FEPROM
  programming

                1:0:    Programmable address strobe 0 enable
                        00 = read disabled, write disabled* Used for FEPROM
  programming
SSCBT:          SSC Bus Time Out Register  (2014 0020)
                23:0:   Bus timeout interval = 4000hex (16.384 ms)
                        range = 1 to FFFFFF (1 µs to 16.77 sec)

ADS0MAT:        Programmable Address Strobe 0 Match Register  (2014 0130)
                29:2:   Match address
                        0 = disabled*

ADS0MAS:        Programmable Address Strobe 0 Mask Register  (2014 0134)
                29:2:   Mask address bits

ADS1MAT:        Programmable Address Strobe 1 Match Register  (2014 0140)
                29:2:   Match address = 20084000 (for BDR)

ADS1MAS:        Programmable Address Strobe 1 Mask Register  (2014 0144)
                29:2:   Mask address bits = 7C (for BDR)

T1CR:           Programmable Timer 0 Control Register  (2014 0100)
                6:      Interrupt enable
                        0 = disabled*
                2:      STP
                        0 = run after overflow*
                0:      RUN
                        0 = counter not running*  (historical)
```

```
T1CR:       Programmable Timer 1 Control Register  (2014 0110)
            6:      Interrupt enable
                    0 = disabled*
            2:      STP
                    0 = run after overflow*
            0:      RUN
                    1 = counter incrementing every microsecond (historical)

TNIR:       Programmable Timer Next Interval Registers  (2014 0108,
                                                         2014 0118)
            31:0:   Timer next interval count (use 2's complement)
                    range = 0* to 1.2 hours

T0IV:       Programmable Timer 0 Interrupt Vector Register  (2014 010C)
            9:2:    Timer interrupt vector = 78hex

T1IV:       Programmable Timer 1 Interrupt Vector Registers (2014 011C)
            9:2:    Timer interrupt vector = 7Chex

TOY:        Time of Year Register  (2014 006C)
            31:0:   Number of 10 ms intervals since written

DLEDR:      Diagnostic LED Register  (2014 0030)
            3:0:    Display bits
                    0 = LEDs on*  (historical)
```

E

NVRAM Partitioning

This appendix describes how the CPU firmware partitions the SSC 1 KB battery-backed-up (BBU) RAM.

Note

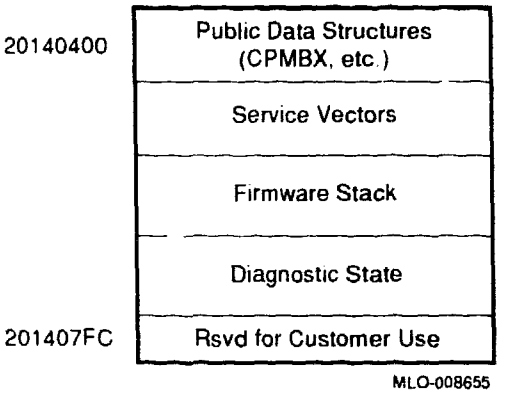
The firmware and diagnostics for MicroVAX 3100 Models 85, 90, 95 and 96 were written to support other systems as well. References to features and functions not available on these models, such as Q–bus and DSSI, will appear on the console and/or printouts from time to time.

E.1 SSC RAM Layout

The KA50/51/55/56 firmware uses the 1K byte of NVRAM on the SSC (see Figure E–1), for storage of firmware specific data structures and other information that must be preserved across power cycles. This NVRAM resides in the SSC chip starting at address 20140400. The NVRAM should not be used by the operating systems except as documented below. This NVRAM is not reflected in the bitmap built by the firmware.

NVRAM Partitioning
E.1 SSC RAM Layout

Figure E–1 KA50/51/55/56 SSC NVRAM Layout



E.1.1 Public Data Structures

Public data structures consist of three bytes, NVR0, NVR1, and NVR2. Their functions are described in Table E–1, Table E–2, and Table E–3.

E.1.1.1 Console Program MailBoX (CPMBX)

The Console Program MailBoX (CPMBX) comprised of NVR0, is a software data structure located at the beginning of NVRAM (20140400). The CPMBX is used to pass information between the CPU firmware and diagnostics, VMB, or an operating system.

Figure E–2 NVR0 (20140400) : Console Program MailBoX (CPMBX)

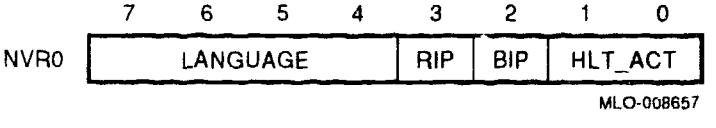


Table E–1 Bit Functions for NVR0

Field	Name	Description
7:4	LANGUAGE	This field specifies the current selected language for displaying halt and error messages on terminals which support MCS.
3	RIP	If set, a restart attempt is in progress. This flag must be cleared by the operating system, if the restart succeeds.

(continued on next page)

Table E–1 (Cont.) Bit Functions for NVR0

Field	Name	Description
2	BIP	If set, a bootstrap attempt is in progress. This flag must be cleared by the operating system if the bootstrap succeeds.
1:0	HLT_ACT	Processor halt action - this field in conjunction with the conditions specified for system halts is used to control the automatic restart/bootstrap procedure. HLT_ACT is normally written by the operating system. 0 : Restart; if that fails, reboot; if that fails, halt. 1 : Restart; if that fails, halt. 2 : Reboot; if that fails, halt. 3 : Halt.

E.1.1.2 Terminal Status

Figure E–3 NVR1 (20140401)

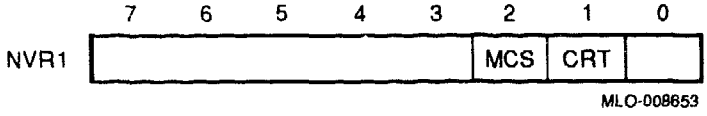


Table E–2 Bit Functions for NVR1

Field	Name	Description
2	MCS	If set, indicates that the attached terminal supports Multinational Character Set. If clear, MCS is not supported.
1	CRT	If set, indicates that the attached terminal is a CRT. If clear, indicates that the terminal is hardcopy.

E.1.1.3 Keyboard Status

Figure E–4 NVR2 (20140402)

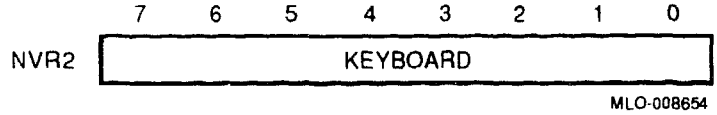


Table E–3 Bit Functions for NVR2

Field	Name	Description
7:0	KEYBOARD	This field indicates the national keyboard variant in use.

E.1.2 Service Vectors

Service vectors point to the routines for the reading or writing of characters by the console.

E.1.3 Firmware Stack

This section contains the stack that is used by all of the firmware, with the exception of VMB, which has its own built-in stack.

E.1.4 Diagnostic State

This area is used by the firmware resident diagnostics. It serves as the primary communications mechanism between the diagnostics and the console program.

E.1.5 USER Area

The KA50/51/55/56 console reserves the last longword (address 201407FC) of the NVRAM for customer use. This location is not tested by the console firmware. Its value is undefined.

F

MOP Counters

The following counters are kept for the Ethernet boot channel. All counters are unsigned integers. V4 counters rollover on overflow. All V3 counters "latch" at their maximum value to indicate overflow. Unless otherwise stated, all counters include both normal and multicast traffic. Furthermore, they include information for all protocol types. Frames received and bytes received counters do not include frames received with errors. Table F–1 displays the byte lengths and ordering of all the counters in both MOP Versions 3.0 and 4.0.

Table F–1 MOP Counter Block

	V3		V4		
Name	**Off**	**Len**	**Off**	**Len**	**Description**
TIME_SINCE_CREATION	00	2	00	16	**Time since last zeroed.** The time which has elapsed, since the counters were last zeroed. Provides a frame of reference for the other counters by indicating the amount of time they cover. For MOP V3, this time is the number of seconds. MOP V4 uses the UTC Binary Relative Time format.

(continued on next page)

Table F–1 (Cont.) MOP Counter Block

Name	V3 Off	V3 Len	V4 Off	V4 Len	Description
Rx_BYTES	02	4	10	8	**Bytes received.** The total number of user data bytes successfully received. This does not include Ethernet data link headers. This number is the number of bytes in the Ethernet data field, which includes any padding or length fields when they are enabled. These are bytes from frames that passed hardware filtering. Wnen the number of frames received is used to calculate protocol overhead, the overhead plus bytes received provides a measurement of the amount of Ethernet bandwidth (over time) consumed by frames addressed to the local system.
Tx_BYTES	06	4	18	8	**Bytes sent.** The total number of user data bytes successfully transmitted. This does not include Ethernet data link headers or data link generated retransmissions. This number is thc number of bytes in the Ethernet data field, which includes any padding or length fields when they are enabled. When the number of frames sent is used to calculate protocol overhead, the overhead plus bytes sent provides a measurement of the amount of Ethernet bandwidth (over time) consumed by frames sent by the local system.

(continued on next page)

Table F–1 (Cont.) MOP Counter Block

Name	V3 Off	V3 Len	V4 Off	V4 Len	Description
Rx_FRAMES	0A	4	20	8	**Frames received.** The total number of frames successfully received. These are frames that passed hardware filtering. Provides a gross measurement of incoming Ethernet usage by the local system. Provides information used to determine the ratio of the error counters to successful transmits.
Tx_FRAMES	0E	4	28	8	**Frames sent.** The total number of frames successfully transmitted. This does not include data link generated retransmissions. Provides a gross measurement of outgoing Ethernet usage by the local system. Provides information used to determine the ratio of the error counters to successful transmits.
Rx_MCAST_BYTES	12	4	30	8	**Multicast bytes received.** The total number of multicast data bytes successfully received. This does not include Ethernet data link headers. This number is the number of bytes in the Ethernet data field. In conjunction with total bytes received, provides a measurement of the percentage of this system's receive bandwidth (over time) that was consumed by multicast frames addressed to the local system.
Rx_MCAST_FRAMES	16	4	38	8	**Multicast frames received.** The total number of multicast frames successfully received. In conjunction with total frames received, provides a gross percentage of the Ethernet usage for multicast frames addressed to this system.

(continued on next page)

Table F–1 (Cont.) MOP Counter Block

Name	V3 Off	V3 Len	V4 Off	V4 Len	Description
Tx_INIT_DEFFERED	1A	4	40	8	**Frames sent[1], initially deferred.** The total number of times that a frame transmission was deferred on its first transmission attempt. In conjunction with total frames sent, measures Ethernet contention with no collisions.
Tx_ONE_COLLISION	1E	4	48	8	**Frames sent [1], single collision.** The total number of times that a frame was successfully transmitted on the second attempt after a normal collision on the first attempt. In conjunction with total frames sent, measures Ethernet contention at a level where there are collisions but the backoff algorithm still operates efficiently.
Tx_MULTI_COLLISION	22	4	50	8	**Frames sent[1], multiple collisions.** The total number of times that a frame was successfully transmitted on the third or later attempt after normal collisions on previous attempts. In conjunction with total frames sent, measures Ethernet contention at a level where there are collisions and the backoff algorithm no longer operates efficiently. NO SINGLE FRAME IS COUNTED IN MORE THAN ONE OF THE ABOVE THREE COUNTERS.

[1]Only one of these three counters will be incremented for a given frame.

(continued on next page)

Table F–1 (Cont.) MOP Counter Block

Name	V3 Off	V3 Len	V4 Off	V4 Len	Description
TxFAIL_COUNT	26	2	-	-	**Send failure count[2].** The total number of times a transmit attempt failed. Each time the counter is incremented, a type of failure is recorded. When Read-counter function reads the counter, the list of failures is also read. When the counter is set to zero, the list of failures is cleared. In conjunction with total frames sent, provides a measure of significant transmit problems. TxFAIL_BITMAP contains the possible reasons.
TxFAIL_BITMAP	2C	2	-	-	**Send failure reason bitmap[2].** This bitmap lists the types of transmit failures that occurred as summarized below: 0 - Excessive collisions 1 - Carrier detect failed 2 - Short circuit 3 - Open circuit 4 - Frame too long 5 - Remote failure to defer
TxFAIL_EXCESS_COLLS	-	-	58	8	**Send failure—Excessive collisions.** Exceeded the maximum number of retransmissions due to collisions. Indicates an overload condition on the Ethernet.

[2]V3 send/receive failures are collapsed into one counter with bitmap indicating which failures occurred.

(continued on next page)

Table F–1 (Cont.) MOP Counter Block

Name	V3 Off	V3 Len	V4 Off	V4 Len	Description
TxFAIL_CARIER_CHECK	-	-	60	8	**Send failure—Carrier check failed.** The data link did not sense the receive signal that is required to accompany the transmission of a frame. Indicates a failure in either the transmitting or receiving hardware. Could be caused by either transceiver, transceiver cable, or a babbling controller that has been cut off.
TxFAIL_SHRT_CIRCUIT	-	-	68	8	**Send failure—Short circuit[3].** There is a short somewhere in the local area network coaxial cable or the transceiver or controller /transceiver cable has failed. This indicates a problem either in local hardware or global network. The two can be distinguished by checking to see if other systems are reporting the same problem.
TxFAIL_OPEN_CIRCUIT	-	-	70	8	**Send failure—Open circuit[3].** There is a break somewhere in the local area network coaxial cable. This indicates a problem either in local hardware or global network. The two can be distinguished by checking to see if other systems are reporting the same problem.
TxFAIL_LONG_FRAME	-	-	78	8	**Send failure—Frame too long[3].** The controller or transceiver cut off transmission at the maximum size. This indicates a problem with the local system. Either it tried to send a frame that was too long or the hardware cutoff transmission too soon.

[3]Always zero.

(continued on next page)

Table F–1 (Cont.) MOP Counter Block

Name	V3 Off	V3 Len	V4 Off	V4 Len	Description
TxFAIL_REMOTE_DEFER	-	-	80	8	**Send failure—Remote failure to defer[3].** A remote system began transmitting after the allowed window for collisions. This indicates either a problem with some other system's carrier sense or a weak transmitter.
RxFAIL_COUNT	2A	2	-	-	**Receive failure count[2].** The total number of frames received with some data error. Includes only data frames that passed either physical or multicast address comparison. This counter includes failure reasons in the same way as the send failure counter. In conjunction with total frames received, provides a measure of data related receive problems. RxFAIL_BITMAP contains the possible reasons.
RxFAIL_BITMAP	2C	2	-	-	**Receive failure reason bitmap[2].** This bitmap lists the types of receive failures that occurred as summarized below: 0 - Block check failure 1 - Framing error 2 - Frame too long
RxFAIL_BLOCK_CHECK	-	-	88	8	**Receive failure—Block check error.** A frame failed the CRC check. This indicates several possible failures, such as EMI, late collisions, or improperly set hardware parameters.

[2]V3 send/receive failures are collapsed into one counter with bitmap indicating which failures occurred.

[3]Always zero.

(continued on next page)

Table F–1 (Cont.) MOP Counter Block

Name	V3 Off	V3 Len	V4 Off	V4 Len	Description
RxFAIL_FRAMING_ERR	-	-	90	8	**Receive failure—Framing error.** The frame did not contain an integral number of 8 bit bytes. This indicates several possible failures, such as EMI, late collisions, or improperly set hardware parameters.
RxFAIL_LONG_FRAME	-	-	98	8	**Receive failure—Frame too long**[3]. The frame was discarded because it was outside the Ethernet maximum length and could not be received. This indicates that a remote system is sending invalid length frames.
UNKNOWN_DESTINATION	2E	2	A0	8	**Unrecognized frame destination.** The number of times a frame was discarded because there was no portal with the protocol type or multicast address enabled. This includes frames received for the physical address, the broadcast address, or a multicast address.
DATA_OVERRUN	30	2	A8	8	**Data overrun.** The total number of times the hardware lost an incoming frame because it was unable to keep up with the data rate. In conjunction with total frames received, provides a measure of hardware resource failures. The problem reflected in this counter is also captured as an event.

[3]Always zero.

(continued on next page)

Table F–1 (Cont.) MOP Counter Block

Name	V3 Off	V3 Len	V4 Off	V4 Len	Description
NO_SYSTEM_BUFFER	32	2	B0	8	**System buffer unavailable[3].** The total number of times no system buffer was available for an incoming frame. In conjunction with total frames received, provides a measure of system buffer related receive problems. The problem reflected in this counter is also captured as an event. This can be any buffer between the hardware and the user buffers (those supplied on Receive requests). Further information as to potential different buffer pools is implementation specific.
NO_USER_BUFFER	34	2	B8	8	**User buffer unavailable[3].** The total number of times no user buffer was available for an incoming frame that passed all filtering. These are the buffers supplied by users on Receive requests. In conjunction with total frames received, provides a measure of user buffer related receive problems. The problem reflected in this counter is also captured as an event.
FAIL_COLLIS_DETECT	-	-	C0	8	**Collision detect check failure.** The approximate number of times that collision detect was not sensed after a transmission. If this counter contains a number roughly equal to the number of frames sent, either the collision detect circuitry is not working correctly or the test signal is not implemented.

[3]Always zero.

G

Error Messages

The error messages issued by the KA50/51/55/56 firmware fall into three categories: halt code messages, VMB error messages, and console messages.

G.1 Machine Check Register Dump

Some error conditions, such as machine check, generate an error summary register dump preceding the error message. For example, examining a nonexistent memory location results in the following display:

```
 >>e/p/l 20000000

MESR=00006000     MEAR=08406010     MMCDSR=01111100   MOAMR=00000000
CESR=80000200     CMCDSR=0000C108   CSEAR1=00000000   CSEAR2=00000000
CIOEAR1=00000000  CIOEAR2=00000000  CNEAR=00000000    ICSR=00000001
PCSTS=FFFFF800    PCADR=FFFFFFF8    TBSTS=C00000E0    TBADR=F5755754
NESTS=0000000C    NEOADR=E014066C   NEOCMD=8000F005   NEICMD=000003FF
NEDATHI=FFFFFFFF  NEDATLO=FF7F9FFF  CEFSTS=0001920A   CEFADR=E0000000
BCETSTS=000003E0  BCETIDX=FFFFFFE0  BCETAG=FFFFFE00   BCEDSTS=00000F00
BCEDIDX=001FFFF8  BCEDECC=00000000  CBTCR=00004000    DSER=00000080
QBEAR=00000000    DEAR=00000000     IPCR0=0000        ECR=0000008A
SCSICSR4=00       SCSICSR6=C0       SCSICSR5=00
?7D MACHINE CHECK  80060000 00000000 20048C68 20048C59 20048C55 40110080

>>>
```

G.2 Halt Code Messages

Except on power-up, which is not treated as an error condition, the following halt messages are issued by the firmware whenever the processor halts (Table G–1).

For example, if the processor encounters a HALT instruction while in kernel mode, the processor halts and the firmware displays the following before entering console I/O mode:

```
?06 HLT INST
PC = 800050D3
```

G.2 Halt Code Messages

The number preceding the halt message is the "halt code." This number is obtained from SAVPSL<13:8>(RESTART_CODE), IPR 43, which is saved on any processor restart operation.

Table G–1 HALT Messages

Code	Message	Description
?02	EXT HLT	External halt, caused by either console BREAK condition, Q22-bus BHALT_L, or DBR<AUX_HLT> bit was set while enabled.
_03	—	Power-up, no halt message is displayed. However, the presence of the firmware banner and diagnostic countdown indicates this halt reason.
?04	ISP ERR	In attempting to push state onto the interrupt stack during an interrupt or exception, the processor discovered that the interrupt stack was mapped NO ACCESS or NOT VALID.
?05	DBL ERR	The processor attempted to report a machine check to the operating system, and a second machine check occurred.
?06	HLT INST	The processor executed a HALT instruction in kernel mode.
?07	SCB ERR3	The SCB vector had bits <1:0> equal to 3.
?08	SCB ERR2	The SCB vector had bits <1:0> equal to 2.
?0A	CHM FR ISTK	A change mode instruction was executed when PSL<IS> was set.
?0B	CHM TO ISTK	The SCB vector for a change mode had bit <0> set.
?0C	SCB RD ERR	A hard memory error occurred while the processor was trying to read an exception or interrupt vector.
?10	MCHK AV	An access violation or an invalid translation occurred during machine check exception processing.
?11	KSP AV	An access violation or translation not valid occurred during processing of a kernel stack not valid exception.
?12	DBL ERR2	Double machine check error. A machine check occurred while trying to service a machine check.
?13	DBL ERR3	Double machine check error. A machine check occurred while trying to service a kernel stack not valid exception.
?19	PSL EXC5[1]	PSL<26:24> = 5 on interrupt or exception.

[1]For the last six cases, the VAX architecture does not allow execution on the interrupt stack while in a mode other than kernel. In the first three cases, an interrupt is attempting to run on the interrupt stack while not in kernel mode. In the last three cases, an REI instruction is attempting to return to a mode other than kernel and still run on the interrupt stack.

(continued on next page)

Table G–1 (Cont.) HALT Messages

Code	Message	Description
?1A	PSL EXC6[1]	PSL<26:24> = 6 on interrupt or exception.
?1B	PSL EXC7[1]	PSL<26:24> = 7 on interrupt or exception.
?1D	PSL REI5[1]	PSL<26:24> = 5 on an REI instruction
?1E	PSL REI6[1]	PSL<26:24> = 6 on an REI instruction.
?1F	PSL REI7[1]	PSL<26:24> = 7 on an REI instruction.
?3F	MICROVERIFY FAILURE	Microcode power-up self-test failed.

[1]For the last six cases, the VAX architecture does not allow execution on the interrupt stack while in a mode other than kernel. In the first three cases, an interrupt is attempting to run on the interrupt stack while not in kernel mode. In the last three cases, an REI instruction is attempting to return to a mode other than kernel and still run on the interrupt stack.

G.3 VMB Error Messages

VMB issues the errors listed in Table G–2.

Table G–2 VMB Error Messages

Code	Message	Description
?40	NOSUCHDEV	No bootable devices found.
?41	DEVASSIGN	Device is not present.
?42	NOSUCHFILE	Program image not found.
?43	FILESTRUCT	Invalid boot device file structure.
?44	BADCHKSUM	Bad checksum on header file.
?45	BADFILEHDR	Bad file header.
?46	BADIRECTORY	Bad directory file.
?47	FILNOTCNTG	Invalid program image format.
?48	ENDOFFILE	Premature end of file encountered.
?49	BADFILENAME	Bad filename given.
?4A	BUFFEROVF	Program image does not fit in available memory.
?4B	CTRLERR	Boot device I/O error.

(continued on next page)

Table G–2 (Cont.) VMB Error Messages

Code	Message	Description
?4C	DEVINACT	Failed to initialize boot device.
?4D	DEVOFFLINE	Device is offline.
?4E	MEMERR	Memory initialization error.
?4F	SCBINT	Unexpected SCB exception or machine check.
?50	SCB2NDINT	Unexpected exception after starting program image.
?51	NOROM	No valid ROM image found.
?52	NOSUCHNODE	No response from load server.
?53	INSFMAPREG	The Q22–bus map initialization failed.
?54	RETRY	No devices bootable, retrying.
?55	IVDEVNAM	Invalid device name.
?56	DRVERR	Drive error.

G.4 Console Error Messages

The error messages listed in Table G–3 are issued in response to a console command that has error(s).

Table G–3 Console Error Messages

Code	Message	Description
?61	CORRUPTION	The console program database has been corrupted.
?62	ILLEGAL REFERENCE	Illegal reference. The requested reference would violate virtual memory protection, the address is not mapped, the reference is invalid in the specified address space, or the value is invalid in the specified destination.
?63	ILLEGAL COMMAND	The command string cannot be parsed.
?64	INVALID DIGIT	A number has an invalid digit.
?65	LINE TOO LONG	The command was too large for the console to buffer. The message is issued only after receipt of the terminating carriage return.
?66	ILLEGAL ADRRESS	The address specified falls outside the limits of the address space.

(continued on next page)

Table G–3 (Cont.) Console Error Messages

Code	Message	Description
?67	VALUE TOO LARGE	The value specified does not fit in the destination.
?68	QUALIFIER CONFLICT	Qualifier conflict; for example, two different data sizes are specified for an EXAMINE command.
?69	UNKNOWN QUALIFIER	The switch is unrecognized.
?6A	UNKNOWN SYMBOL	The symbolic address in an EXAMINE or DEPOSIT command is unrecognized.
?6B	CHECKSUM	The command or data checksum of an X command is incorrect. If the data checksum is incorrect, this message is issued, and is not abbreviated to "Illegal command".
?6C	HALTED	The operator entered a HALT command.
?6D	FIND ERROR	A FIND command failed either to find the RPB or 128 KB of good memory.
?6E	TIME OUT	During an X command, data failed to arrive in the time expected (60 seconds).
?6F	MEMORY ERROR	A machine check occurred with a code indicating a read or write memory error.
?70	UNIMPLEMENTED	Unimplemented function.
?71	NO VALUE QUALIFIER	Qualifier does not take a value.
?72	AMBIGUOUS QUALIFIER	There were not enough unique characters to determine the qualifier.
?73	VALUE QUALIFIER	Qualifier requires a value.
?74	TOO MANY QUALIFIERS	Too many qualifiers supplied for this command.
?75	TOO MANY ARGUMENTS	Too many arguments supplied for this command.
?76	AMBIGUOUS COMMAND	There were not enough unique characters to determine the command.
?77	TOO FEW ARGUMENTS	Insufficient arguments supplied for this command.
?78	TYPEAHEAD OVERFLOW	The typeahead buffer overflowed.
?79	FRAMING ERROR	A framing error was detected on the console serial line.
?7A	OVERRUN ERROR	An overrun error was detected on the console serial line.
?7B	SOFT ERROR	A soft error occurred.
?7C	HARD ERROR	A hard error occurred.
?7D	MACHINE CHECK	A machine check occurred.

(continued on next page)

Table G–3 (Cont.) Console Error Messages

Code	Message	Description
?7E	CONSOLE STACK OVERRUN	SSC RAM stack overflowed into NVR.
?7F	COMMAND NOT SUPPORTED	Command on similar modules not supported on this product.
?80	ILLEGAL PASSWORD	Password is not 16 characters in length.
?81	INCORRECT PASSWORD	Password entered does not match previously entered password.
?82	PASSWORD FACILITY NOT ENABLED	A password has not been set.

H

Related Documents

The following documents contain information relating to the maintenance of systems that use the KA50/51/55/56 CPU modules.

Title	Part Number[1]
Guide to BA42B-Based MicroVAX 3100 Systems Service Information Kit	EK–M3100–IN
MicroVAX 3100 Models 30, 40, 80, 85, 90, 95, 96 System Illustrated Parts Breakdown	EK–MV310–IP
MicroVAX 3100 BA42B Enclosure Maintenance	EK–M3100–MG
MicroVAX 3100 BA42B Enclosure System Options	EK–M3100–OP
OpenVMS Factory Installed Software User Guide	EK–A0377–UG

[1] # = current revision, which is always shipped.

Glossary

ASCII

American standard code for information interchange.

BFLAG

Boot FLAG is the longword supplied in the SET BFLAG and BOOT /R5: commands that qualify the bootstrap operation. SHOW BFLAG displays the current value.

BHALT

Q22–bus Halt signal is usually tied to the front panel Halt switch.

BIP

Boot In Progress flag in CPMBX<2>

Bootstrap

A link between console mode (the system firmware) and programming mode (the operating system).

Bugcheck

Software or hardware error fatal to OpenVMS processor or system.

Cache memory

A small, high-speed memory placed between slower main memory and the processor. A cache increases effective memory transfer rates and processor speed.

CMOS

Complementary metal oxide semiconductor.

CPMBX

Console Program Mailbox is used to pass information between operating systems and the firmware.

CRC

Character code recognition. The use of pattern recognition techniques to identify characters by automatic means.

CQBIC

CVAX to Q22–bus interface chip.

CSR

Control status register. A register used to control the operation of a device and record the status of an operation or both.

CPU

Central processing unit. The main unit of a computer containing the circuits that control the interpretation and execution of instructions. The CPU holds the main storage, arithmetic unit, and special registers.

DCOK

Q22–bus signal indicating dc power is stable. This signal is tied to the Restart switch on the System Control Panel.

DE

Diagnostic Executive is a component of the ROM-based diagnostics responsible for set-up, execution, and clean-up of component diagnostic tests.

DMA

Direct memory access. A method of accessing a device's memory without interacting with the device's CPU.

DNA

Digital Network Architecture.

EPROM

Erasable programmable read-only memory. EPROM is a type of read-only memory that can be erased by using ultraviolet light, returning the device to a blank state.

ECC

Error Correction Code. Code that carries out automatic error correction by performing an exclusive "or" operation on the transferred data and applying a correction mask.

Factory Installed Software (FIS)

Operating system software loaded into a system disk during manufacture. On site, the FIS is bootstraped in the system, prompting a predefined menu of questions on the final configuration.

FEPROM

Flash Erasable Programmable Read-Only Memory (FEPROM). FEPROMs use electrical (bulk) erasure rather than ultraviolet erasure.

FIFO

First-in/first-out. A method used for processing or recovering data in which the oldest item is processed or recovered first.

Firmware

Functionally it consists of diagnostics, bootstraps, console, and halt entry/exit code.

FPU

Floating-point unit. A unit that handles the automatic positioning of the decimal point during arithmetic operations.

FRU

Field replaceable unit.

GPR

General Purpose Registers. On the KA52/53, they are the sixteen standard VAX longword registers R0 through R15. The last four registers, R12 through R15, are also known by their unique mnemonics: AP (Argument Pointer), FP (Frame Pointer), SP (Stack Pointer), and PC (Program Counter), respectively.

Initialization

The sequence of steps that prepare the system to start. Initialization occurs after a system has been powered up.

IPL

Interrupt Priority Level ranges from 0 to 31 (0 to 1F hex).

IPR

Internal Processor Registers implemented by the processor chip set. These longword registers are only accessible with the instructions MTPR (Move To Processor Register) and MFPR (Move From Processor Register) and require kernel mode privileges. This document uses the prefix "PR$_" when referencing these registers.

ISE

Integrated storage element. An intelligent disk drive used on the Digital Storage Systems Interconnect.

IT

Interval timer.

LED

Light emitting diode.

Machine check

An operating system action triggered by certain system errors that can be fatal to system operation. Once triggered, machine check handler software analyzes the error, comparing it to predetermined failure scenarios. Three outcomes are possible: the system continues to run, the software program is halted, or the system crashes.

µs

Microsecond (10e-6 seconds)

MMJ

Modified modular jack.

MOP

Maintenance Operations Protocol specifies message protocol for network loopback assistance, network bootstrap, and remote console functions.

ms

Millisecond (10e-3 seconds)

MSCP

Mass Storage Control Protocol is used in Digital disks and tapes.

NVR

Nonvolatile random access memory. A memory device that retains information in the absence of power.

NVRAM

Nonvolatile RAM. On the KA52/53, this is 1 Kb of battery backed-up RAM on the SSC.

PC

Program Counter or R15.

PCB

Process Control Block is a data structure pointed to by the PR$_PCBB register and contains the current process' hardware context.

PFN

Page Frame Number is an index of a page (512 bytes) of local memory. A PFN is derived from the bit field <23:09> of a physical address.

PR$_ICC

Interval Clock Control and Status, IPR 24.

PR$_IPL

Interrupt Priority Level, IPR 18.

PR$_MAPEN

Memory Management Mapping Enable, IPR 56.

PR$_PCBB

Process Control Block Base register, IPR 16.

PR$_RXCS

R(X)eceive Console Status, IPR 32.

PR$_RXDB

R(X)eceive Data Buffer, IPR 33.

PR$_SAVISP

SAVed Interrupt Stack Pointer, IPR 41.

PR$_SAVPC

SAVed Program Counter, IPR 42.

PR$_SAVPSL

SAVed Program Status Longword, IPR 43.

PR$_SCBB

System Control Block Base register, IPR 17.

PR$_SISR

Software Interrupt Summary Register, IPR 21.

PR$_TODR

Time Of Day Register, IPR 27, is commonly referred to as the Time Of Year register or TOY clock.

PR$_TXCS

T(X)ransmit Console Status, IPR 34.

PR$_TXDB

T(X)ransmit Data Buffer, IPR 35.

PROM

Programmable read-only memory. A read-only memory device that can be programmed.

PSL, PSW

Processor Status Longword is the VAX extension of the PSW (Processor Status Word). The PSW (lower word) contains instruction condition codes and is accessible by nonprivileged users; however, the upper word contains system status information and is accessible by privileged users.

QBMB

Q22–bus Map Base Register found in the CQBIC determines the base address in local memory for the scatter/gather registers.

QDSS

Q22–bus video controller for workstations.

QMR

Q22–bus Map Register.

QNA

Q22–bus Ethernet controller module.

RAM

Random access memory. A read/write memory device.

RAP

Register address port.

RIP

Restart In Progress flag in CPMBX<3>.

ROM

Read-only memory. A memory device that cannot be altered during the normal use of the computer.

RPB

Restart parameter block.

SCB

System Control Block. A data structure pointed to by PR$_SCBB. It contains a list of longword exception and interrupt vectors.

SCSI

Small computer system interface. An interface designed for connecting disks and other peripheral devices to computer systems. SCSI is defined by an American National Standards Institute (ANSI) standard.

SDD

Symptom-Directed Diagnosis. Online analysis of nonfatal system errors in order to locate potential system fatal errors before they occur.

SGEC

Second Generation Ethernet Chip.

SHAC

Single Host Adapter Chip.

SP

Stack pointer. An address location that contains the address of the processor-defined stack. The processor-defined stack is an area of memory set aside for temporary storage or for procedure and interrupt service linkages.

SRM

Standard Reference Manual, as in *VAX SRM*.

SSC

System Support Chip.

TOY

Time of year.

VAXcluster configuration

A highly integrated organization of OpenVMS systems that communicate over a high-speed communications path. VAXcluster configurations have all the functions of single-node systems, plus the ability to share CPU resources, queues, and disk storage. Like a single-node system, the VAXcluster configuration provides a single security and management environment. Member nodes can share the same operating environment or serve specialized needs.

VMB

Virtual machine bootstrap. The VMB program loads and runs the operating system.

OpenVMS

Virtual memory system. The operating system for a VAX computer.

Index

A

Acceptance testing, 4–13 to 4–14
Algorithm
  to find a valid RPB, 4–32
  to restart operating system, 4–31
ANALYZE/ERROR, 5–14
  interpreting CPU errors using, 5–15
  interpreting DMA to host transaction faults using, 5–28
  interpreting memory errors using, 5–18
  interpreting system bus faults using, 5–26
ANALYZE/SYSTEM, 5–21
Asynchronous communications interfaces
  support for, 2–4
Asynchronous communications options
  list of, 2–4

B

Binary load and unload (X command), 3–35
Bits
  RPB$V_DIAG, 4–24
  RPB$V_SOLICT, 4–24
Block diagram, 1–3
Boot Block Format, 4–23
BOOT command, 3–13
Boot Flags
  RPB$V_BBLOCK, 4–23
Bootstrap
  conditions, 4–17
  definition of, 4–17
  disk and tape, 4–23
Bootstrap (cont'd)
  failure, 4–18
  initialization, 4–18
  memory layout, 4–19
  memory layout after successful bootstrap, 4–21
  network, 4–24
  preparing for, 4–18
  primary, 4–20
  secondary, 4–20

C

Comment command (!), 3–38
! (comment command), 3–38
Communications devices, 2–4
Communications options, 2–4
Configuration
  memory, 1–9
Connectors
  function of, 1–6
  identification of, 1–5
Console command
  LOGIN, 3–21
Console commands
  address space control qualifiers, 3–9
  address specifiers, 3–3
  binary load and unload (X), 3–35
  BOOT, 3–13
  ! (comment), 3–38
  CONTINUE, 3–15
  data control qualifiers, 3–9
  DEPOSIT, 3–15
  EXAMINE, 3–16
  FIND, 3–17

Console commands (cont'd)
HALT, 3–18
HELP, 3–18
INITIALIZE, 3–20
keywords, 3–10
list of, 3–11
MOVE, 3–22
NEXT, 3–23
qualifier and argument conventions, 3–3
qualifiers, 3–9
REPEAT, 3–24
SEARCH, 3–25
SET, 3–27
SHOW, 3–28
START, 3–31
symbolic addresses, 3–4
syntax, 3–3
TEST, 3–31
UNJAM, 3–35
X (binary load and unload), 3–35
Console error messages
sample of, 5–41
Console I/O mode
special characters, 3–2
Console port, testing, 5–58
Console security feature
values, 3–28
CONTINUE command, 3–15
Controls
function of, 1–6
identification of, 1–5

D

DEPOSIT command, 3–15
Device Dependent Bootstrap Procedures, 4–23
Diagnostic executive, 4–6
error field, 5–42
Diagnostics
relationship to UETP, 5–56
Diagnostics, RZ-series, 4–5
Diagnostic tests
list of, 4–7
parameters for, 4–7
DNA Maintenance Operations Protocol (MOP), 4–24
Documents
related, H–1

E

Entry Point
definition of, B–1
Error during UETP, 5–57
diagnosing, 5–56
Error Log Utility
relationship to UETP, 5–56
Error messages
console, sample of, 5–41
EXAMINE command, 3–16
External mass storage devices, 2–2

F

FE utility, 5–47
Files–11 lookup, 4–23
FIND command, 3–17
Firmware
power-up sequence, 4–1
updating, 6–1
Flags
restart in progress, 4–31

G

General purpose registers (GPRs)
symbolic addresses for, 3–4

H

H3103 loopback connector, 5–58
H8572 loopback connector, 5–58
Halt
dispatch, C–1
HALT command, 3–18
Halt protection, override, 5–48
HELP command, 3–18

I

Indicators
  function of, 1–6
  identification of, 1–5
INIT, 4–18
Initialization
  following a processor halt, 4–31
  prior to bootstrap, 4–18
INITIALIZE command, 3–20
Initial power-up test
  See IPR
Internal mass storage devices
  list of, 2–2
IPL_31, 4–19
iSYS$TEST logical name, 5–56

K

KA50/51/55/56 CPU module
  block diagram of, 1–3
KA50/51/55/56 system
  communications options, 2–4
  configurations of, 2–1
  mass storage device configurations, 2–1
  memory configurations, 2–1
KA50 CPU module
  features of, 1–1
KA50 system
  configurations of, 1–1
KA51 CPU module
  features of, 1–1
KA51 system
  configurations of, 1–1
KA55 CPU module
  features of, 1–1
KA55 system
  configurations of, 1–1

L

Language selection menu
  messages, list of, 4–2
Local Memory Partitioning, 4–19
Log file generated by UETP
  OLDUETP.LOG, 5–57
LOGIN command, 3–21
Loopback connectors
  H3103, 5–58
  H8572, 5–58
  list of, 5–59
Loopback tests, 5–58
  console port, 5–58
  Ethernet, 5–58

M

Mass storage devices, 2–1
  external, 2–2
  internal, 2–2
  SCSI ID assignments, 2–4
Memory
  acceptance testing of, 4–13
  configurations, 1–10
  expansion connector identification, 1–9
  expansion of, 1–9
  isolating FRU, 4–14, 5–48
  rules for adding, 1–9
  testing, 5–48
Memory configuration
  KA50/51/55/56 system, 2–1
Memory modules, 1–9
Memory option
  installation of, 1–11
Memory test, 1–13
MEM test, 1–13
MicroVAX data types
  support of, 1–4
MicroVAX instructions
  support of, 1–5
MOM$LOAD, 4–25

MOP, functions, 5–53
MOP functions, 4–26
MOP program load sequence, 4–25
MOVE command, 3–22
MS44 memory modules, 1–9

N

Network listening, 4–30
NEXT command, 3–23
NVRAM
  CPMBX, E–2
  partitioning, E–1

O

OLDUETP.LOG file, 5–56
Onboard memory
  location of, 1–9
OpenVMS
  error handling, 5–4
  event record translation, 5–14
Operating System
  bootstrap, 4–17
  restarting a halted, 4–31
Operating System Restart
  definition of, 4–31

P

Parameters
  for diagnostic tests, 4–9
  in error display, 5–42
Patchable Control Store
  Error messages, 6–8
PFN bitmap, 4–18
Ports
  function of, 1–6
  identification of, 1–5
POST
  See Power-on self-tests
Power-on self-tests
  description, 4–2
  errors handled by, 4–5
  kernel, 4–3
Power-on self-tests (cont'd)
  mass storage, 4–5
power-up
  machine state, 4–14
  memory layout, 4–15
Power-up sequence, 4–1
Power-up tests, 4–1
Primary Bootstrap, 4–20

R

Registers
  initializing the general purpose, 4–18
Related documents, H–1
REPEAT command, 3–24
REQ_PROGRAM, 4–30
Restart, 4–31
Restart Parameter Block (RPB)
  RIP flag, 4–31
ROM-based diagnostics, 4–6 to 4–10
  console displays during, 5–41
  isolating failures with, 5–43
  list of, 4–6
  parameters, 4–7
  utilities, 4–6
RPB
  initialization, C–5
  locating, 4–32
RPB Signature Format, 4–32
RZ-series ISE
  diagnostics, 4–5

S

Scripts, 4–11
  list of, 4–12
SCSI ID assignments
  recommendations for, 2–4
SEARCH command, 3–25
Secondary Bootstrap, 4–20
SET command, 3–27
SET HOST/DUP command, 3–27
SHOW command, 3–28

SICL messages, 5–34
converting appended MEL files, 5–37
START command, 3–31
Symbolic addresses, 3–4
for any address space, 3–7
for GPRs, 3–4
Synchronous communications options
list of, 2–5
Synchronous communications standards
support for, 2–5
System hang, 5–58

T

TEST command, 3–31
Tests, diagnostic
list of, 4–6
parameters for, 4–9
Troubleshooting
procedures, general, 5–2
UETP, 5–57

U

UETINIT01.EXE image, 5–57
UETP
interpreting OpenVMS failures with, 5–56
UETP.LOG file, 5–56
UNJAM, 4–18
UNJAM command, 3–35
User Environment Test Package (UETP)
interpreting output of, 5–56
running multiple passes of, 5–56
typical failures reported by, 5–57
Utilities, diagnostic, 4–6

V

VAXELN
and VMB, 4–20
VAXsimPLUS, 5–3, 5–32
customizing, 5–39
enabling SICL, 5–40
installing, 5–38
Virtual Memory Boot (VMB), 4–21
definition of, 4–20
primary bootstrap, 4–20
secondary bootstrap, 4–23

W

Warmstart, 4–31

X

X command (binary load and unload), 3–35