

## **<REFERENCE>(VAX\_mod\_XXX) Service Manual**

Order Number EK-650EA-MG-001

This manual is intended for Digital customer service engineers. It covers processor-specific and troubleshooting information. This manual is to be used with the *VAX 6000 Platform Service Manual*.

**digital equipment corporation  
maynard, massachusetts**

---

**First Printing, November 1990**

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.


The software, if any, described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license. No responsibility is assumed for the use or reliability of software or equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Copyright ©1990 by Digital Equipment Corporation.

All Rights Reserved.  
Printed in U.S.A.

---

The following are trademarks of Digital Equipment Corporation:

DEMNA	PDP	VAXcluster
DEC	ULTRIX	VAXELN
DEC LANcontroller	UNIBUS	VMS
DECnet	VAX	XMI
DECUS	VAXBI	

**FCC NOTICE:** The equipment described in this manual generates, uses, and may emit radio frequency energy. The equipment has been type tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such radio frequency interference when operated in a commercial environment. Operation of this equipment in a residential area may cause interference, in which case the user at his own expense may be required to take measures to correct the interference.

---

# Contents

---

Preface	xi
---------	----

---

## Chapter 1 Introduction

---

1.1 System Functional Description	1-2
1.2 Troubleshooting Flowcharts	1-4

## Chapter 2 Diagnostics

---

2.1 Diagnostic Overview	2-2
2.2 Self-Test and Additional Power-Up Tests	2-4
2.2.1 Checking Self-Test Results: Console Display	2-6
2.2.2 Checking Self-Test Results: Status LEDs	2-8
2.2.3 KA65A LEDs	2-10
2.2.4 Determining Failing Test from LEDs	2-12
2.2.5 KA65A Error LED	2-14
2.2.6 Checking Self-Test Results: XBER and XGPR Registers	2-16
2.3 ROM-Based Diagnostic Monitor Program	2-18
2.3.1 RBD Monitor Control Characters	2-20
2.3.2 START Command	2-22
2.3.3 START Command Qualifiers	2-23
2.3.4 RBD Test Printout, Passing	2-26
2.3.5 RBD Test Printout, Failing	2-28
2.3.6 SUMMARY Command	2-30
2.3.7 Sample RBD Session	2-32
2.3.8 Running ROM-Based Diagnostics on I/O Devices	2-38
2.4 ROM-Based Diagnostics	2-40
2.4.1 <REFERENCE>(xmp) Self-Test — RBD 0	2-42
2.4.2 CPU/Memory Interaction Tests — RBD 1	2-46
2.4.3 <REFERENCE>(xbi) Tests — RBD 2	2-48

2.4.4	DWMBB Tests — RBD 2 Subtests . . . . .	2-50
2.4.5	MS65A Memory Tests — RBD 3 . . . . .	2-52
2.4.6	KA65A Cache Tests — RBD 4 . . . . .	2-56
2.4.7	Multiprocessor Tests — RBD 5 . . . . .	2-58
2.5	VAX Diagnostic Supervisor Programs . . . . .	2-60
2.5.1	Running VAX/DS in Standalone Mode . . . . .	2-62
2.5.2	Running VAX/DS in User Mode . . . . .	2-64
2.5.3	Sample VAX/DS Session . . . . .	2-66
2.5.4	VAX/DS Diagnostics . . . . .	2-70

### Chapter 3 <REFERENCE>(xmp) Scalar Processor

---

3.1	<REFERENCE>(xmp) Physical Description and Specifications . . . . .	3-2
3.2	<REFERENCE>(xmp) Configuration Rules . . . . .	3-4
3.3	<REFERENCE>(xmp) Functional Description . . . . .	3-6
3.4	Boot Processor . . . . .	3-10
3.5	Power-Up Sequence . . . . .	3-12
3.6	ROM-Based Diagnostics . . . . .	3-16
3.7	VAX/DS Diagnostics . . . . .	3-18
3.8	Machine Checks . . . . .	3-20
3.9	Console Commands . . . . .	3-22
3.10	<REFERENCE>(xmp) Handling Procedures . . . . .	3-24
3.11	How to Replace the Only Processor . . . . .	3-28
3.12	How to Replace the Boot Processor . . . . .	3-30
3.13	How to Add a New Processor or Replace a Secondary Processor . . . . .	3-32
3.14	Using EVUCA to Patch the EEPROM . . . . .	3-34
3.15	<REFERENCE>(xmp) Registers . . . . .	3-36

### Chapter 4 FV64A Vector Processor

---

4.1	<REFERENCE>(xrv) Physical Description and Specifications . . . . .	4-2
4.2	<REFERENCE>(xmp)/FV64A Coprocessors . . . . .	4-4
4.3	<REFERENCE>(xrv) Configuration Rules . . . . .	4-6
4.4	<REFERENCE>(xrv) Functional Description . . . . .	4-8
4.5	Self-Test Results: Console Display and Self-Test LED . . . . .	4-10

4.6	Self-Test Results: Scalar XGPR Register .....	4-12
4.7	ROM-Based Diagnostics .....	4-14
4.8	VAX/DS Diagnostics .....	4-16
4.9	Machine Checks .....	4-17
4.10	Vector Console Commands .....	4-18
4.11	<REFERENCE>(xrv) Handling Procedures .....	4-22
4.12	How to Replace a Vector Module .....	4-26
4.13	Vector Processor Registers .....	4-28

## Chapter 5 MS65A Memory

---

5.1	MS65A Physical Description .....	5-2
5.2	MS65A Configuration Rules .....	5-4
5.3	MS65A Specifications .....	5-5
5.4	MS65A Functional Description .....	5-6
5.5	MS65A Interleaving .....	5-8
5.6	Console Commands for Interleaving .....	5-10
5.7	MS65A Addressing .....	5-12
5.8	Memory Self-Test .....	5-14
5.9	Memory Self-Test Errors .....	5-16
5.10	MS65A Control and Status Registers .....	5-18

## Chapter 6 DWMBB I/O Adapter

---

6.1	DWMBB Physical Description .....	6-2
6.1.1	Physical Layout .....	6-2
6.1.2	Specifications .....	6-4
6.2	<REFERENCE>(xbi) Configuration Rules .....	6-6
6.3	DWMBB Functional Description .....	6-8
6.4	<REFERENCE>(xbi) Registers .....	6-10

## Appendix A Console Error Messages

---

## Appendix B Boot Status and Error Messages

---

B.1	Ethernet MOP Boot Status and Error Messages . . . . .	B-1
B.2	Disk Boot Status and Error Boot Messages . . . . .	B-2
B.3	Tape Status and Error Boot Messages . . . . .	B-2
B.4	CI Status and Error Boot Messages . . . . .	B-4

## Appendix C KA65A LED Patterns Indicating Console Errors

---

## Appendix D Parse Trees

---

## Appendix E Restoring a Corrupted EEPROM

---

## Glossary

---

## Index

---

## Examples

---

2-1	Self-Test Results . . . . .	2-6
2-2	Examining the XCR0 and XBER Registers . . . . .	2-14
2-3	XGPR Register After Power-Up Test Failure . . . . .	2-16
2-4	START Command . . . . .	2-22
2-5	RBD Test Printout, Passing . . . . .	2-26
2-6	RBD Test Printout, Failing . . . . .	2-28
2-7	SUMMARY Command . . . . .	2-30
2-8	Sample RBD Session, Part 1 of 3 . . . . .	2-32
2-9	Sample RBD Session, Part 2 of 3 . . . . .	2-34
2-10	Sample RBD Session, Part 3 of 3 . . . . .	2-36
2-11	Running RBDs on I/O Devices . . . . .	2-38

2-12	<REFERENCE>(xmp) Self-Test — RBD 0 . . . . .	2-42
2-13	Running <REFERENCE>(xmp) Self-Test (RBD 0) on a Secondary Processor . . . . .	2-43
2-14	CPU/Memory Interaction Tests — RBD 1 . . . . .	2-46
2-15	<REFERENCE>(xbi) Tests — RBD 2 . . . . .	2-48
2-16	RBD Test on All Modules with Halt on Error . . . . .	2-52
2-17	RBD Test on Module in Slot A . . . . .	2-52
2-18	RBD Test with Module Error . . . . .	2-53
2-19	RBD Test with Confirm Switch . . . . .	2-53
2-20	KA65A Cache Tests — RBD 4 . . . . .	2-56
2-21	Multiprocessor Tests — RBD 5 . . . . .	2-58
2-22	Running VAX/DS in Standalone Mode . . . . .	2-62
2-23	Running VAX/DS in User Mode . . . . .	2-64
2-24	Sample VAX/DS Session, Part 1 of 2 . . . . .	2-66
2-25	Sample VAX/DS Session, Part 2 of 2 . . . . .	2-68
3-1	ROM and EEPROM Version Numbers . . . . .	3-8
3-2	VAX/DS Commands for Running Standalone Processor Diagnostics . . . . .	3-18
3-3	Replacing a Single Processor . . . . .	3-28
3-4	Replacing Boot Processor . . . . .	3-30
3-5	Adding or Replacing Secondary Processor . . . . .	3-32
3-6	Patching the EEPROM with EVUCA . . . . .	3-34
4-1	Self-Test Results . . . . .	4-10
4-2	XGPR Register After Power-Up Test Failure . . . . .	4-12
4-3	VAX/DS Commands for Testing Vector Processors . . . . .	4-16
5-1	SET MEMORY and INITIALIZE Commands . . . . .	5-10
5-2	MS65A Memory Module Results in Self-Test . . . . .	5-14
5-3	MS65A Memory Module Node Exclusion . . . . .	5-16
E-1	Restoring a Corrupted EEPROM, Part 1 of 2 . . . . .	E-2
E-2	Restoring a Corrupted EEPROM, Part 2 of 2 . . . . .	E-4

## Figures

---

1-1	<REFERENCE>(VAX_mod_XXX) System Architecture . . . . .	1-2
1-2	Power-Up . . . . .	1-4
1-3	Boot VMS . . . . .	1-6
1-4	Control Panel Lights Do Not Work . . . . .	1-7
1-5	System Shutdown 30 Seconds After Power-Up . . . . .	1-8
1-6	No Console Output, Control Panel Fault LED Is On . . . . .	1-9
1-7	No Console Output, Control Panel Fault LED Is Off . . . . .	1-10
1-8	DWMBB Fails Self-Test . . . . .	1-11
2-1	Diagnostics Design . . . . .	2-2
2-2	Location of Status LEDs . . . . .	2-8
2-3	<REFERENCE>(xmp) LEDs After Power-Up Self-Test . . . . .	2-10
3-1	<REFERENCE>(xmp) Module . . . . .	3-2
3-2	Typical <REFERENCE>(xmp) Configuration . . . . .	3-4
3-3	<REFERENCE>(xmp) Block Diagram . . . . .	3-6
3-4	Selection of Boot Processor . . . . .	3-10
3-5	<REFERENCE>(xmp) Power-Up Sequence, Part 1 of 2 . . . . .	3-12
3-6	<REFERENCE>(xmp) Power-Up Sequence, Part 2 of 2 . . . . .	3-14
3-7	The Stack in Response to a Machine Check . . . . .	3-20
3-8	Holding the <REFERENCE>(xmp) Module . . . . .	3-24
3-9	Inserting the <REFERENCE>(xmp) Module in an XMI Card Cage . . . . .	3-26
4-1	<REFERENCE>(XRV) Module (Side 1) . . . . .	4-2
4-2	<REFERENCE>(XRV) Module (Side 2) . . . . .	4-3
4-3	VAX 6000 Model 500 Vector Processing System . . . . .	4-4
4-4	Scalar/Vector Configurations . . . . .	4-6
4-5	<REFERENCE>(XRV) Block Diagram . . . . .	4-8
4-6	XGPR Register . . . . .	4-12
4-7	The Stack in Response to a Machine Check . . . . .	4-17
4-8	Holding the <REFERENCE>(xrv) Module . . . . .	4-22
4-9	Inserting the <REFERENCE>(xrv) Module in an XMI Card Cage . . . . .	4-24
4-10	Replacing a Vector Module in an XMI Card Cage . . . . .	4-26
5-1	MS65A Module . . . . .	5-2
5-2	MS65A Configuration . . . . .	5-4
5-3	MS65A Block Diagram . . . . .	5-6



5-4	MS65A Interleaving . . . . .	5-8
5-5	MS65A Addressing . . . . .	5-12
6-1	<REFERENCE>(XBIA_TITLE) . . . . .	6-2
6-2	<REFERENCE>(XBIB_TITLE) . . . . .	6-3
6-3	<REFERENCE>(VAX_XXXX) Slot Numbers . . . . .	6-6
6-4	<REFERENCE>(XBI_TITLE) Block Diagram . . . . .	6-8
C-1	KA65A Module LEDs . . . . .	C-2
D-1	<REFERENCE>(xmp) Machine Check Parse Tree . . . . .	D-2
D-2	<REFERENCE>(xmp) Hard Error Interrupt Parse Tree . . . . .	D-7
D-3	<REFERENCE>(xmp) Soft Error Interrupt Parse Tree . . . . .	D-10
D-4	<REFERENCE>(xrv) Machine Check Parse Tree . . . . .	D-12
D-5	<REFERENCE>(xrv) Hard Error Interrupt Parse Tree . . . . .	D-13
D-6	<REFERENCE>(xrv) Soft Error Interrupt Parse Tree . . . . .	D-14
D-7	<REFERENCE>(xrv) Disable Fault Parse Tree . . . . .	D-15

## Tables

---

1	VAX 6000 Series Documentation . . . . .	xii
2	VAX 6000 Model Level Documentation . . . . .	xiii
3	Associated Documents . . . . .	xiv
2-1	ROM-Based Diagnostics Run at Power-Up . . . . .	2-4
2-2	Status LEDs After Self-Test . . . . .	2-9
2-3	<REFERENCE>(xmp) Status LEDs . . . . .	2-12
2-4	KA65A Error LED . . . . .	2-14
2-5	XMI Base Addresses . . . . .	2-17
2-6	Interpreting XGPR Failing Test Numbers . . . . .	2-17
2-7	RBD Monitor Commands . . . . .	2-18
2-8	ROM-Based Diagnostic Programs — Callable Tests . . . . .	2-18
2-9	RBD Monitor Control Characters . . . . .	2-20
2-10	START Command Qualifiers . . . . .	2-23
2-11	ROM-Based Diagnostic Programs — Callable Tests . . . . .	2-40
2-12	<REFERENCE>(xmp)/<REFERENCE>(xrv) Self-Test — RBD 0 . . . . .	2-43
2-13	CPU/Memory Interaction Tests — RBD 1 . . . . .	2-47
2-14	<REFERENCE>(XBI_TITLE) RBD Tests . . . . .	2-50
2-15	Memory Tests — RBD 3 . . . . .	2-54
2-16	RBD 3 Parameters . . . . .	2-55

2-17	KA65A Cache Tests — RBD 4 . . . . .	2-57
2-18	Multiprocessor Tests — RBD 5 . . . . .	2-59
2-19	RBD 5 Parameters . . . . .	2-59
2-20	VAX Diagnostic Program Levels . . . . .	2-60
2-21	VAX/DS Documentation . . . . .	2-60
2-22	VAX Diagnostic Supervisor Programs . . . . .	2-70
3-1	<REFERENCE>(xmp) Specifications . . . . .	3-3
3-2	KA65A ROM-Based Diagnostics . . . . .	3-16
3-3	<REFERENCE>(xmp) VAX/DS Diagnostics . . . . .	3-18
3-4	Machine Check Parameters . . . . .	3-20
3-5	Console Commands . . . . .	3-22
3-6	<REFERENCE>(xmp) Internal Processor Registers . . . . .	3-36
3-7	<REFERENCE>(XMI) Registers for the <REFERENCE>(xmp) . . . . .	3-39
3-8	<REFERENCE>(xmp) Registers in <REFERENCE>(XMI) Private Space . . . . .	3-40
4-1	<REFERENCE>(XRV) Specifications . . . . .	4-5
4-2	Processor Module Combinations . . . . .	4-7
4-3	Interpreting XGPR Failing Test Numbers . . . . .	4-13
4-4	FV64A ROM-Based Diagnostics . . . . .	4-14
4-5	<REFERENCE>(XRV) VAX/DS Diagnostics . . . . .	4-16
4-6	<REFERENCE>(xrv) Machine Check Parameters . . . . .	4-17
4-7	Vector Console Commands . . . . .	4-18
4-8	<REFERENCE>(XRV) Internal Processor Registers . . . . .	4-28
5-1	<REFERENCE>(XMA_TITLE) Specifications . . . . .	5-5
5-2	MS65A Control and Status Registers . . . . .	5-18
6-1	DWMBB/A Specifications . . . . .	6-4
6-2	DWMBB/B Specifications . . . . .	6-5
6-3	<REFERENCE>(xbi) Cables . . . . .	6-5
6-4	<REFERENCE>(xbi) Configuration . . . . .	6-7
6-5	VAXBI Registers . . . . .	6-10
6-6	<REFERENCE>(xbi) XMI Registers . . . . .	6-11
A-1	Console Error Messages Indicating Halt . . . . .	A-1
A-2	Standard Console Error Messages . . . . .	A-3
C-1	KA65A Console LED Patterns . . . . .	C-2

# Preface

---

## Intended Audience

This manual is written for Digital customer service engineers servicing the <REFERENCE>(VAX\_mod\_XXX) system.

## Document Structure

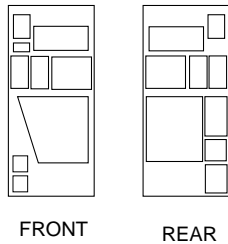
This manual uses a structured documentation design. There are many topics, organized into small sections for efficient reference. Each topic begins with an abstract. You can quickly gain a comprehensive overview by reading only the abstracts. Next is an illustration or example, which also provides quick reference. Last in the structure is descriptive text.

This manual has six chapters and five appendixes:

- **Chapter 1, Introduction**, gives an overview of the <REFERENCE>(vax\_mod\_xxx) system architecture and flowcharts for troubleshooting the system.
- **Chapter 2, Diagnostics**, describes the <REFERENCE>(vax\_xxxx) self-test, ROM-based diagnostics, and software diagnostics that run under the VAX Diagnostic Supervisor.
- **Chapter 3, <REFERENCE>(XmP) Scalar Processor, Chapter 4, <REFERENCE>(xrv) Vector Processor, Chapter 5, <REFERENCE>(XMA) Memory, and Chapter 6, DWMBB I/O Adapter**, give information on module specifications, configuration rules, and registers.
- **Appendix A** lists the console error messages. **Appendix B** contains the boot status and error messages. **Appendix C** shows KA65A LED patterns that indicate console errors. **Appendix D** gives the parse trees for the KA65A and FV64A processors. **Appendix E** is a procedure for restoring a corrupted EEPROM. A **Glossary** and **Index** provide additional reference support.

## Conventions Used in This Document

The icons shown below are used in illustrations for designating part placement in VAX 6000 systems. A shaded area in the icon shows the location of the component or part being discussed.



## <REFERENCE>(XXX) Documents

There are two sets of documentation: manuals that apply to all VAX 6000 series systems and manuals that are specific to one VAX 6000 model. Table 1 lists the manuals in the VAX 6000 series documentation set.

**Table 1: VAX 6000 Series Documentation**

<b>Title</b>	<b>Order Number</b>
<b>Operation</b>	
<REFERENCE>(XXX) <i>Owner's Manual</i>	EK-600EA-OM
<REFERENCE>(XXX) <i>Vector Processor Owner's Manual</i>	EK-60VAA-OM
<REFERENCE>(v6) <i>Vector Processor Programmer's Guide</i>	EK-60VAA-PG
<b>Service and Installation</b>	
<REFERENCE>(v6) <i>Platform Technical User's Guide</i>	EK-600EA-TM
<REFERENCE>(XXX) <i>Installation Guide</i>	EK-600EA-IN
<REFERENCE>(v6) <i>Installationsanleitung</i>	EK-600GA-IN
<REFERENCE>(v6) <i>Guide d'installation</i>	EK-600FA-IN
<REFERENCE>(v6) <i>Guía de instalacion</i>	EK-600SA-IN
<REFERENCE>(v6) <i>Platform Service Manual</i>	EK-600EA-MG

**Table 1 (Cont.): VAX 6000 Series Documentation**

<b>Title</b>	<b>Order Number</b>
<b>Options and Upgrades</b>	
<i>VAX 6000: XMI Conversion Manual</i>	EK-650EA-UP
<i>VAX 6000: Installing MS65A Memories</i>	EK-MS65A-UP
<i>VAX 6000: Installing the H7236 Battery Backup Option</i>	EK-60BBA-IN
<i>VAX 6000: Installing the FV64A Vector Option</i>	EK-60VEA-IN
<i>VAX 6000: Installing the VAXBI Option</i>	EK-60BIA-IN

Manuals specific to models are listed in Table 2.

**Table 2: VAX 6000 Model Level Documentation**

<b>Title</b>	<b>Order Number</b>
<b>Models 200/300/400</b>	
<i>VAX 6000 Model 300 and 400 Service Manual</i>	EK-624EA-MG
<i>VAX 6000: Installing Model 200/300/400 Processors</i>	EK-6234A-UP
<b>Model 500</b>	
<i>&lt;REFERENCE&gt;(MX) Mini-Reference</i>	EK-650EA-HR
<i>&lt;REFERENCE&gt;(MX) Service Manual</i>	EK-650EA-MG
<i>&lt;REFERENCE&gt;(MX) System Technical User's Guide</i>	EK-650EA-TM
<i>VAX 6000: Installing Model 500 Processors</i>	EK-KA65A-UP

## **Associated Documents**

Table 3 lists other documents that you may find useful.

**Table 3: Associated Documents**

<b>Title</b>	<b>Order Number</b>
<b>System Hardware Options</b>	
<i>VAXBI Expander Cabinet Installation Guide</i>	EK-VBIEA-IN
<i>VAXBI Options Handbook</i>	EB-32255-46
<b>System I/O Options</b>	
<i>CIBCA User Guide</i>	EK-CIBCA-UG
<i>CIXCD Interface User Guide</i>	EK-CIXCD-UG
<i>DEC LANcontroller 200 Installation Guide</i>	EK-DEBNI-IN
<i>DEC LANcontroller 400 Installation Guide</i>	EK-DEMNA-IN
<i>InfoServer 100 Installation and Owners Guide</i>	EK-DIS1K-IN
<i>KDB50 Disk Controller User's Guide</i>	EK-KDB50-UG
<i>KDM70 Controller User Guide</i>	EK-KDM70-UG
<i>RRD40 Disc Drive Owner's Manual</i>	EK-RRD40-OM
<i>RA90/RA92 Disk Drive User Guide</i>	EK-ORA90-UG
<i>SA70 Enclosure User Guide</i>	EK-SA70E-UG
<b>Operating System Manuals</b>	
<i>Guide to Maintaining a VMS System</i>	AA-LA34A-TE
<i>Guide to Setting Up a VMS System</i>	AA-LA25A-TE
<i>Introduction to VMS System Management</i>	AA-LA24A-TE
<i>ULTRIX-32 Guide to System Exercisers</i>	AA-KS95B-TE
<i>VMS Upgrade and Installation Supplement: VAX 6000 Series</i>	AA-LB36C-TE
<i>VMS Networking Manual</i>	AA-LA48A-TE
<i>VMS System Manager's Manual</i>	AA-LA00A-TE
<i>VMS VAXcluster Manual</i>	AA-LA27B-TE

**Table 3 (Cont.): Associated Documents**

<b>Title</b>	<b>Order Number</b>
<b>Peripherals</b>	
<i>HSC Installation Manual</i>	EK-HSCMN-IN
<i>H4000 DIGITAL Ethernet Transceiver Installation Manual</i>	EK-H4000-IN
<i>Installing and Using the VT320 Video Terminal</i>	EK-VT320-UG
<i>RV20 Optical Disk Owner's Manual</i>	EK-ORV20-OM
<i>SC008 Star Coupler User's Guide</i>	EK-SC008-UG
<i>TA78 Magnetic Tape Drive User's Guide</i>	EK-OTA78-UG
<i>TA90 Magnetic Tape Subsystem Owner's Manual</i>	EK-OTA90-OM
<i>TK70 Streaming Tape Drive Owner's Manual</i>	EK-OTK70-OM
<i>TU81/TA81 and TU/81 PLUS Subsystem User's Guide</i>	EK-TUA81-UG
<b>VAX Manuals</b>	
<i>VAX Architecture Reference Manual</i>	EY-3459E-DP
<i>VAX Systems Hardware Handbook — VAXBI Systems</i>	EB-31692-46
<i>VAX Vector Processing Handbook</i>	EC-H0739-46

# Chapter 1

## Introduction

---

This chapter is an overview of the <REFERENCE>(vax\_mod\_xxx) system. Sections include:

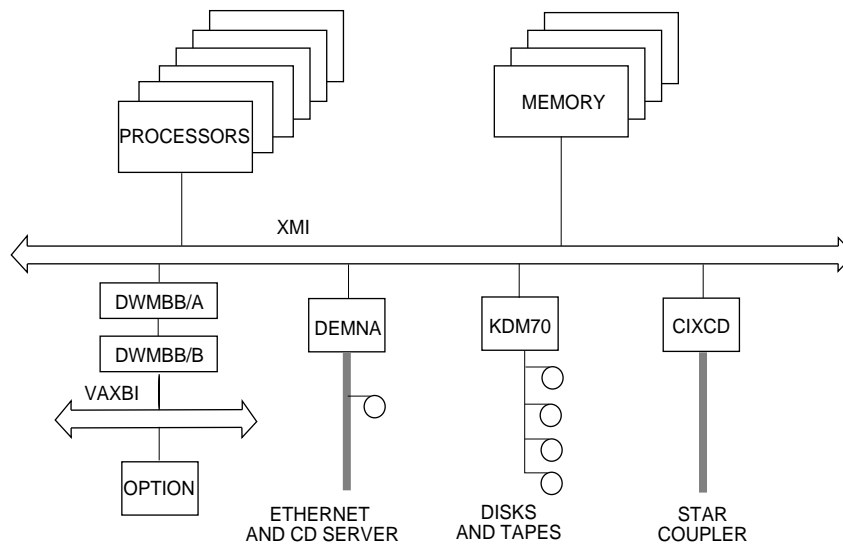
- System Functional Description
- Troubleshooting Flowcharts



## 1.1 System Functional Description

The <REFERENCE>(VAX\_mod\_XXX) system supports multi-processing with up to six <REFERENCE>(xmp) processors. The system uses the <REFERENCE>(XMI) bus as the system and I/O bus.

Figure 1-1: <REFERENCE>(VAX\_mod\_XXX) System Architecture



msb-0310-90

The <REFERENCE>(XMI) bus is the system and I/O bus; the VAXBI bus can also be used for I/O. The <REFERENCE>(XMI) bus is a 64-bit system bus<sup>1</sup> that interconnects the processors, memory modules, and I/O adapters.

The <REFERENCE>(XMI) bus has three types of nodes: processor nodes (<REFERENCE>(xmp) and <REFERENCE>(xrv)), memory nodes (<REFERENCE>(XMA)), and I/O adapter nodes.

A **processor node** is a single-board scalar processor (KA65A) or a scalar/vector processor pair (KA65A/FV64A). The CPU comprises two chips, one of which is a floating-point accelerator. A write-back cache subsystem improves system performance. In a multiprocessing system one scalar processor becomes the boot processor during power-up, and that boot processor loads the operating system and handles communication with the operator console. The other processors become secondary processors and receive system information from the boot processor.

This system supports multiprocessing with up to six processors. It supports vector processing with up to two scalar/vector processor pairs. Symmetric multiprocessing is supported, allowing a program to execute on any processor.

A **memory node** is an <REFERENCE>(XMA). Memory is a global resource equally accessible by all processors on the <REFERENCE>(XMI) bus. Each <REFERENCE>(XMA) module has 32, 64, or 128 Mbytes of memory, consisting of MOS 1-Mbit or 4-Mbit dynamic RAMs, ECC logic, and control logic. Memory access is automatically interleaved between modules. An optional battery backup unit protects memory in case of power failure.

**I/O adapters** are installed on the XMI bus. If this system has a VAXBI bus, the DWMBB adapter is used to connect VAXBI I/O adapters to the XMI bus.

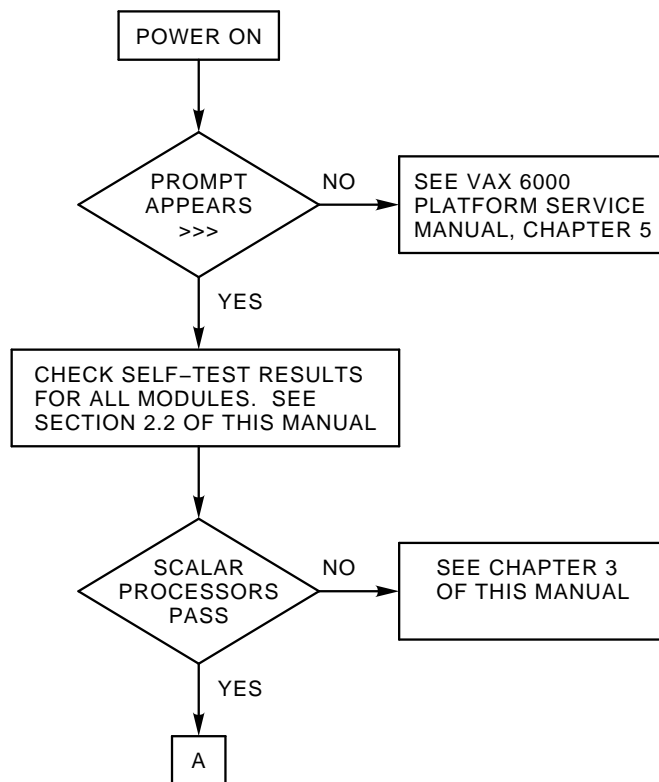
---

<sup>1</sup> The <REFERENCE>(XMI) bus has a 64-nanosecond bus cycle, with a maximum throughput of 100 Mbytes per second.

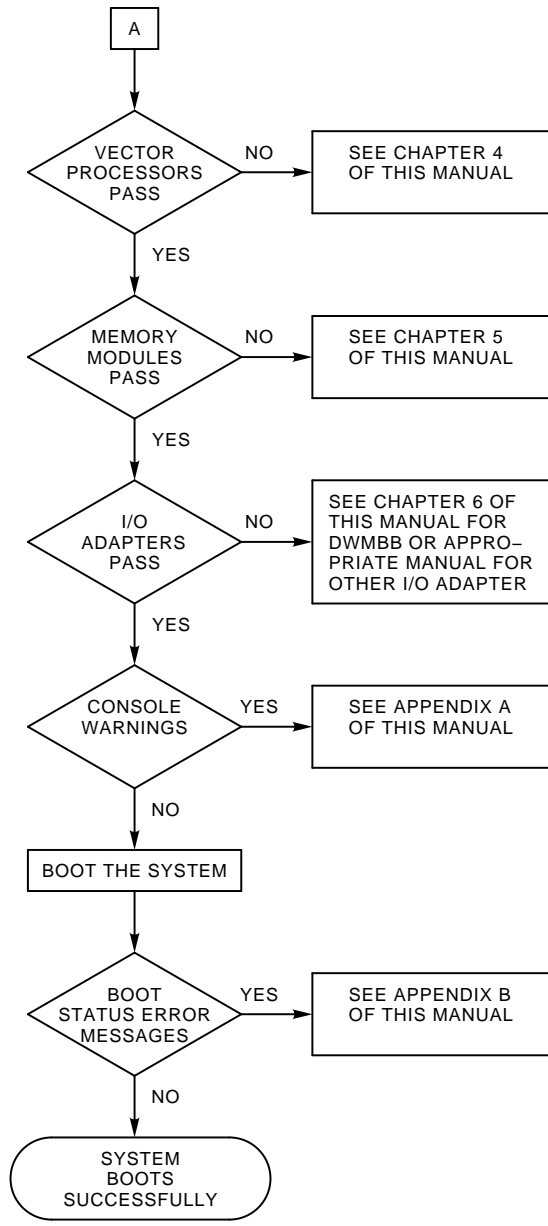
## 1.2 Troubleshooting Flowcharts

The following flowcharts reference sections in this manual and in the VAX 6000 Platform Service Manual.

Figure 1-2: Power-Up

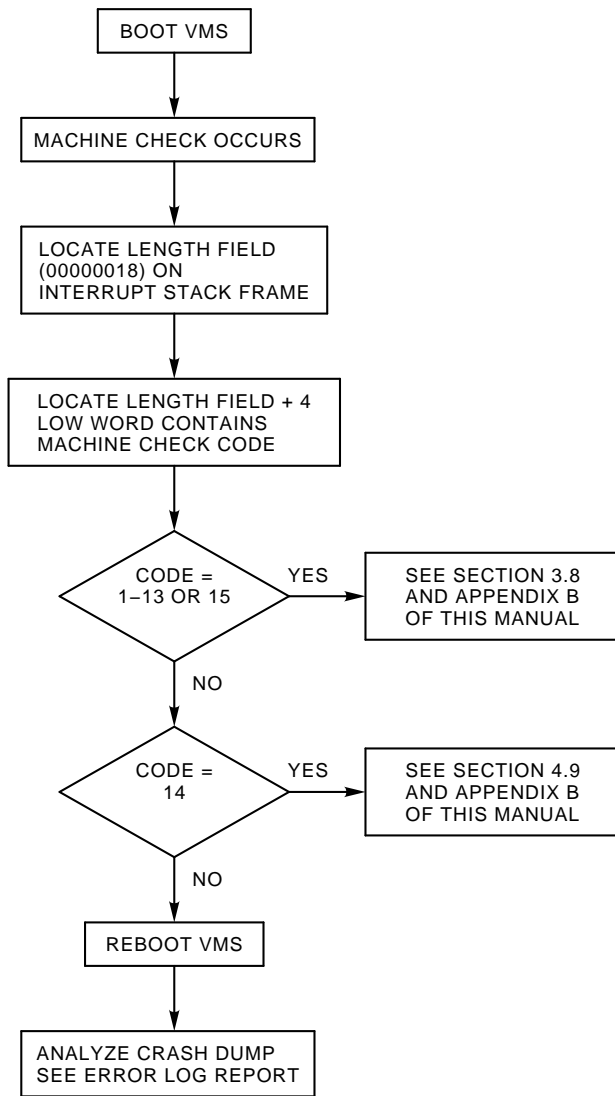


msb-p378A-90



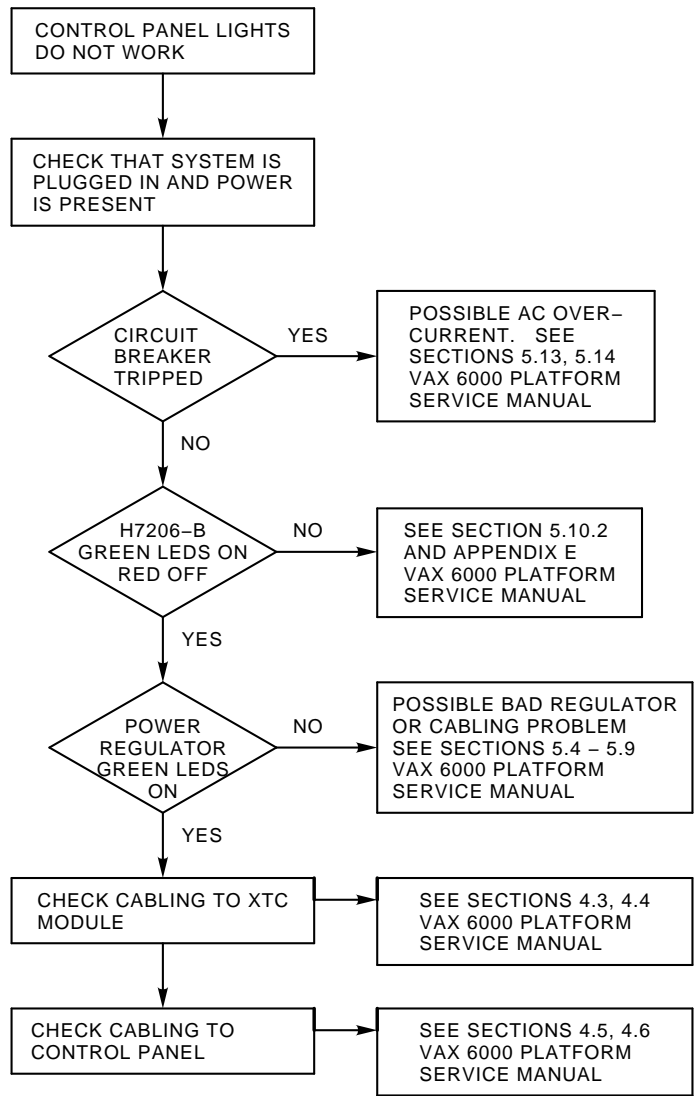
msb-p378Br-90

**Figure 1-3: Boot VMS**



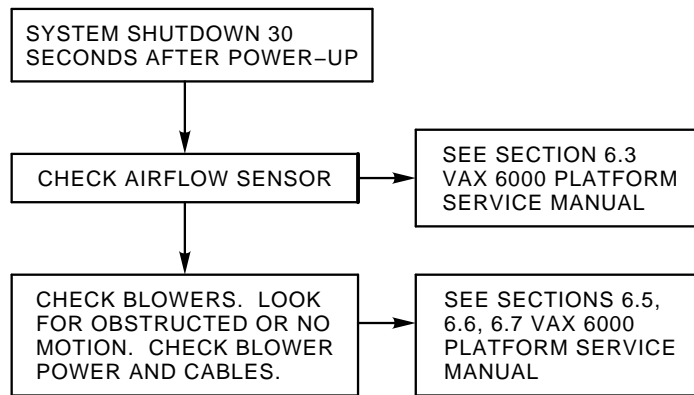
msb-p379r-90

**Figure 1-4: Control Panel Lights Do Not Work**



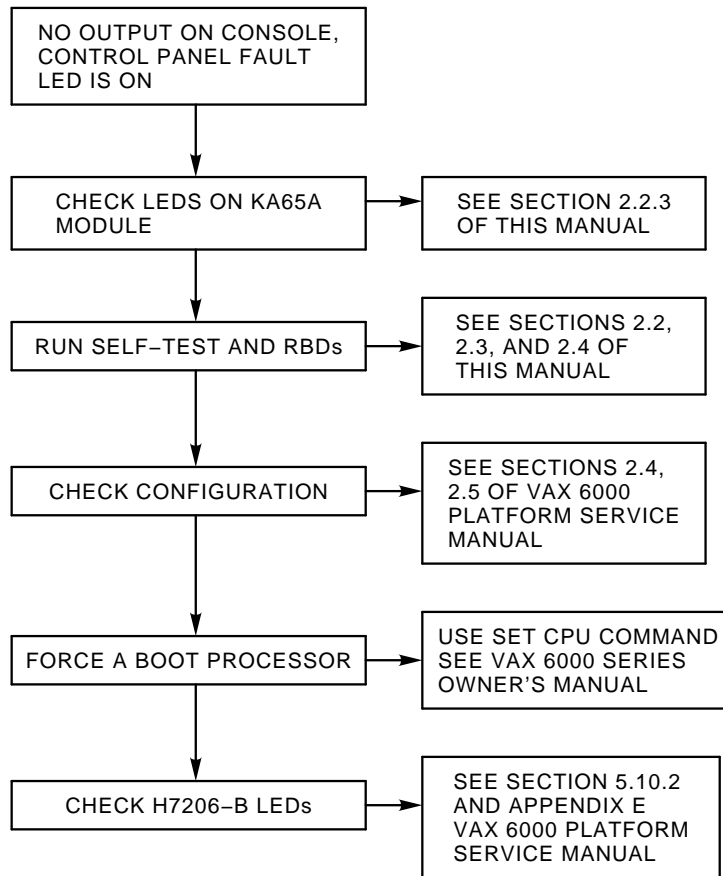
msb-p380r-90

**Figure 1-5: System Shutdown 30 Seconds After Power-Up**



msb-p381-90

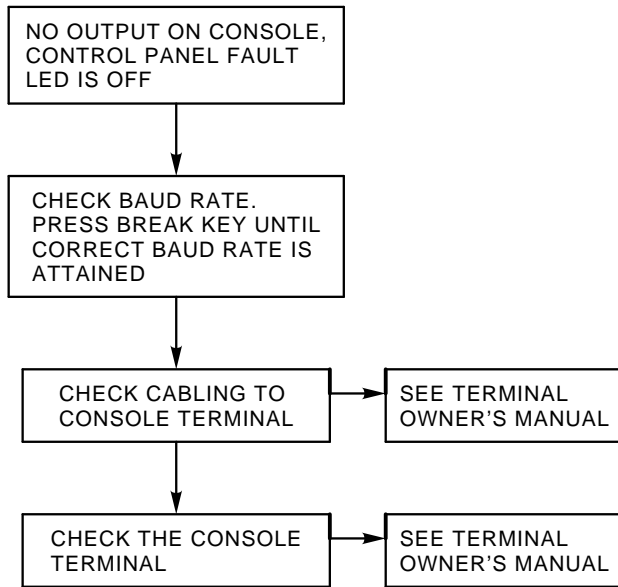
**Figure 1-6: No Console Output, Control Panel Fault LED Is On**



msb-p382-90

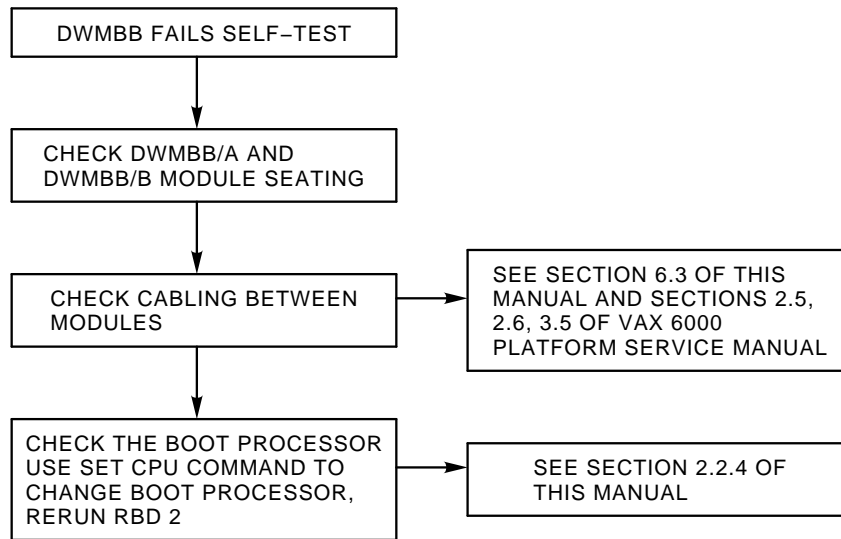


**Figure 1-7: No Console Output, Control Panel Fault LED Is Off**



msb-p383-90

**Figure 1-8: DWMBB Fails Self-Test**



msb-p384-90

## Chapter 2

# Diagnostics

---

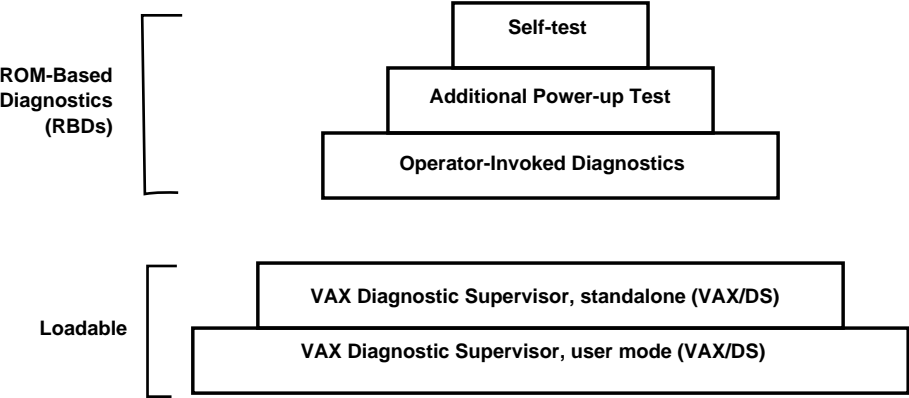
This chapter describes diagnostics for the <REFERENCE>(vax\_mod\_xxx) system. Sections include:

- Diagnostic Overview
- Self-Test and Additional Power-Up Tests
  - Checking Self-Test Results: Console Display
  - Checking Self-Test Results: Status LEDs
  - KA65A LEDs
  - Determining Failing Test from LEDs
  - KA65A Error LED
  - Checking Self-Test Results: XBER and XGPR Registers
- ROM-Based Diagnostic Monitor Program
  - RBD Monitor Control Characters
  - START Command
  - START Command Qualifiers
  - RBD Test Printout, Passing
  - RBD Test Printout, Failing
  - SUMMARY Command
  - Sample RBD Session
  - Running ROM-Based Diagnostics on I/O Devices
- ROM-Based Diagnostics
  - KA65A Self-Test — RBD 0
  - CPU/Memory Interaction Tests — RBD 1
  - DWMBB Tests — RBD 2
  - MS65A Memory Tests — RBD 3
  - KA65A Cache Tests — RBD 4
  - Multiprocessor Tests — RBD 5
- VAX Diagnostic Supervisor Programs
  - Running VAX/DS in Standalone Mode
  - Running VAX/DS in User Mode
  - Sample VAX/DS Session
  - VAX/DS Diagnostics

## 2.1 Diagnostic Overview

The <REFERENCE>(vax\_mod\_XXX) system is tested with two types of diagnostics: ROM-based and loadable. The ROM-based diagnostics (RBD) include self-tests, additional power-up tests, and callable diagnostics (from the RBD monitor). The loadable diagnostics run under the VAX Diagnostic Supervisor (VAX/DS) in standalone mode or user mode (see Figure 2-1).

Figure 2-1: Diagnostics Design



msb-0182-90

**Self-Tests**

Each module on the <REFERENCE>(XMI) bus, except DWMBB and FV64A, has its own self-test resident in ROM. At power-up, initialization, booting, or system reset, each module runs its own self-test. The processor self-test completes within 10 seconds. The memory test completes in less than 60 seconds.

**Additional Power-Up Tests**

Following the modules' self-tests, three additional tests are run and reported in the self-test display: CPU/memory interaction tests, multiprocessor tests, and DWMBB tests.

All CPUs that have passed self-test run the CPU/memory interaction test. The CPU/memory interaction test checks that the processors can access memory. Memory also has a self-test that tests actual memory locations. The CPU/memory interaction test is the second test for memory and serves as a check on the memory's <REFERENCE>(XMI) interface and on some CPU logic that can be tested only by accessing memory.

The multiprocessor test runs after the CPU/memory interaction test. Combined results of the CPU/memory and multiprocessor tests are printed on the ETF line of the self-test display.

If the system contains a VAXBI bus, the DWMBB modules are tested by the boot processor before it queries the VAXBI options for the results of their self-tests. Results from both tests are printed in the XBI lines on the self-test printout.

**Operator-Invoked ROM-Based Diagnostics**

From the console prompt, you can enter RBD mode and run any of six ROM-based diagnostics. These six diagnostics are the KA65A/FV64A self-test, CPU/memory interaction tests, DWMBB tests, memory tests, <REFERENCE>(xmp) cache tests, and multiprocessor tests. In RBD mode, you have the capability of running tests other than those in the default suite, running multiple passes of tests, and receiving an error report with information about any failing tests.

**VAX Diagnostic Supervisor (VAX/DS)**

From the console prompt, you can boot VAX/DS from the compact disk server or other media and run VAX/DS level 3 diagnostics (standalone mode). From your operating system, run VAX/DS and run level 2R diagnostics (user mode). Level 2 VAX/DS diagnostics may be run either in standalone or user mode.

## 2.2 Self-Test and Additional Power-Up Tests

**Self-test and additional power-up tests are ROM-based diagnostics (RBDs) that check each module at power-up, when the system is reset, and during booting. Results can be checked in the console display, the processor module LEDs, and the XBER and XGPR registers.**

**Table 2-1: ROM-Based Diagnostics Run at Power-Up**

<b>RBD Program</b>	<b>Total Tests</b>	<b>Tests Run at Power-Up</b>	<b>Description</b>
0	45 57	45 57	Runs CPU tests (scalar only) Runs CPU tests (scalar and vector)
1	18 22	16 20	Runs CPU/memory interaction tests Runs scalar/vector CPU/memory interaction tests
2	48	39	Runs DWMBB tests
3	14	0	Sizes and runs additional tests on main memory
4	3	0	Checks for cache incoherency
5	7 8	7 7	Runs multiprocessor tests Runs vector multiprocessor tests

Self-test and additional power-up tests are invoked and results are reported under several circumstances:

- At power-up
- When the control panel Restart button is pressed
- During boot procedure
- In console mode, with the systemwide INITIALIZE command

Self-test results are reported in three ways:

- Console display
- Processor module LEDs
- XBER and XGPR registers

Sections 2.2.1, 2.2.2, and 2.2.6 explain how to check self-test results.

The tests run during self-test can be individually invoked in RBD mode using the ROM-based diagnostics monitor program. Here you can examine each test more closely and determine which test is failing.

## 2.2.1 Checking Self-Test Results: Console Display

**You can check self-test results in three ways: the self-test display, the lights on the modules, and the contents of the XBER and XGPR registers.**

### Example 2–1: Self-Test Results

```
#123456789 0123456789 0123456789 0123456789 012345#1
F  E  D  C  B  A  9  8  7  6  5  4  3  2  1  0  NODE #
      A  A  .  .  .  M  M  M  M  .  .  P  P  P      TYP 2
      O  +  .  .  .  +  +  +  +  .  .  +  +  +      STF 3
      .  .  .  .  .  .  .  .  .  .  .  .  E  D  B      BPD 4
      .  .  .  .  .  .  .  .  .  .  .  .  +  +  -      ETF 5
      .  .  .  .  .  .  .  .  .  .  .  .  B  D  E      BPD 6
.  .  .  .  .  .  .  +  +  +  .  +  .  .  +  .  XBI E + 7
      .  .  .  .  .  A4 A3 A2 A1 .  .  .  .  .  .  ILV
      .  .  .  .  .  64 64 64 64 .  .  .  .  .  .  256 Mb
Console = V1.00  RBDs = V1.00  EEPROM = 1.00/1.00  SN = SG01234567 8
>>>
```

- 1** The first line of the self-test printout is the progress trace. This line prints if a <REFERENCE>(xmp) module is in slot 1. The progress trace has two purposes: to give a visual indication that the system is functioning during self-test, and, if self-test fails, to display the failing test number. The numbers correspond to the 45 tests in the <REFERENCE>(xmp) self-test (57 tests if the <REFERENCE>(xmp) in slot 1 is part of a scalar/vector pair; see Chapter 4). When self-test passes, the line prints as in Example 2–1. If a test fails, the failing test number is the last one printed. For example, if test 14 fails, the line is printed as follows:

```
#123456789 01234
```

- 2** This line indicates the type (TYP) of module at each <REFERENCE>(XMI) node. Processors are type P, memories are type M, and I/O adapters are type A. In this example, processors are at nodes 1, 2, and 3, memories at nodes 6 through 9, and I/O adapters in nodes D and E.



- ③ This line shows self-test fail status (STF), which are the results of on-board self-test. Possible values for processors are:
- + (pass)
  - (fail)

All processors passed self-test in this example.

- ④ The BPD line indicates boot processor designation. When the system completes on-board self-test, the processor with the lowest XMI ID number that passes self-test and is eligible is selected as boot processor — in this example, the processor at node 1.

The results on the BPD line indicate:

- The boot processor (B)
  - Processors eligible to become the boot processor (E)
  - Processors ineligible to become the boot processor (D)
- ⑤ During extended test (ETF) all processors run additional tests, which include CPU/memory interaction and multiprocessor tests. On line ETF, results are reported for each processor in the same way as on line STF—a plus sign (+) indicates that extended test passed and a minus sign that extended test failed. In this example, the processor at node 1 (originally selected boot processor) failed the CPU/memory interaction tests.
- ⑥ Another BPD line is displayed, because it is possible for a different CPU to be designated boot processor before the system actually boots. This occurs in this example, because the processor at node 1 failed the extended test. The lowest-numbered processor that passed both tests is the processor at node 2. However, a previous SET CPU/NOPRIMARY command has made this processor ineligible to be boot processor (indicated by the designation D on the BPD line). Therefore, the processor at node 3 is designated boot processor.
- ⑦ A plus sign at the right of the XBI line means both the DWMBB/A and DWMBB/B modules passed self-test. Plus signs within the line mean VAXBI options at those nodes passed their self-tests.
- ⑧ The bottom line of the self-test display shows the ROM and EEPROM version numbers and the system serial number.

## 2.2.2 Checking Self-Test Results: Status LEDs

**You can check self-test results in the console display, in the lights on the modules, or in the XBER and XGPR registers. Before module status LEDs can be checked, the control panel switch must be set to Enable.**

Figure 2-2: Location of Status LEDs

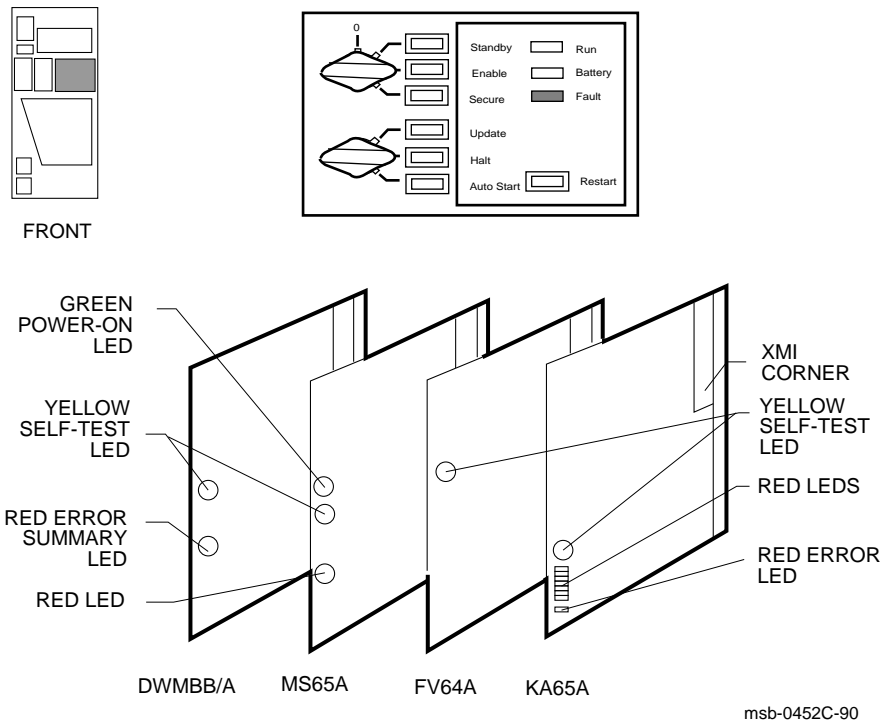


Table 2–2 lists each module’s LED status indicating self-test passed or self-test failed.

**Table 2–2: Status LEDs After Self-Test**

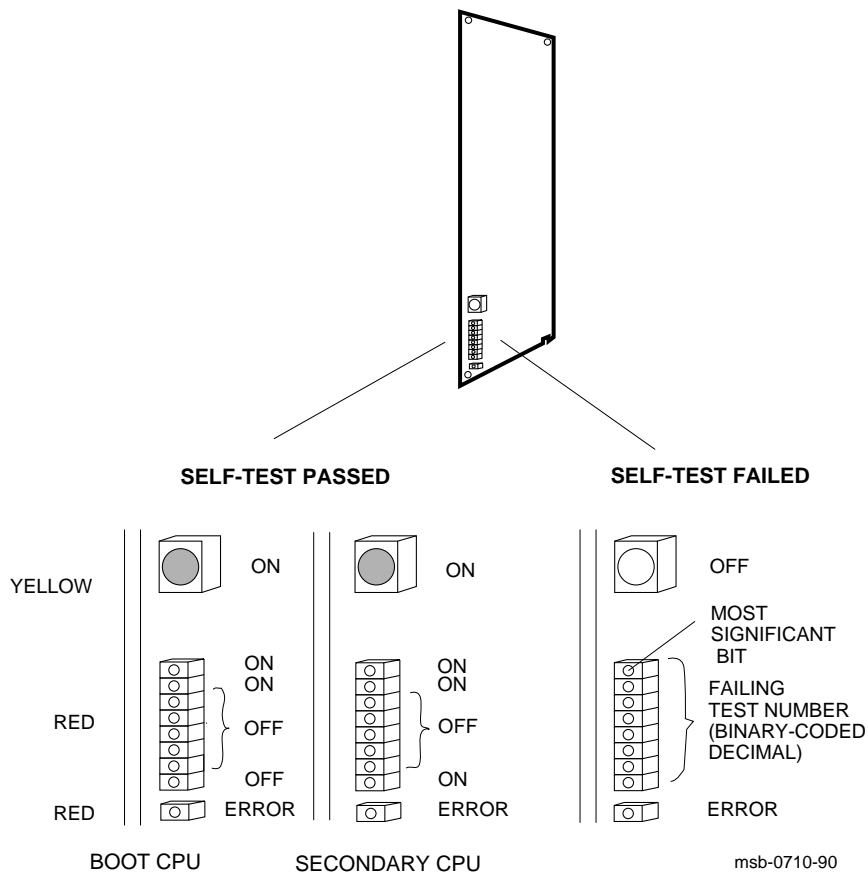
<b>Module</b>	<b>Self-Test Passed</b>	<b>Self-Test Failed</b>
KA65A boot processor	Yellow ON Top two red ON	Yellow OFF Some red ON
FV64A vector processor	Yellow ON	Yellow OFF
KA65A secondary processor	Yellow ON Top two and second from bottom red ON	Yellow OFF Some red ON
MS65A memory	Yellow ON Green ON	Yellow ON <sup>1</sup> Green ON
DWMBB/A adapter	Yellow ON	Yellow OFF

<sup>1</sup>The yellow LED on the memory module is used to indicate *only* that self-test has completed.

### 2.2.3 KA65A LEDs

The KA65A LEDs indicate if self-test passed or failed, if the module is the boot processor or a secondary, and the failing test number.

Figure 2-3: <REFERENCE>(xmp) LEDs After Power-Up Self-Test



If self-test passes, the large yellow LED at the top of the LEDs is ON. (Here self-test means the on-board power-up tests, RBD 0, the CPU/memory interaction tests, RBD 1, and the multiprocessing tests, RBD 5.) The top two red LEDs (next to the yellow one) are also ON, and the next five red LEDs are OFF. The LED next to the Error LED is OFF if the processor is the boot processor, and ON if it is a secondary processor.

If self-test fails, the yellow LED is OFF, and the top eight red LEDs contain an error code that corresponds to the number of the failing test. The test number is represented in binary-coded decimal, with the most significant bit at the top. A bit is ONE if the light is ON.

For example, assume a processor fails self-test (yellow LED is OFF) and shows the following pattern in the top eight red LEDs:

```

TOP
(MSB) off    0
      off    0 = 3
      on     1
      on     1
      off    0
      off    0
      on     1 = 2
(LSB) off    0
BOTTOM

```

The failing test number decodes to 0011 0010 (binary-coded decimal 32). If you then ran RBD 0 with the /TR and /HE qualifiers, the last test number you would see displayed is T0032.

If either of the top two red LEDs is off, a failure has occurred during the self-test sequence. But system power-up self-test actually comprises four sets of tests: <REFERENCE>(xmp) power-up tests (RBD 0), CPU/memory interaction tests (RBD 1), DWMBB tests (RBD 2), and multiprocessing tests (RBD 5). Interpretation of the red LEDs depends on which set of tests was running, as explained in Section 2.2.4. Interpretation of the error LED is explained in Section 2.2.5. Appendix C shows LED patterns that indicate console errors.

## 2.2.4 Determining Failing Test from LEDs

**When self-test fails, the red LEDs on the KA65A contain the failing test number. Check the state of other module LEDs to determine the failing RBD and device, as shown in Table 2-3.**

**Table 2-3: <REFERENCE>(xmp) Status LEDs**

KA65A LEDs		Diagnostic and Test Number	Device Failing	Self-Test Line
Yellow	Red			
<b>FV64A Yellow LED OFF</b>				
OFF	1-45	Power-up self-test (RBD 0) T0001-T0045 See Table 2-12.	<REFERENCE>(xmp)STF	
ON	46-57	Power-up self-test (RBD 0) T0046-T0057	FV64A	STF
OFF	61-82	CPU/memory test - Memory 1 (RBD 1) T0001-T0022 See Table 2-13.	<REFERENCE>(xmp)ETF <REFERENCE>(xrv), or <REFERENCE>(xma) 1 (module with low- est XMI node num- ber)	
ON	79-82	CPU/memory test - Memory 1 (RBD 1) T0079-T0082	FV64A	ETF
OFF	83	CPU/memory test - Memory 2 T0003 (equivalent to ST1/T=3)	<REFERENCE>(xma)ETF 2	
OFF	84	CPU/memory test - Memory 3	MS65A 3	ETF
OFF	85	CPU/memory test - Memory 4	MS65A 4	ETF
OFF	86	CPU/memory test - Memory 5	MS65A 5	ETF
OFF	87	CPU/memory test - Memory 6	MS65A 6	ETF
OFF	88	CPU/memory test - Memory 7	MS65A 7	ETF
OFF	89	CPU/memory test - Memory 8	MS65A 8	ETF

**Table 2-3 (Cont.): <REFERENCE>(xmp) Status LEDs**

<b>KA65A LEDs</b>		<b>Diagnostic and</b>	<b>Self-Test</b>	
<b>Yellow</b>	<b>Red</b>	<b>Test Number</b>	<b>Device Failing</b>	<b>Line</b>
<b>FV64A Yellow LED ON</b>				
OFF	91-97	Multiprocessor test (RBD 5) T0001-T0007 See Table 2-18.	<REFERENCE>(xmp)ETF or <REFERENCE>(xrv)	
<b>DWMBB/A Yellow LED OFF</b>				
ON	1-48	DWMBB test (RBD 2) T0001-T0048 See Table 2-14.	DWMBB	XBI

If a processor's yellow LED is OFF and the red LEDs show an error code in the range 1-45, the power-up self-test failed and the processor board is bad. If the processor board has an attached vector processor and the error code is 46-57, the vector processor board is bad. On the self-test console display, the processor shows a minus sign (-) on the STF line.

After the power-up tests, each processor runs the CPU/memory interaction tests and the multiprocessor tests. If a test fails, the processor shows a minus sign (-) on the ETF line of the self-test console display. The LED error codes are numbered from 61 to 82, which is the failing CPU/memory test number (1 through 22) plus 60, or 91 to 97, which is the failing multiprocessor test number (1 through 7) plus 90.

Each processor, after testing with the first memory, runs the CPU/memory interaction tests on every other good memory module. (However, only CPU/memory interaction test T0003 is run.) If a failure occurs, the memory module is probably bad, although the processor's yellow light is OFF and the memory module's yellow light is ON. If several processors fail on the same memory, that memory module is certainly bad. Try using SET MEMORY to configure the bad module out of the interleave set. For error codes 83 through 89, consult Table 2-3 to determine the failing memory.

The last series is the DWMBB tests. If one fails, the red LEDs contain an error code, although the processor's yellow self-test LED is ON (because the CPU itself has passed). The failing test numbers are listed in Table 2-14. Note that only the boot processor performs the DWMBB tests.

## 2.2.5 KA65A Error LED

**The KA65A error LED (bottom red LED) is an error summary or indicates the state of backup cache.**

**Table 2-4: KA65A Error LED**

<b>XCR0&lt;14&gt;</b>	<b>XCR0&lt;13&gt;</b>	<b>Error LED ON</b>
0	0	Error summary (power-up state)
0	1	Backup cache is off
1	0	Error summary or backup cache is off
1	1	Off—Error LED does not light

### **Example 2-2: Examining the XCR0 and XBER Registers**

```
>>> E/P/L E1880024      ! Examine the XCR0 register of the KA65A
E1880024 00000200      ! processor in slot 1. The result shows
                        ! bits 14 and 13 are clear.
>>> E/P/L E1880004      ! Then examine the XBER register of the
E1880004 8000A300      ! same processor. The result shows bit
                        ! 31 is set.
```



The state of the error LED depends on bits in two registers. At power-up it reflects the state of XBER bit 31, the Error Summary bit. (The error LED is ON if the Error Summary bit is set.)

At times other than power-up, the meaning of the error LED is determined by the state of XCR0 bits 14 and 13. See Table 2-4 and Example 2-2.

In Example 2-2 the error LED on the KA65A processor in slot 1 is lit. The reason for this is determined by examining the XCR0 and XBER registers. The addresses for these registers are calculated by adding an offset to the base address of the slot that contains the KA65A module. The offset for XCR0 is 24; for XBER it is 4. (All numbers are hexadecimal.) See Table 2-5 for the base addresses of the XMI slots.

## 2.2.6 Checking Self-Test Results: XBER and XGPR Registers

**You can check self-test results in the self-test display, in the lights on the modules, or in the XBER and XGPR registers. Use the XBER and XGPR registers when a failure occurs during power-up and the failing test number cannot be found in the module LEDs.**

### Example 2-3: XGPR Register After Power-Up Test Failure

```
>>> E/P/L E190000C      ! Examine the longword at physical address
E190000C 30xxxxxxx     ! E190000C, the address of the XGPR
                        ! register of the <REFERENCE>(xmp) processor in slot 2.
>>> E/P/L E1900004     ! Then examine the XBER register (bit 10
E1900004 xxxxx4xx     ! set). The result indicates that test 30
                        ! of the <REFERENCE>(xmp) self-test failed. See
                        ! Table 2-6 to interpret the data returned.
>>> E/P/L E188000C     ! Examine the XGPR register of the <REFERENCE>(xmp)
E188000C 13xxxxxx     ! processor in slot 1. Derivation of the
                        ! address is explained below.
>>> E/P/L E1880004     ! Then examine the XBER register (bit 10
E1880004 xxxxx0xx     ! clear). DWMBB test 13 failed.
```

When a failure occurs in power-up test, you can examine the XGPR register to determine the failing test number. The XGPR register of the <REFERENCE>(xmp) processor that failed self-test or CPU/memory interaction test, or of the boot processor if DWMBB test failed, contains the failing test number. If all power-up tests pass, the XGPR register contains other data and should be ignored.

To examine the XGPR register, first see Table 2-5 to determine the base address (BB) of the <REFERENCE>(xmp) processor's node. Then calculate the address of the XGPR register by adding 0C (hex) to the base address.

The failing test number is derived from the upper byte (bits <31:24>) of the longword returned. For self-test, the upper byte contains the failing test number. If a CPU/memory interaction test fails, this byte contains the failing test number plus 60. If a multiprocessor test fails, this byte contains

the failing test number plus 90. All numbers are expressed in binary-coded decimal (BCD). See Table 2–6.

**Table 2–5: XMI Base Addresses**

Slot	Node	Base Address (BB)
1	1	E188 0000
2	2	E190 0000
3	3	E198 0000
4	4	E1A0 0000
5	5	E1A8 0000
6	6	E1B0 0000
7	7	E1B8 0000
8	8	E1C0 0000
9	9	E1C8 0000
10	A	E1D0 0000
11	B	E1D8 0000
12	C	E1E0 0000
13	D	E1E8 0000
14	E	E1F0 0000

**Table 2–6: Interpreting XGPR Failing Test Numbers**

Failing Diagnostic	XBER <10>	XGPR <31:24> (BCD)	Test Numbers
Self-test	Set	1–57	1–57
CPU/memory interaction test	Clear	61–82	1–22
Additional memory	Clear	83–89	3
Multiprocessor test	Clear	91–97	1–7
DWMBB test	Clear	1–48	1–48

## 2.3 ROM-Based Diagnostic Monitor Program

**Access the ROM-Based Diagnostic Monitor program through the console program. Type T/R at the console prompt to enter RBD mode. RBD mode has three commands with qualifiers that run the RBD tests. It also has a set of control characters, described in Section 2.3.1.**

**Table 2-7: RBD Monitor Commands**

<b>Command</b>	<b>Function</b>
ST[ART] <i>n</i>	Starts RBD <i>n</i> , where <i>n</i> is the number of the RBD program listed in Table 2-11
SU[MMARY]	Prints a summary report of the last RBD program run
QU[IT]	Exits the RBD monitor and returns control to the console program

**Table 2-8: ROM-Based Diagnostic Programs — Callable Tests**

<b>RBD Program</b>	<b>Total Tests</b>	<b>Default Number of Callable Tests</b>	<b>Description</b>
0	45 57	45 57	Runs CPU tests (scalar only) Runs CPU tests (scalar and vector)
1	18 22	18 22	Runs CPU/memory interaction tests Runs scalar/vector CPU/memory interaction tests
2	48	48	Runs DWMBB tests
3	14	10	Sizes and runs additional tests on main memory
4	3	0	Checks for cache incoherency
5	7 8	7 7	Runs multiprocessor tests Runs vector multiprocessor tests

To enter the RBD monitor, at the console prompt type:

```
>>> T/R          ! This is the abbreviation for TEST/RBD.
                !
RBDn>           ! RBD prompt appears signifying entrance into
                ! RBD mode, where n is the XMI node number of
                ! the processor running the RBD monitor program.
```

The RBD commands are explained in this section. Table 2-7 gives the commands, their abbreviations, and functions.

Six programs run from the ROM-based diagnostics (RBD) monitor program. These programs are the <REFERENCE>(XMP) and <REFERENCE>(xrv) self-tests, CPU/memory interaction tests, DWMBB tests, memory tests, <REFERENCE>(xmp) cache tests, and multiprocessor tests. Each of these programs has several tests, as shown in Table 2-8. The RBDs are designed for use by Digital customer service personnel.

Each RBD has a default number of tests that run at power-up, and another default number of tests that run when the program is called from the RBD monitor (see Table 2-8). The CPU diagnostic (RBD 0) runs all its tests in both modes. The power-up default for the CPU/memory interaction diagnostic (RBD 1) is 16 (scalar only—20 with the vector tests), and the callable default is all tests. RBD 3, the Memory diagnostic, does not run on power-up. In callable mode, 10 of the 14 tests run when invoked; tests 1 through 10 are defaults. RBD 4, the test for cache incoherency, also does not run on power-up. In callable mode, a test number must be supplied to run any test. The multiprocessing diagnostic (RBD 5) runs seven tests in both modes. To run tests other than the default suite from the RBD monitor, issue a command such as the following, which invokes all memory tests (RBD 3):

```
RBDn> ST3/T=1:14
```

It is helpful to use the trace qualifier, /TR, with the RBD START command. (See Section 2.3.3.) This qualifier shows each individual test as it is run. If a test fails, the program displays error messages. By default, the RBDs continue testing after an error is encountered. Adding the halt-on-error qualifier, /HE, causes the program to halt when the first error is encountered. Testing can be aborted at any time by typing CTRL/C.

To exit RBD mode, type QUIT at the RBD prompt. Your next prompt is from console mode.

### 2.3.1 RBD Monitor Control Characters

**Several control characters are supported by the RBD monitor program. These characters manage the program process as shown in Table 2-9.**

**Table 2-9: RBD Monitor Control Characters**

<b>Character</b>	<b>Environment</b>	<b>Function</b>
<code>CTRL/C</code>	Test running	Stops the execution of an RBD test and executes cleanup code.
<code>DELETE</code>	RBD command line	Use for deleting erroneous characters entered on the command line.
<code>CTRL/Q</code>	Test running	Resumes output to terminal that was suspended with <code>CTRL/S</code> .
<code>CTRL/R</code>	At RBD prompt	Refreshes the command line; useful when characters are deleted.
<code>CTRL/S</code>	Test running	Suspends output to the terminal until <code>CTRL/Q</code> is typed.
<code>CTRL/T</code>	Test running	Displays informational status line about currently running diagnostic.
<code>CTRL/U</code>	At RBD prompt	Disregards previous input.
<code>CTRL/Y</code>	Test running	Stops the execution of an RBD test and does not execute any cleanup code.
<code>CTRL/Z</code>	At RBD prompt	Exits RBD monitor program and enters console program; same effect as the QUIT command.

When CTRL/C is entered from the console terminal that began execution of the RBD test, the diagnostic stops execution, runs cleanup code, and returns control to the RBD monitor program. This happens immediately when running RBD 0, RBD 1, or RBD 2; there may be a wait of up to one minute for a response when RBD 3 is running. If CTRL/C is typed at the RBD monitor prompt, it has the same effect as CTRL/U.

When you use the DELETE key (or rubout key), characters being deleted are preceded by a backslash ( \ ) and print as they are rubbed out. When the next valid character is typed, it is preceded by a backslash ( \ ) to delineate the deleted characters. You can use CTRL/R to refresh the line.

When a CTRL/T is received by the RBD monitor program from the console terminal that began execution of the RBD test, the diagnostic displays an informational status line and continues test execution. A CTRL/T entered at the RBD prompt is ignored.

When the RBD monitor program receives a CTRL/U, the program disregards all previous input typed and returns the RBD prompt. If a test is running when CTRL/U is entered, CTRL/U is ignored.

When a CTRL/Y is received by the RBD monitor program from the console terminal that began execution of the RBD test, the diagnostic stops execution and returns control to the RBD monitor program. No cleanup code is run, and the unit under test is left in an indeterminate state. A CTRL/Y entered at the RBD monitor prompt has the same effect as CTRL/U.

When the RBD monitor program receives a CTRL/Z, the program exits and control is returned to the console program. The next prompt is the console prompt. CTRL/Z has the same effect as the QUIT command. If CTRL/Z is entered while an RBD test is running, CTRL/Z has the same effect as CTRL/C: it halts the test and executes cleanup code.

## 2.3.2 START Command

**The RBD monitor START command invokes a specific RBD program. It takes an argument indicating the RBD program to be run, and can take any of 13 qualifiers.**

### Example 2-4: START Command

```
>>> T/R                ! Command to enter RBD monitor program.
RBD3>                  ! RBD monitor prompt, where 3 is the hexa-
                        ! decimal node number of the processor
                        ! that is currently receiving your input.
RBD3> ST0/TR           ! Runs the CPU tests, testing the <REFERENCE>(XMP)
                        ! at <REFERENCE>(XMI) node number 3. Test results
                        ! are written to the console terminal.
RBD3> ST1/HE/IE/BE     ! Runs the default tests in the CPU/memory
                        ! interaction RBD, halting on the first
                        ! error encountered, inhibiting error output,
                        ! ringing the bell when the first error is
                        ! encountered.
```

The START command syntax is:

```
STn[/qualifier] [parameter]
```

where:

- *n* is the RBD to be run (see Table 2-11).
- [/qualifier] is one of those listed in Section 2.3.3.
- [parameter] is a program-specific value used in RBD 3. (For the meaning of this parameter, see Section 2.4.5.)



### 2.3.3 START Command Qualifiers

**The START command has qualifiers that allow you to control the output of the tests—to run portions of a test, to run nondefault tests, and to loop on tests.**

**Table 2–10: START Command Qualifiers**

Qualifier	Default	Function
/BE	Disabled	Bell sounds when an error is encountered
/C	Disabled	Destructive test confirmation
/DS	Disabled	Disable status reports
/HE	Disabled	Halt on the test that incurs a hard error
/HS	Disabled	Halt on the test that incurs a soft error
/IE	Disabled	Inhibit all error output
/IS	Disabled	Inhibit summary reports
/LE	Disabled	Loop on the test that incurs a hard error
/LS	Disabled	Loop on the test that incurs a soft error
/P= <i>n</i>	Enabled	Make <i>n</i> passes of the test or tests indicated
/QV	Disabled	Quick verify mode
/T= <i>n</i> [: <i>m</i> ]	Enabled	/T= <i>n</i> runs test <i>n</i> ; /T= <i>n</i> : <i>m</i> runs a range of tests from <i>n</i> through <i>m</i>
/TR	Disabled	Print a trace of test numbers, as they run

**NOTE:** *A qualifier is valid only for the command with which it is issued. Qualifiers do not remain in effect for the session once they are issued.*

See Example 2–4 for examples and a description of the START command syntax.

With /BE, the RBD monitor program rings the bell on the console terminal whenever an error is encountered. This is useful when error printout is inhibited and a loop is being performed on an intermittent error (/LE).

/C enables execution of destructive tests. See Section 2.4.5 for information on the destructive tests.

`/DS` disables printout of the diagnostics test results. The summary report is run, unless it is specifically disabled.

`/HE` halts on hard error and stops execution of tests as soon as the first hard error is encountered. (In this context, a hard error is defined as a recoverable, repeatable error, for example, a ROM checksum error. This differs from a fatal error, which is an unrecoverable fault, for example, an unexpected interrupt or exception. A fatal error is always cause for program abortion, regardless of the state of the `/HE` or `/LE` qualifier.) The test number is printed, and a summary indicating failure of the RBD is printed to the console terminal. Also the RBD monitor prompt is returned. Continue on error is the default condition, so if you want to halt on error, you must specifically invoke it in your command line.

`/HS` halts on soft error and stops execution of tests as soon as the first soft error is encountered. (In this context, a soft error is defined as a recoverable error that goes away after retry, for example, a corrected read data memory error.) The test number is printed, and a summary indicating failure of the RBD is printed to the console terminal. Also the RBD monitor prompt is returned. Continue on soft error is the default condition, so if you want to halt on soft error, you must specifically invoke it in your command line.

`/IE` inhibits all error output, suppressing printing of RBD results. This qualifier is used primarily for module repair, in conjunction with the `/LE` or `/LS` qualifier. Errors are counted even when the printing is disabled.

`/IS` suppresses printout of RBD summary after the end of the last pass performed by the RBD.

`/LE` loops on the test where the first hard error is detected. Even if the error is intermittent, looping continues on the test indicated. To terminate `/LE`, enter `CTRL/C`, `CTRL/Z`, or `CTRL/Y`. After entering one of these control characters, a summary report is printed. A fatal error causes the program to abort, regardless of the state of this qualifier.

`/LS` loops on the test where the first soft error is detected. Even if the error is intermittent, looping continues on the test indicated. To terminate `/LS`, enter `CTRL/C`, `CTRL/Z`, or `CTRL/Y`. After entering one of these control characters, a summary report is printed.

`/P=n` runs *n* number of passes of the RBD test invoked, where *n* is a decimal number. If *n* is 0, all selected tests run for an infinite number of passes. If the `/P` qualifier is not used, the program defaults to one pass of the test invoked. When used with the `/T=n.m` qualifier, you run a range of tests. To terminate `/P=n`, enter `CTRL/C`, `CTRL/Z`, or `CTRL/Y`. After entering one of these control characters, a summary report prints out and the RBD monitor prompt returns.

`/QV` selects the quick verify version of any selected test that supports this mode.

`/T=n[:m]` selects individual tests (`/T=n`) or a range of tests (`/T=n:m`) where  $n$  and  $m$  are decimal numbers. For example, to run tests T0005 through T0008, use `/T=5:8`. If no `/T` qualifier is used, the diagnostic runs its default suite of tests.

`/TR` prints each test number as it is completed. This qualifier allows you to trace the progress of the diagnostic as it runs. Without the `/TR` qualifier, just the summary line is printed.

One **parameter field** can be appended to the `START` command string to control aspects of the diagnostic that are not covered by the qualifiers. The parameter must be appended after any qualifiers specified and separated from the qualifiers by a space. The format of the parameter field is one to four hexadecimal characters. The use of a parameter field is implementation specific and is optional.

## 2.3.4 RBD Test Printout, Passing

**The RBD printout results are different when the RBD tests pass and when they fail. Example 2-5 shows a passing printout, and Example 2-6 is a sample failure printout.**

### Example 2-5: RBD Test Printout, Passing

```
>>> T/R                               ! Command to enter RBD monitor program at
                                       ! console prompt.
RBD3>                                  ! RBD monitor prompt, where 3 is the hexa-
                                       ! decimal node number of the processor
                                       ! that is currently receiving your input.
RBD3> ST2/TR E                          ! Runs the XBI self-test, testing the <REFERENCE>(XBI)
                                       ! at <REFERENCE>(XMI) node number E. Test results
                                       ! written to the console terminal:

; XBIP_TST1                            1.002
; T0001 T0002 T0003 T0004 T0005 T0006 T0007 T0008 T0009 T00103
; T0011 T0012 T0013 T0014 T0015 T0016 T0017 T0018 T0019 T0020
; T0021 T0022 T0023 T0024 T0025 T0026 T0027 T0028 T0029 T0030
; T0031 T0032 T0033 T0034 T0035 T0036 T0037 T0038 T0039 T0040
; T0041 T0042 T0043 T0044 T0045 T0046 T0047 T0048
;          P4      35      80806      17
; 00000000 00000000 00000000 00000000 00000000 00000000 000000008

RBD3> QU9                               ! RBD prompt returns; test ran successfully.
                                       ! Exit RBD program.
>>>
```

The callouts in Example 2-5 are explained below.

- <sup>1</sup> This entry designates which test is being run. Here it is the XBIP\_TST, the self-test for the DWMBB.

XMP/V\_ST indicates RBD 0, the CPU tests  
CPUMEM indicates RBD 1, the CPU/memory interaction tests  
XBIP\_TST indicates RBD 2, the DWMBB tests  
XMA2\_RBD indicates RBD 3, the Memory tests  
XMP\_BC indicates RBD 4, the cache tests  
XMP\_MP indicates RBD 5, the multiprocessor tests

- <sup>2</sup> This field lists the revision number of the RBD program.

- ③ These T00nn fields appear only with the /TR qualifier; each entry corresponds to a test being run and prints out as the test starts running. In a passing RBD, the final T00nn number corresponds to the last test run.
- ④ This field indicates whether the RBD passed or failed; P for passed, F for failed.
- ⑤ This field is the XMI node number of the boot processor executing the RBD. It matches the number in your RBD prompt.
- ⑥ This field is always 8080—the device type of the boot processor.
- ⑦ This field displays the total number of passes (in decimal) executed by the RBD. The default number of passes is 1. If you use the START command with the qualifier /P=5, for example, then this field will show 5, indicating 5 passes were completed.
- ⑧ This line contains the summary of the RBD failures. In a successful RBD run, the line will contain all zeros as shown here. Currently only the second and third fields are used. The second field contains the number of hard errors detected during the run. The third field contains the number of soft errors detected during the run.
- ⑨ The console prompt is usually returned in response to the RBD QUIT command, as shown in this example. However, when tests that cause parity errors are run, the response to QUIT is a system reset. Self-test is then run, and the self-test results are printed. The tests that cause a system reset are: tests 1, 2, 4, and 22 of RBD 1; tests 2, 3, 4, 30, and 31 of RBD 2; and tests 5 and 9 of RBD 3.

## 2.3.5 RBD Test Printout, Failing

**The RBD printout results are different when the RBD passes and when it fails. Example 2-6 is a sample failure printout, and Example 2-5 shows a passing printout.**

### Example 2-6: RBD Test Printout, Failing

```
>>> T/R                                     ! Command to enter RBD monitor program at
                                           ! console prompt.
RBD2>                                       ! RBD monitor prompt, where 2 is the hexa-
                                           ! decimal node number of the processor
                                           ! that is currently receiving your input.
RBD2> ST0/TR                               ! Execute RBD 0 (CPU test) and trace results.

; XMP/V_ST      1.00

; T0001 T0002 T0003 T0004 T0005 T0006 T0007 T0008 T0009 T0010
; T0011 T0012 T0013 T0014 T0015 T0016 T0017 T0018 T0019 T0020
; T0021 T0022 T0023 T0024 T0025 T0026 T0027 T0028 T0029 ❶

;      F ❷      2 ❸      8080 ❹      1 ❺
;      HE ❻ MAXMIRAM ❼      XX ❽      T0029 ❾
;      28 ❿ 5555AAAA ⓫ A8AAAAAA ⓬ 00000000 ⓭ E1008000 ⓮ E008C410 ⓯ 08 ⓰
; T0030 T0031 T0032 T0033 T0034 T0035 T0036 T0037 T0038 T0039
; T0040 T0041 T0042 T0043 T0044 T0045 ⓱

;      F      1      8080      1 ⓲
; 00000000 00000001 ⓳ 00000000 00000000 00000000 00000000 00000000

RBD2>                                       ! RBD prompt returns; test completed.
RBD2> QUIT                               ! Exit RBD program.
>>>                                       ! Console prompt reappears.
```

The callouts in Example 2-6 are explained below. (See also Example 2-5 for explanation of other fields of the printout.)

- ❶ These T00nn fields appear only with the /TR qualifier; each entry corresponds to a test being run. The entry prints out as the test starts running. This T00nn number is the number of the failing test and is followed by a failure report. In this example, test 29 failed. The /HE qualifier was not used, so testing continues.
- ❷ F indicates failure of the previous test listed, test 29.
- ❸ This field is the XMI node number of the boot processor executing the RBD. It matches the number in your RBD prompt.

- ④ This field is always 8080—the device type of the boot processor.
- ⑤ This field displays the total number of passes (in decimal) executed by the RBD. The default number of passes is 1.
- ⑥ The class of error is displayed here. *HE* indicates that the error was a hard error. *SE* means the error was a soft error, and *FE* indicates a fatal error. (See Section 2.3.3 for a definition of these errors.)
- ⑦ This field describes the failing logic. Here, the XMI interface chip is the failing logic.
- ⑧ This field is the unit number used in memory, multiprocessing, and DWMBB tests.
- ⑨ This field lists the number of the test that failed; test 29 failed here.
- ⑩ This is a two-digit (decimal) generic error code.
- ⑪ The expected data is listed here. 5555AAAA is the data test 29 expected.
- ⑫ The received data is listed here. A8AAAAAA is the data test 29 received.
- ⑬ This field shows any unexpected interrupt vectors.
- ⑭ This is the address in memory where the referenced error is found.
- ⑮ This is the address of the failing PC at the time of error.
- ⑯ This is the error number within the failing test. In this example, the failure was detected at failure point 8 in T0029. This is a decimal field.
- ⑰ This final T00*nn* number corresponds to the last test run.
- ⑱ This entire line is the summary line, and a repeat of the failure summary. It lists the pass/fail code (P or F), the node number and device type number of the boot processor executing the RBD, and the number of passes of the RBD.
- ⑲ This is the number of hard errors detected.

### 2.3.6 SUMMARY Command

**The RBD monitor SUMMARY command displays a summary of the last diagnostic run.**

#### Example 2-7: SUMMARY Command

```
>>> T/R                                ! Command to enter RBD monitor program
RBD1> ST0/IE/IS/P=100                  ! Execute RBD 0 (CPU test), inhibiting
                                        ! error outputs and summary report.

; XMP/V_ST      1.00

RBD1> SU                                ! Request a summary.

; XMP/V_ST      1.00

;          P1          12      80803          1004
; 00000000 00000000 00000000 00000000 00000000 00000000 000000005

RBD1>
```



The callouts in Example 2-7 are explained below.

- ❶ This field indicates whether the RBD passed or failed; P for passed, F for failed.
- ❷ This field is the XMI node number of the boot processor executing the RBD. It will match the number in your RBD prompt, which also indicates the node number of your boot processor.
- ❸ This field is the device type number of the boot processor executing the RBD.
- ❹ This field displays the total number of passes executed by the RBD.
- ❺ This line contains the summary of the RBD failures. Presently only the second and third fields are used. The second field contains the number of hard errors detected during the run. The third field contains the number of soft errors detected during the run.

## 2.3.7 Sample RBD Session

**Examples 2-8, 2-9, and 2-10 show a sample RBD session.**

### Example 2-8: Sample RBD Session, Part 1 of 3

```
>>> T/R ①
RBD1> ST0/TR ②
;XMP/V_ST      1.00
; T0001 T0002 T0003 T0004 T0005 T0006 T0007 T0008 T0009 T0010
; T0011 T0012 T0013 T0014 T0015 T0016 T0017 T0018 T0019 T0020
; T0021 T0022 T0023 T0024 T0025 T0026 T0027 T0028 T0029 T0030
; T0031 T0032 T0033 T0034 T0035 T0036 T0037 T0038 T0039 T0040
; T0041 T0042 T0043 T0044 T0045 T0046 T0047 T0048 T0049 T0050
; T0051 T0052 T0053 T0054 T0055 T0056 T0057
;          P          1      8080          1
;00000000 00000000 00000000 00000000 00000000 00000000 00000000
RBD1> ST1/TR/HE ③
;CPUMEM        1.00
; T0001 T0002 T0003 T0004 T0005 T0006 T0007 T0008 T0009 T0010
; T0011 T0012 T0013 T0014 T0015 T0016 T0017 T0018 T0019 T0020
; T0021 T0022
;          P          1      8080          1
;00000000 00000000 00000000 00000000 00000000 00000000 00000000
RBD1> ST2/TR 5 ④
;XBIP_TST      1.00
;          F          1      8080          1
;          HE NO_UNIT          XX      T0000
;          52 00000000 00000000 00000000 E1880000 E007AF8E 01
;          F          1      8080          1
;00000000 00000001 00000000 00000000 00000000 00000000 00000000
```

- ① Enter RBD mode from console mode. The RBD prompt appears and indicates you are operating from the boot processor at node 1.
- ② Run RBD 0 and trace the tests. The CPU test runs all 57 tests successfully.
- ③ Run RBD 1, trace it, and halt on the first hard error found. All CPU/memory interaction RBD tests run and pass.
- ④ Run RBD 2, testing the DWMBB at XMI node 5. The value NO\_UNIT on the third line of output indicates that the node value of node 5 is not correct; no DWMBB was found at this node.

### Example 2-9: Sample RBD Session, Part 2 of 3

```
RBD1> ST2/TR/T=2:4/P=3 E5
;XBIP_TST      1.00
; T0002 T0003 T0004 T0002 T0003 T0004 T0002 T0003 T0004
;          P          1      8080          3
;00000000 00000000 00000000 00000000 00000000 00000000 00000000
RBD1> ST3/TR/T=16
RBD1> ST3/TR/T=1
RBD1> ST3/TR/T=1 /C7
;XMA2_RBD      0.80
; T0001
;          P          1      8080          1
;00000000 00000000 00000000 00000000 00000000 00000000 00000000
RBD1> ST4/TR/T=18
;XMP_BC        1.00
; T0001
;          S          1      8080          1
;          XX      Cache      XX      T0001
;00000800 00000000 00000000 1F2C0000 00000300 00000000 00000000 00000000
;          P          1      8080          1
;00000000 00000000 00000000 00000000 00000000 00000000 00000000
RBD1> ST5/TR9
;XMP_MP        1.00
; T0001 T0002 T0003 T0004 T0005 T0006 T0007
;          P          1      8080          1
;00000000 00000000 00000000 00000000 00000000 00000000 00000000
RBD1> QUIT10
      [self-test results may be displayed here]
```

- ⑤ Run RBD 2, testing the DWMBB at XMI node E; trace the tests as they run, and run tests 2 through 4 of RBD 2; make 3 passes over these selected tests.

Note that the *T00nn* line lists each of the three tests three times, since the */P=3* called for 3 passes of the tests. And the final parameter in the summary line is a 3, indicating that 3 passes completed.

- ⑥ Run RBD 3, trace it, and run only test 1 of this RBD. This test is one of the memory tests that is not part of the default suite of tests. This test corrupts memory. You must add a */C* qualifier to the *START* command, to indicate that you do indeed intend to run this destructive test.

The */C* qualifier was not given in this example. The command line is echoed, waiting for */C* to be typed.

At this point you can press Return to return to the command prompt (RBD1>), or you can type the */C* qualifier followed by Return.

- ⑦ Run RBD 3, trace the tests as they run, run only test 1, and */C* allows the test to run. In this example, the test completed with no errors.
- ⑧ Run RBD 4, test 1, with trace set.
- ⑨ Run RBD 5 and trace the tests. All tests pass.
- ⑩ Exit from RBD mode and enter console mode. The console prompt is usually returned in response to the *RBD QUIT* command; however, when tests that cause parity errors are run, the response to *QUIT* is a system reset. Self-test is then run, and the self-test results are printed. The tests that cause a system reset are tests 1, 2, 4, and 22 of RBD 1; tests 2, 3, 4, 30, and 31 of RBD 2; and tests 5 and 9 of RBD 3.

### Example 2-10: Sample RBD Session, Part 3 of 3

```
>>> SET CPU 211
>>> T/R

RBD2> ST0/TR12
;XMP/V_ST      1.00

; T0001 T0002 T0003 T0004 T0005 T0006 T0007 T0008 T0009 T0010
; T0011 T0012 T0013 T0014 T0015 T0016 T0017 T0018 T0019 T0020
; T0021 T0022 T0023 T0024 T0025 T0026 T0027 T0028 T0029 T0030
; T0031 T0032 T0033 T0034 T0035 T0036 T0037 T0038 T0039 T0040
; T0041 T0042 T0043 T0044 T0045 T0046 T0047 T0048 T0049 T0050
; T0051 T0052 T0053 T0054 T0055 T0056 T0057

;          P          1      8080          1
;00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

- ⑪ Make another processor the primary processor so that RBD 0 can be run on it.
- ⑫ Run RBD 0 and trace the tests. This processor has an attached vector processor. All 57 tests run successfully.

## 2.3.8 Running ROM-Based Diagnostics on I/O Devices

**Some XMI and VAXBI devices can be tested from the console terminal with their on-board ROM-based diagnostics. The Z console command is used to send commands to these nodes.**

### Example 2-11: Running RBDs on I/O Devices

```
>>> SHOW CONFIGURATION1
      Type      Rev
1+ KA65A (8080) 0006
2+ KA65A (8080) 0006
3+ FV64A (0000) 0000
4+ MS65A (4001) 0084
8+ MS65A (4001) 0084
D+ DEMNA (0C03) 0601
E+ DWMBB/A (2002) 0001

XBI E
1+ DWMBB/B (210F) 000A
4+ KDB50 (010E) 132E
6+ TBK70 (410B) 0307
8+ CIBCA-B (0108) 41C2
C+ DEBNI (0118) 0200

>>> Z D2
?0033 Z connection successfully started
t/r3

RBDD> ST0/TR4

;Selftest      3.00

; T0001 T0002 T0003 T0004 T0005 T0006 T0007 T0008 T0009 T0010
; T0011 T0012 T0013 T0014 T0015 T0016 T0017 T0018

;      P      D      0C03      1
;00000000 00000000 00000000 00000000 00000000 00000000 00000000

RBDD> QUIT5
^P
?0031 Z connection terminated by ^P

>>> Z/BI:6 E6
?0033 Z connection successfully started
t/r

RBDD6> ST 0/TR7

;T1035_St      1.00
```

**Example 2-11 Cont'd on next page**



### Example 2–11 (Cont.): Running RBDs on I/O Devices

```
; T01 T02 T03 T04 T05 T06 T07 T08 T09 T10 T11 T12 T13 T14
; T15 T16 T17

;          P          6          410B 00000001
;00000000 00000000 00000000 00000000 00000000 00000000 00000000

;  PUDR: 5FF43FDF8

RBD6> QUIT
^P
?0031 Z connection terminated by ^P

>>>
```

The callouts in Example 2–11 are explained below.

- ① The SHOW CONFIGURATION console command shows that this system includes a DEMNA at node D of the XMI bus and a TBK70 at node 6 of the VAXBI attached at XMI node E. (See <REFERENCE>(VAX\_xxxx) *Series Owner's Manual* for more information on the SHOW CONFIGURATION command.)
- ② The Z command is typed at the console prompt. A connection is established to XMI node D. The console returns a message confirming that the connection has been made.
- ③ After the console message is returned in ②, no prompt is printed. Typing  $\uparrow/\uparrow$  invokes the RBD monitor on the adapter being tested and returns the RBD monitor prompt. Note that the *D* in the RBD prompt refers to the XMI node.
- ④ The RBD is started with trace set.
- ⑤ The QUIT command exits the RBD monitor. The Z connection remains until CTRL/P is entered.
- ⑥ Steps ② through ⑤ are repeated to run the RBD of the TBK70 at node 6 of the VAXBI attached at XMI node E.
- ⑦ The START command for VAXBI RBDs requires a space before the 0. When run with the /TR qualifier, test traces are printed.
- ⑧ The last line of the summary report indicates the contents of the Power-Up Diagnostic Register. To interpret the contents of this register, refer to the technical manual for the device being tested.

## 2.4 ROM-Based Diagnostics

The <REFERENCE>(xmp) diagnostic ROM has six diagnostics: RBD 0 and RBD 1 test the specified scalar and, if present, attached vector processor; RBD 2 tests the optional DWMBB adapter; RBD 3 tests <REFERENCE>(xma) memories; RBD 4 tests the <REFERENCE>(xmp) cache; and RBD 5 tests multiprocessor interaction.

Table 2–11: ROM-Based Diagnostic Programs — Callable Tests

RBD Program	Total Tests	Default Number of Callable Tests	Description
0	45	45	Runs CPU tests (scalar only)
	57	57	Runs CPU tests (scalar and vector)
1	18	18	Runs CPU/memory interaction tests
	22	22	Runs scalar/vector CPU/memory interaction tests
2	48	48	Runs DWMBB tests
3	14	10	Sizes and runs additional tests on main memory
4	3	0	Checks for cache incoherency
5	7	7	Runs multiprocessor tests
	8	7	Runs vector multiprocessor tests

RBD 0 is the same as the <REFERENCE>(xmp) self-test. It is useful for running several passes when a processor fails self-test intermittently. Section 2.4.1 shows examples of running RBD 0 on both the boot processor and a secondary processor, and lists the tests in RBD 0.

RBD 1 is the same as the CPU/memory interaction tests. It is useful for running several passes when a processor fails CPU/memory interaction tests intermittently. Section 2.4.2 shows an example and lists the tests.

RBD 2 is the set of tests that the boot processor runs for each DWMBB I/O adapter when the system is powered on. (The DWMBB has no on-board self-test of its own.) Section 2.4.3 has an example of this diagnostic and a list of tests.

RBD 3 is a set of XMI memory tests that sizes and runs extended tests on all of memory. Section 2.4.5 shows examples of this RBD and lists the tests.

RBD 4 is a set of tests you can run following system crash to check for cache incoherency. Section 2.4.6 has an example of this diagnostic and a list of tests.

RBD 5 tests the interaction of multiple <REFERENCE>(xmp) processors and, if present, multiple <REFERENCE>(xrv) processors. Section 2.4.7 includes an example of this RBD and a list of its tests.

For a detailed explanation of the diagnostic printout, see Section 2.3.7.

## 2.4.1 <REFERENCE>(xmp) Self-Test — RBD 0

**RBD 0 is equivalent to the <REFERENCE>(xmp) self-tests. The first 45 tests test scalar CPU modules; tests 46–57 test vector modules.**

### Example 2–12: <REFERENCE>(xmp) Self-Test — RBD 0

```
>>> T/R          ! Command to enter RBD monitor program.
RBD1>           ! RBD monitor prompt, where 1 is the hexa-
                ! decimal node number of the boot processor.
RBD1> ST0/TR/HE  ! Runs the <REFERENCE>(xmp) self-test on boot processor
                ! Trace prints each test number; halt on error
                ! Test results written to the console terminal:

; XMP/V_ST      1.00

; T0001 T0002 T0003 T0004 T0005 T0006 T0007 T0008 T0009 T0010
; T0011 T0012 T0013 T0014 T0015 T0016 T0017 T0018 T0019 T0020
; T0021 T0022 T0023 T0024 T0025 T0026 T0027 T0028 T0029❶

;      F❷      2      8080      1❸
;      HE MAXMIRAM      XX      T0029❶
;      28 5555AAAA A8AAAAAA 00000000 E1008000 E008C410 08

;      F      1      8080      1
; 00000000 00000001 00000000 00000000 00000000 00000000 00000000

RBD1>
```

In Example 2–12:

- ❶ Test 29 failed. The /HE switch causes execution to stop when the error is encountered.
- ❷ F indicates failure.
- ❸ The diagnostic ran for one pass.

**Example 2–13: Running <REFERENCE>(xmp) Self-Test (RBD 0) on a Secondary Processor**

```

>>> SET CPU 2 ❶
>>> T/R
RBD2> ST0/TR ❷

;XMP/V_ST          1.00

; T0001 T0002 T0003 T0004 T0005 T0006 T0007 T0008 T0009 T0010
; T0011 T0012 T0013 T0014 T0015 T0016 T0017 T0018 T0019 T0020
; T0021 T0022 T0023 T0024 T0025 T0026 T0027 T0028 T0029 T0030
; T0031 T0032 T0033 T0034 T0035 T0036 T0037 T0038 T0039 T0040
; T0041 T0042 T0043 T0044 T0045 T0046 T0047 T0048 T0049 T0050
; T0051 T0052 T0053 T0054 T0055 T0056 T0057

;          P          2          8080          1
;00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

In Example 2–13:

- ❶ This command causes the <REFERENCE>(xmp) module at node 2 to become the primary processor. This processor has an attached vector processor.
- ❷ The prompt indicates that the CPU at node 2 is the primary processor. RBD 0 is run on this processor.

**Table 2–12: <REFERENCE>(xmp)/<REFERENCE>(xrv) Self-Test — RBD 0**

Test	Function
<b>Scalar Processor Tests</b>	
T0001	<REFERENCE>(xmp) ROM Test
T0002	IPL Step-Down Test
T0003	MSSC Configuration Register Test
T0004	MSSC RAM Test
T0005	MSSC Output Port Test
T0006	MSSC Programmable Address Decode Strobe Test
T0007	MSSC UART External Loopback Test
T0008	MSSC UART Internal Loopback Test

**Table 2–12 (Cont.): <REFERENCE>(xmp)/<REFERENCE>(xrv) Self-Test — RBD 0**

<b>Test</b>	<b>Function</b>
T0009	<REFERENCE>(xmp) EEPROM Test
T0010	MSSC Input Port Test
T0011	MSSC Bus Timeout Control Register Test
T0012	MSSC Programmable Timers Test
T0013	MSSC TOY Clock Test
T0014	MSSC Interval Timer Test
T0015	Interrupts at IPL 14 to 17 Test
T0016	Primary Cache Tag Store Test
T0017	Primary Cache Data RAM March Test
T0018	Backup Cache Tag Store Test
T0019	Backup Cache Data Line Test
T0020	Backup Cache Data RAM March Test
T0021	Backup Cache Data Parity RAM Test
T0022	Cache Mask Write Test
T0023	Flush Cache Test
T0024	Data Parity Logic Test
T0025	Backup Tag Store Parity Error Test
T0026	DCSR Register Test
T0027	ECC Logic Test
T0028	ECC RAM March Test
T0029	MAXMI RAM March Test
T0030	XDEV Register Test
T0031	XBER and XBEER Registers Test
T0032	XFADR and XFAER Registers Test
T0033	XGPR Register Test
T0034	XCR Register Test

**Table 2–12 (Cont.): <REFERENCE>(xmp)/<REFERENCE>(xrv) Self-Test — RBD 0**

<b>Test</b>	<b>Function</b>
T0035	NSCSR Register Test
T0036	CNAK and TTO on Read Test
T0037	CNAK and TTO on Write Test
T0038	CNAK and TTO on IVINTR Test
T0039	Interprocessor IVINTR Test
T0040	Write Error IVINTR Test
T0041	Multiple Interrupt Test
T0042	MP-Chip Critical Path Test
T0043	MF-Chip Test
T0044	Disable MF-Chip Test
T0045	MF-Chip Critical Path Test
<b>Vector Processor Tests</b>	
T0046	VECTL Registers Test
T0047	Verse Registers Test
T0048	Load/Store Registers Test
T0049	VIB Error Logic Test
T0050	Other VECTL Chip Logic Test
T0051	Verse and Favor Test
T0052	Load/Store Translation Buffer and CAM Test
T0053	Load/Store Cache Test
T0054	Load/Store Instruction Test
T0055	Load/Store Tag Tests
T0056	Load/Store Error Cases Test
T0057	<REFERENCE>(xrv) Critical Path Test

## 2.4.2 CPU/Memory Interaction Tests — RBD 1

**RBD 1 is equivalent to the CPU/memory interaction tests. The first 18 tests test scalar processor modules; tests 19–22 test vector processor modules. Tests 1 and 2 are not run on power-up.**

### Example 2–14: CPU/Memory Interaction Tests — RBD 1

```
>>> T/R                ! Command to enter RBD monitor program
RBD3>                  ! RBD monitor prompt, where 3 is the hexa-
                        ! decimal node number of the processor
                        ! that is currently receiving your input.
RBD3> ST1/TR/HE/QV     ! Runs the CPU/memory interaction RBD with
                        ! trace, halt on error, and quick verify.
                        ! Test results written to the console
                        ! terminal:

;CPUMEM                1.00
; T0001 T0002 T0003 T0004 T0005 T0006 T0007 T0008 T0009 T0010
; T0011 T0012 T0013 T0014 T0015 T0016 T0017 T0018
;          P①          3      8080          1②
;0000000000 00000000 00000000 00000000 00000000 00000000 00000000
RBD3>
```

In the example above:

- ① P means that the diagnostic ran successfully.
- ② One pass was completed. (The system did not have a vector processor.)



**Table 2–13: CPU/Memory Interaction Tests — RBD 1**

<b>Test</b>	<b>Function</b>
<b>Scalar Processor Tests</b>	
T0001	Parity Error CNAK Read Test
T0002	Parity Error CNAK Write Test
T0003	Cache Disable Test
T0004	Interlock Instruction Cache Disable Test
T0005	Cache Read Fill Test
T0006	Cache Location Displacement Test
T0007	Interlock Instruction Cache Test
T0008	Hexword Writeback Test
T0009	Invalidate Bus Test
T0010	Error Transition Mode Test
T0011	IBUS March and Parity Error Test
T0012	Bad ACP During Writeback Test
T0013	At-Speed Cache Access Test
T0014	Upper Address Bit Test
T0015	Single-Bit ECC Error Test
T0016	Double-Bit ECC Error Test
T0017	Memory Write Merge Test
T0018	Backup Cache Tag Exerciser
<b>Vector Processor Tests</b>	
T0019	Cache Test
T0020	Write Buffer Test
T0021	Duplicate Tag Test
T0022	Miscellaneous Error Test

### 2.4.3 <REFERENCE>(xbi) Tests — RBD 2

**The <REFERENCE>(xbi) ROM-based diagnostic, RBD 2, checks functions of both <REFERENCE>(xbi) modules. RBD 2 tests the <REFERENCE>(XBI) modules and can trace the subtests, pinpointing errors.**

#### Example 2–15: <REFERENCE>(xbi) Tests — RBD 2

```
>>> T/R
RBD1> ST2/TR E①
;XBIP_TST      1.00
; T0001 T0002 T0003 T0004 T0005 T0006 T0007 T0008 T0009 T0010
; T0011 T0012 T0013 T0014 T0015 T0016 T0017 T0018 T0019 T0020
; T0021 T0022 T0023 T0024 T0025 T0026 T0027 T0028 T0029 T0030
; T0031 T0032 T0033 T0034 T0035 T0036 T0037 T0038 T0039 T0040
; T0041 T0042 T0043 T0044 T0045 T0046 T0047 T0048
;
;          P②          1          8080          1③
;000000000 00000000 00000000 00000000 00000000 00000000 00000000
```

The <REFERENCE>(XBI) has no on-board self-test. The boot processor ROM code tests <REFERENCE>(XBI)s during additional power-up tests. At power-up, the boot processor first sizes all <REFERENCE>(XBI)s and then serially tests each one.

- ❶ When invoking RBD 2 from the monitor, the START command requires a parameter. This parameter is the XMI node number (hexadecimal) of the <REFERENCE>(xbia) module of the <REFERENCE>(xbi) to be tested.
- ❷ This diagnostic ran successfully.
- ❸ One pass was completed.

## 2.4.4 DWMBB Tests — RBD 2 Subtests

**RBD 2 consists of 48 subtests, as shown in Table 2-14.**

**Table 2-14: <REFERENCE>(XBI\_TITLE) RBD Tests**

<b>Test</b>	<b>Function</b>
T0001	<REFERENCE>(XBI) CSR Test
T0002	<REFERENCE>(XMI) Low Longword Parity Error Test
T0003	<REFERENCE>(XMI) High Longword Parity Error Test
T0004	<REFERENCE>(XMI) Function and ID Parity Error Test
T0005	<REFERENCE>(xbia) Loopback Transaction Test
T0006	<REFERENCE>(xbia) Loopback DMA Buffer Test
T0007	<REFERENCE>(xbia) Loopback Nonexistent Memory Interrupt Test
T0008	Retry Timeout Disable Test
T0009	Timeout Disable Test
T0010	Data NO ACK Test
T0011	RER Error Interrupt Test
T0012	Lockout Assertion Test
T0013	Quick PMR Memory Test
T0014	DMA ECC Error Interrupt Test
T0015	PMR ECC Error Interrupt Test
T0016	ECC Syndrome Test
T0017	ECC Disable Test
T0018	Extended Addressing Test
T0019	34-bit Addressing Test
T0020	Invalid PFN Interrupt Test
T0021	Failing Command and Mask Test
T0022	Responder Request Test
T0023	<REFERENCE>(xbib) CSR Test

**Table 2–14 (Cont.): <REFERENCE>(XBI\_TITLE) RBD Tests**

<b>Test</b>	<b>Function</b>
T0024	Illegal I/O Command Test
T0025	BIIC Loopback Transaction Test
T0026	BIIC VAXBI Transaction Test
T0027	VAXBI Window Space Test
T0028	DMA Test
T0029	DMA Loopback DMA Buffer Test
T0030	XMI Parity Error Interrupt Test
T0031	Write Sequence Error Interrupt Test
T0032	Return Vector/Multiple Interrupt Test
T0033	I/O Buffer C/A Fetch Parity Error Interrupt Test
T0034	I/O Buffer Data Fetch Parity Error Interrupt Test
T0035	IDMA Buffer Data Fetch Parity Error Interrupt Test
T0036	VAXBI Interlock Read Error Interrupt Test
T0037	DMA-A Buffer C/A Load Parity Error Interrupt Test
T0038	DMA-A Buffer Data Load Parity Error IVINTR/INTR Test
T0039	DMA-B Buffer C/A Load Parity Error Interrupt Test
T0040	DMA-B Buffer Data Load Parity Error IVINTR/INTR Test
T0041	I/O Buffer Data Load Parity Error IVINTR/INTR Test
T0042	BCI Parity Error Test
T0043	Nonexistent Memory Interrupt Test
T0044	CRD Error Interrupt Test
T0045	VAXBI Interrupt Test
T0046	VAXBI IP Interrupt Test
T0047	Control Reset Test
T0048	No Stall Timeout Test

## 2.4.5 MS65A Memory Tests — RBD 3

**RBD 3 sizes memory, runs extended memory tests, and indicates any failing tests. Examples 2-16 through 2-19 show the use of various qualifiers.**

### Example 2-16: RBD Test on All Modules with Halt on Error

```
>>>                                     ! Console program prompt
>>> T/R                                   ! Command to enter RBD monitor program
RBD3>                                     ! RBD monitor prompt, where 3 is the hexa-
                                           ! decimal node number of the processor
                                           ! that is currently receiving your input

RBD3> ST3/TR/HE                           ! Runs the default <REFERENCE>(XMA) RBD
                                           ! test; test results written to the
                                           ! console terminal; tests will halt on
                                           ! any error found (/HE)

;XMA2_RBD          1.00
; T0002 T0003 T0004 T0005 T0006 T0007 T0008
;           P           3      8001           1
;00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

### Example 2-17: RBD Test on Module in Slot A

```
RBD3> ST3/TR A                           ! Runs the <REFERENCE>(XMA) RBD test
                                           ! on memory module in slot A only;
                                           ! test results written to the console
                                           ! terminal

;XMA2_RBD          1.00
; T0002 T0003 T0004 T0006 T0007 T0008 T0009
;           P           3      8001           1
;00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

### Example 2-18: RBD Test with Module Error

```
RBD3> ST3/TR                                ! Runs the default <REFERENCE>(XMA) RBD
                                           ! test; test results written to the
                                           ! console terminal; a hard error is found
                                           ! in the memory module in slot 8

;XMA2_RBD          1.00
; T0002 T0003 T0004 T0006 T0007 T0008 T0009
;      F          3      8001          1
;      HE ERRLOGIC      08      T0005
;      00 00000000 00000000 00000000 00000000 20073E32 16
;      F          3      8001          1
;00000000 00000001 00000000 00000000 00000000 00000000 00000000
```

### Example 2-19: RBD Test with Confirm Switch

```
RBD3> ST3/TR/T=5:12 A /C                    ! Runs RBD tests T0005 through T0012 on
                                           ! memory module in slot A. Confirm
                                           ! destructive memory test switch (/C)
                                           ! is required on tests T0005, T0009
                                           ! and T0012.

;XMA2_RBD          1.00
; T0005 T0006 T0007 T0008 T0009 T0010 T0011
;      S          1      8001          1      ! Test status prints out every
;      XX          RAM      XX      T0011      ! 60 sec until tests are completed;
                                           ! /DS disables test status printout.
; T0012
;      S          1      8001          1
;      XX          RAM      XX      T0012
;      P          3      8001          1
;00000000 00000000 00000000 00000000 00000000 00000000 00000000

RBD3> QUIT                                  ! Exit from RBD monitor program
>>>                                         ! Console prompt returns
```

**Table 2–15: Memory Tests — RBD 3**

<b>Test</b>	<b>Function</b>	<b>Approximate Run Time (For 32-Mbyte Module)</b>
T0001 <sup>1</sup>	Memory Self-Test	12 sec <sup>2</sup>
T0002 <sup>3</sup>	CSR Addressability Test	3 sec
T0003 <sup>3</sup>	CSR Read/Write, Write 1 to Clear Test	3 sec
T0004 <sup>3</sup>	SEADR Register Test	3 sec
T0005 <sup>1</sup>	Parity Error Test	3 sec
T0006 <sup>3</sup>	Error Correction Code Circuit Test	3 sec
T0007 <sup>3</sup>	Data Path Test	3 sec
T0008 <sup>3</sup>	Write Mask Logic Test	3 sec
T0009 <sup>1</sup>	Block State Test	3 sec
T0010 <sup>3</sup>	EEPROM Update Test	3 sec
T0011 <sup>3</sup>	Interleave Address and Boundary Test	22 sec
T0012 <sup>1</sup>	ECC RAM March Test	9 min
T0013 <sup>1</sup>	March Test	10 min
T0014 <sup>1</sup>	Modified MOVI Test	4 hrs

<sup>1</sup>The /C qualifier is required for these tests.

<sup>2</sup>If self-test fails, there is a 60 second timeout.

<sup>3</sup>Tests T0002–T0004, T0006–T0008, and T0010–T0011 are run by default.



Tests T0002, T0003, T0004, T0006, T0007, T0008, T0010, and T0011 are run by default. All other tests must be selected by the user. Tests are performed on all <REFERENCE>(XMA) modules unless the user specifies a single <REFERENCE>(XMA). Parameters specified in the command line (refer to Table 2–16) allow one or all memory modules to be tested. These parameters also allow RBD tests to be run from main memory or ROM for RBD tests T0013 and T0014.

**Table 2–16: RBD 3 Parameters**

<b>Parameter<sup>1</sup></b>	<b>Function</b>
00	Run tests T0013 and T0014 from main memory (RAM) and test all memory modules
0 <i>n</i>	Run tests T0013 and T0014 from main memory (RAM) and test memory module <i>n</i> only
10	Run tests T0013 and T0014 from ROM and test all memory modules
1 <i>n</i>	Run tests T0013 and T0014 from ROM and test memory module <i>n</i> only

<sup>1</sup>Where *n* is the memory module backplane slot number that is specified in hex parameters 0*n* and 1*n*.

The CPU/memory interaction diagnostic also runs tests that exercise memory (see Section 2.4.2).

## 2.4.6 KA65A Cache Tests — RBD 4

**RBD 4 tests backup cache. A test number must be supplied to run any of the three tests.**

### Example 2–20: KA65A Cache Tests — RBD 4

```
>>> T/R                               ! Command to enter RBD monitor program
RBD4>                                  ! RBD monitor prompt, where 4 is the hexa-
                                       ! decimal node number of the processor
                                       ! that is currently receiving your input.
RBD4> ST4/TR①
;XMP_BC      1.00
;      S      4      8080      1
;      XX NoTstSel②      XX      T0000
;      P③      4      8080      1
;00000000 00000000 00000000 00000000 00000000 00000000 00000000
RBD4> ST4/TR/T=1④
;XMP_BC      1.00
; T0001⑤
;      S      4      8080      1
;      XX Cache      XX      T0001
;00000800 00000000 00000000 1F0C0000 00000300 00000000 00000000 00000000
;      P⑥      4      8080      1
;00000000 00000000 00000000 00000000 00000000 00000000 00000000
RBD4>
```

- ① RBD 4 is started without specifying a test number.
- ② No tests are run, and a status message is given that no test was selected.
- ③ Since no tests were run, no failures were detected.
- ④ The command to run RBD 4 is reissued, this time with a test number.
- ⑤ Test 1, as requested, is run.
- ⑥ Test 1 passes.

**Table 2–17: KA65A Cache Tests — RBD 4**

<b>Test</b>	<b>Function</b>
T0001	Parity Error Test
T0002	Cache Coherency Checker
T0003	Memory Locked Location Test

## 2.4.7 Multiprocessor Tests — RBD 5

**RBD 5 is equivalent to the multiprocessor tests. The first 7 tests check scalar processor modules; test 8 checks vector processor modules.**

### Example 2–21: Multiprocessor Tests — RBD 5

```
>>> T/R                ! Command to enter RBD monitor program
RBD3>                  ! RBD monitor prompt, where 3 is the hexa-
                        ! decimal node number of the processor
                        ! that is currently receiving your input.

RBD3> ST5/TR①
;XMP_MP      1.00
; T0001 T0002 T0003 T0004 T0005 T0006 T0007
;          P②      1      8080      1③
;00000000 00000000 00000000 00000000 00000000 00000000 00000000
RBD3>
```

In the example above:

- ① RBD 5 is run with trace set.
- ② The diagnostic ran successfully.
- ③ One pass was completed.

**Table 2–18: Multiprocessor Tests — RBD 5**

<b>Test</b>	<b>Function</b>
<b>Scalar Processor Tests</b>	
T0001	Interprocessor Interrupt (IVINTR) Test
T0002	Write Error Interrupt (WEINTR) Test
T0003	Cache Invalidate Test
T0004	XMI Bus Arbitration Test
T0005	XMI Bus Arbitration Collision Test
T0006	XMI Suppress Assertion Test
T0007	Memory Lock and Interrupt Exerciser Test
<b>Vector Processor Test</b>	
T0008	Vector Cache Coherency Test

**Table 2–19: RBD 5 Parameters**

<b>Parameter</b>	<b>Function</b>
No parameter	When no parameter is specified, all scalar processors that have passed power-up test will be tested.
xxxx	Specifies a hexadecimal bit mask indicating slot positions of the scalar processors to be tested. For example, a parameter of 322 indicates that processors in slots 1, 5, 8, and 9 will be tested. All processors specified are tested, even those that did not pass power-up test.

## 2.5 VAX Diagnostic Supervisor Programs

**The VAX Diagnostic Supervisor (VAX/DS) is a monitor that controls operation of diagnostic programs. You can use VAX/DS in one of two modes: standalone mode (exclusive use of the system) or user mode (under the VMS operating system).**

**Table 2–20: VAX Diagnostic Program Levels**

<b>Level</b>	<b>Type of Test</b>	<b>Run-Time Environment</b>
1	System exercisers	Runs under the VMS operating system without VAX/DS
2R	Function tests of peripheral devices	Runs under the VMS operating system with VAX/DS
2	Exercisers and function tests of peripheral devices and processors	Runs under VAX/DS in user mode and standalone mode
3	Function tests and logic tests of peripheral devices and processors	Runs under VAX/DS in standalone mode

**Table 2–21: VAX/DS Documentation**

<b>Document</b>	<b>Order Number</b>
<i>VAX Diagnostic Supervisor User's Guide</i>	AA-FK66A-TE
<i>VAX Diagnostic Software Handbook</i>	AA-F152A-TE
<i>VAX Diagnostic Design Guide</i>	AA-FK67A-TE
<i>VAX Systems Hardware Handbook</i>	EB-31692-46

The VAX Diagnostic Supervisor (VAX/DS) can be run in interactive mode. You type commands in response to the VAX/DS program prompt:

DS>

VAX/DS lets you load diagnostic programs into system memory, select devices to be tested, and run the programs. The VAX/DS command language also lets you control the execution of diagnostic programs; you can specify which tests or sections of a program should run, and how many passes it should run. You can also show the current state of parameters that affect the operation of diagnostic programs. The programs report their results through VAX/DS to the terminal.

VAX/DS supports three types of diagnostic programs:

- **Logic tests**  
Test a specific section of a device's logic circuitry. Logic tests provide the greatest degree of detail in determining the location of faulty hardware.
- **Function tests**  
Test the functions of the device. For example, a function test for a disk drive would test the drive's reading and writing capabilities. Function tests can detect the location of faulty hardware, although the results may be less exact than those of a logic test.
- **Exercisers**  
Test entire systems or subsystems and verify that a system can function properly over a period of time. Exercisers can detect both hardware faults resulting from the simultaneous use of a system's numerous devices and intermittent faults occurring only once or twice over a long period of time.

VAX/DS also supports EVUCA, the EEPROM patch and console boot device utility.

Table 2-22 lists the VAX/DS programs available for the <REFERENCE>(VAX\_mod\_XXX) system. Each program has a HELP file available. To access the help files for any diagnostic, at the VAX/DS prompt, type:

```
DS> HELP [VAX/DS diagnostic program name]
```

## 2.5.1 Running VAX/DS in Standalone Mode

**You can use VAX/DS in one of two modes: standalone mode (exclusive use of the system) or user mode (under VMS).**

### Example 2-22: Running VAX/DS in Standalone Mode

```
>>> BOOT/XMI:m/FILENAME:ISL_LVAX/R5:10 EX01
  Initializing system
      [Self-test display prints]
  Loading system software
      [System messages print]
  Ethernet Initial System Load Function
  FUNCTION ID      FUNCTION
    1      -      Display Menu
    2      -      Help
    3      -      Choose Service2
    4      -      Stop
  Enter a function Id value: 32
  Service options:
    1 = Find Services
    2 = Enter Known Service Name
  =>13
  Servers found:: 2
  Service Name Format:
    Service Name
    Server Name
    Ethernet ID
  #1
  NSS_SYSDISK
  ESS_08002B15FCE1
  08-00-2B-15-FC-E1
  #2
  6000_DIAG_*4
  ESS_08002B15FCE1
  08-00-2B-15-FC-E1
  Enter a number => 25
      [Diagnostic Supervisor Banner prints]
DS>
```



Load the diagnostic CD into the CD server console load device.

- ① Boot the system. The BOOT command you enter depends on the system configuration. If the CD server is connected to a DEMNA, use the command shown; if it is connected to a DEBNI or DEBNA, use the command:

```
>>> BOOT/XMI:m/FILENAME:ISL_LVAX/BI:n/R5:10 ET0
```

- ② The system prompts, *Enter a function Id value:*. Enter 3 to choose service.
- ③ The system displays the service options menu and the prompt, *=>*. Enter 1 to select *Find Services*. The available Ethernet services are located and displayed. In this example, one server was found with two CD drives.
- ④ One of the services will show the diagnostic CD that you loaded. Here it is in the second drive.
- ⑤ The system prompts, *Enter a number =>*. Enter the number of the service with the Diagnostic Supervisor CD. If more than one service name 6000\_DIAG\_\* of the VAX 6000 CMPLT DIAG CD ROM is loaded in the CD drives, choose the drive where the \* revision letter is the highest. This will be the most recent version of the diagnostic media.

## 2.5.2 Running VAX/DS in User Mode

**You can use VAX/DS in one of two modes: standalone mode (exclusive use of the system) or user mode (under VMS).**

### Example 2–23: Running VAX/DS in User Mode

```
$                                     ! At the operating system prompt, run
$ RUN EMSAA                          ! the VAX/DS program.
                                     ! [VAX/DS banner prints, as in example above]
DS>                                  ! VAX/DS prompt appears.
                                     ! Run VAX/DS level 2R or 2 programs.
DS> EXIT                             ! Type EXIT to exit VAX/DS
$                                     ! Operating system prompt returns.
```

Table 2-20 describes the levels of VAX/DS programs. Check Table 2-22 for the programs you wish to run, and determine if you will run VAX/DS in standalone or user mode.

In both standalone and user mode, VAX/DS functions the same way. Typically a program running in user mode provides less detailed results than one running in standalone mode. For more information on VAX/DS, see the documents listed in Table 2-21.

### 2.5.3 Sample VAX/DS Session

**When you run the VAX/DS programs, run the system autosizer program EVSBA first. This program, which takes several minutes to execute, will save you time as you proceed with other tests. Certain conditions cause the generation of an unexpected trap or interrupt. Use the method shown to avoid these conditions.**

#### Example 2-24: Sample VAX/DS Session, Part 1 of 2

```
>>> SET BOOT DIAG /XMI:9/R5:10 DU1
>>> BOOT DIAG ❶
           [self-test results print]
Loading system software
* Initializing adapter
* Specified adapter initialized successfully
* Connecting to boot disk
* Reading bootblock from disk
* Passing control to transfer address

           Copyright Digital Equipment Corporation
           1989, 1990.
           All Rights Reserved.

DIAGNOSTIC SUPERVISOR. ZZ-EMSAA-V14.0-561  1-DEC-1990 11:39:12
DS> RUN EVSBA ❷
           [banner prints]
.. Program: EVSBA - AUTOSIZER level 3, revision 7.0, 3 tests,
   at 11:43:33.20.
.. End of run, 0 errors detected, pass count is 1,
   time is 1-DEC-1990 11:44:59.66
```

- ① The SET BOOT command stores a nickname for a set of parameters to the BOOT command. (The lower key switch on the control panel must be set to Update when this command is issued.) This BOOT command loads VAX/DS from disk. For more information on the BOOT and SET BOOT commands, see the <REFERENCE>(vax\_xxxx) *Series Owner's Manual*.
- ② The off-line autosizer program EVSBA identifies hardware on your system and builds a database for the VAX Diagnostic Supervisor. The autosizer eliminates the need for you to type in the name and characteristics of the hardware you intend to test under VAX/DS with level 3 diagnostic programs.

## Example 2-25: Sample VAX/DS Session, Part 2 of 2

```
DS> SHO DEV ③
_DUA   KDM70   HUB   61C80000  XMI Node Number (1 to E) =00000009(X)
Bus Request Level (4 - 7) =5.
_DUA1  RA70    _DUA   72000000
_KA0   KA65A   HUB   61980000  XMI Node Number (1 to E) =00000003(X)
Vector Unit Present=Yes
_DUA2  RA70    _DUA   72000000
_EXA0  DEMNA   HUB   61D80000  XMI Node Number (1 to E) =0000000B(X)
_PAA0  CIXCD   HUB   61E00000  XMI Node Number (1 to E) =0000000C(X)
CI Node Number (0 to 224) =1.
_DWMB0 DWMB0   HUB   61E80000  XMI Node Number (1 to E) =0000000D(X)
BI Node Number (HEX)=00000002(X)
_TXA   DHB32   _DWMB0 7A006000  BI Node Number (HEX) =00000003(X)
_SLA   DSB32   _DWMB0 7A01E000  BI Node Number (HEX) =0000000F(X)
_DWMB0 DWMB0   HUB   61F00000  XMI Node Number (1 to E) =0000000E(X)
BI Node Number (HEX)=00000001(X)
_BLA0  DWBLA   _DWMB0 7C004000  BI Node Number (HEX)=00000002(X)
_MUB0  TU81    _BLA0   7C4BF940  CSR=774500(O) VECTOR=000260(O) BR=5.
_TXB   DMB32   _DWMB0 7C006000  BI Node Number (HEX) =00000003(X)
_DUB   KDB50   _DWMB0 7C008000  BI Node Number (HEX)=00000004(X)
_DUB0  RA70    _DUB    7C500000
_MUC   TBK70   _DWMB0 7C00C000  BI Node Number (HEX)=00000006(X)
_MUC6  TK70    _MUC    7C580000
_ETD   DEBNA   _DWMB0 7C018000  BI Node Number (HEX)=0000000C(X)
_ETD0  LANCE   _ETD    7C700000
DS> SELECT ALL ④
DS> SET TRACE
DS> RUN EVKAQ

      [banner prints]

.. Program: ZZ-EVKAQ, VAX Basic Instructions Exerciser, revision 3.5, 92
tests, at 11:46:11.90.
Testing: _KA0

Test 1: BRB Instruction Test
Test 2: BRW Instruction Test
Test 3: BBC Instruction Test
.
.
.
Test 90: XORL2 Instruction Test
Test 91: XORL3 Instruction Test
Test 92: ROTL Instruction Test
.. End of run, 0 errors detected, pass count is 1,
   time is 1-DEC-1990 11:46:09.88
DS>
```

- ③ You can use the autosizer to print a list of system hardware by running the program EVSBA under VAX/DS and typing the VAX/DS command SHOW DEVICE. The command lists system devices, similar to the SHOW CONFIGURATION command in console mode.
- ④ SELECT ALL selects all devices listed in ③. SET TRACE enables printing of test numbers and names when the diagnostic runs.

## 2.5.4 VAX/DS Diagnostics

**Table 2–22 lists the VAX Diagnostic Supervisor tests currently available for the <REFERENCE>(VAX\_mod\_XXX) system.**

**Table 2–22: VAX Diagnostic Supervisor Programs**

<b>Diagnostic</b>	<b>Level</b>	<b>Diagnostic Title</b>
EMSAA <sup>1</sup>		<REFERENCE>(VAX_mod_XXX) Diagnostic Supervisor
EVSBA	3	VAX Standalone Autosizer
EVUCA	3	VAX 6000 EEPROM Utility
<b>&lt;REFERENCE&gt;(xmp)-Specific Diagnostic</b>		
EMKAX <sup>1</sup>	3	Manual Tests
<b>&lt;REFERENCE&gt;(xrv)-Specific Diagnostics</b>		
EVKAG	2	VAX Vector Instruction Exerciser, Part 1
EVKAH	2	VAX Vector Instruction Exerciser, Part 2
<b>VAX CPU Cluster Exerciser</b>		
EVKAQ	2	VAX Basic Instructions Exerciser, Part 1
EVKAR	2	VAX Basic Instructions Exerciser, Part 2
EVKAS	2	VAX Floating-Point Instruction Exerciser, Part 1
EVKAT	2	VAX Floating-Point Instruction Exerciser, Part 2
EVKAU	3	VAX Privileged Architecture Instruction Test, Part 1
EVKAV	3	VAX Privileged Architecture Instruction Test, Part 2

<sup>1</sup>Diagnostic software with file names beginning with EM are tests created specifically for the <REFERENCE>(VAX\_mod\_XXX) system. This software is not transportable.



**Table 2–22 (Cont.): VAX Diagnostic Supervisor Programs**

<b>Diagnostic</b>	<b>Level</b>	<b>Diagnostic Title</b>
<b>CIBCA-BA Diagnostics</b>		
EVGEE	3	CIBCA-B Repair Level Diagnostic, Part 1
EVGEF	3	CIBCA-B Repair Level Diagnostic, Part 2
EVGEG	3	CIBCA-B Repair Level Diagnostic, Part 3
EVGAA	3	CI Functional Diagnostic, Part 1
EVGAB	3	CI Functional Diagnostic, Part 2
EVGAC	3	Standalone CI Exerciser
EVGDA	3	CIBCA EEPROM Update Utility
EVXCI	1	VAX CI Exerciser
<b>CIXCD Diagnostics</b>		
EVGAA	3	CI Functional Test, Part 1
EVGAB	3	CI Functional Test, Part 2
EVGAC	3	Standalone CI Exerciser
EVXCI	1	VAX CI Exerciser
EVGEA	3	XCD Repair Level Diagnostic
EVGEB	3	XCD Firmware Loader Program
<b>DEC LANcontroller 200 Diagnostics</b>		
EVDYD	2R	DEBNI Online Functional Diagnostic
EVDWC	2R	VAX NI Exerciser
<b>DEC LANcontroller 400 Diagnostics</b>		
EVDYE	2R	DEMNA NI Functional Diagnostic
EVGDB	2	DEMNA EEPROM Update Utility
EVDWC	2R	VAX NI Exerciser

**Table 2–22 (Cont.): VAX Diagnostic Supervisor Programs**

<b>Diagnostic</b>	<b>Level</b>	<b>Diagnostic Title</b>
<b>DHB32 Diagnostics</b>		
EVDAR	3	DHB32 Diagnostic
EVDAS	2R	DMB32/DHB32 Asynchronous Diagnostic
<b>DMB32 Diagnostics</b>		
EVD AJ	2R	DMB32 Online Asynchronous Port Test
EVD AK	3	DMB32 Standalone Functional Verification
EVD AL	2R	DMB32 Online Synchronous Port Test
EVD AN	2R	DMB32 Online Data Communications Link
<b>DRB32 Diagnostics</b>		
EVD RH	3	DRB32-M, -E Functional Diagnostic
EVD RI	3	DRB32-W Functional Diagnostic
<b>DSB32 Diagnostics</b>		
EVD AP	3	DSB32 Level 3 Diagnostic
EVD AQ	2R	DSB32 Level 2R Diagnostic
<b>KDB50 Diagnostics</b>		
EVD RF	3	UDA50/KDB50 Basic Subsystem Diagnostic
EVD RG	3	UDA50/KDB50 Disk Drive Exerciser
EVD RB	3	UDA/KDB50 Basic Disk Formatter
EVD RJ	3	VAX UDA50-A/KDB50/KDM70 Exerciser
EVD RK	3	VAX Bad Block Replace Utility
EVD RL	3	VAX Disk Resident Error Log Utility
EVD RAE	2R	Generic MSCP Disk Exerciser

**Table 2–22 (Cont.): VAX Diagnostic Supervisor Programs**

<b>Diagnostic</b>	<b>Level</b>	<b>Diagnostic Title</b>
<b>KDM70 Diagnostics</b>		
EVRAE	2R	Generic MSCP Disk Exerciser
EURLJ	3	VAX UDA50/KDB50/KDM70 Exerciser
EVRLM	3	KDM70 EEPROM Update Utility
EURLN	3	DUP Control Program
<b>KLESI-B/TU81 Diagnostics</b>		
EVMBBA	2R	VAX TU81 Data Reliability
EVMBB	3	VAX Front-End/Host Functional Diagnostic
<b>MS65A Online Memory Diagnostic</b>		
EVKAM	2R	VAX Memory User Mode Test
<b>RV20 Diagnostics</b>		
EVRVA	3	RV20 Level 3 Functional Diagnostic
EVRVB	2R	RV20 Level 2R Diagnostic
EVRVC	2R	RV60/20 Level 2R DUP Diagnostic
<b>TBK Diagnostic</b>		
EVMDA	2R	VAX TK50/TK70 Exerciser
<b>TM32 Diagnostics</b>		
EVMEA	2R	TM32 L2R Reliability Diagnostic
EMBEB	3	TM32 L3 Functional Diagnostic Part 1
EMBEC	3	TM32 L3 Functional Diagnostic Part 2

# <REFERENCE>(xmp) Scalar Processor

---

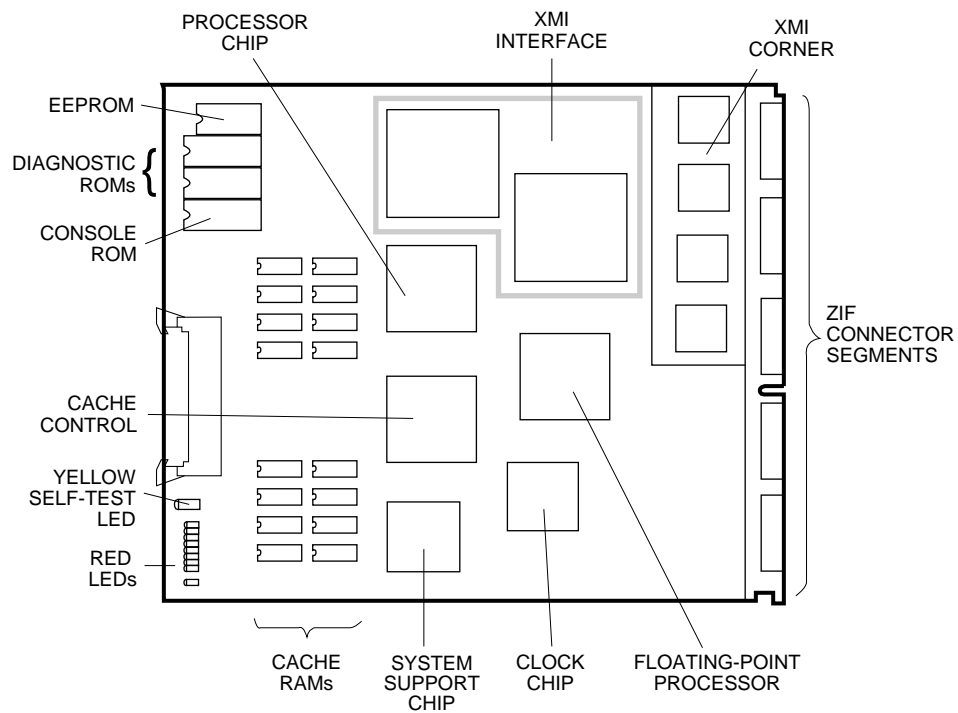
This chapter contains the following sections:

- Physical Description and Specifications
- Configuration Rules
- Functional Description
- Boot Processor
- Power-Up Sequence
- ROM-Based Diagnostics
- VAX/DS Diagnostics
- Machine Checks
- Console Commands
- <REFERENCE>(xmp) Handling Procedures
- How to Replace the Only Processor
- How to Replace the Boot Processor
- How to Add a New Processor or Replace a Secondary Processor
- Using EVUCA to Patch the EEPROM
- <REFERENCE>(xmp) Registers

### 3.1 <REFERENCE>(xmp) Physical Description and Specifications

The <REFERENCE>(xmp) is a single-module VAX processor. The module designation is <REFERENCE>(Txxxx). <REFERENCE>(VAX\_mod\_XXX) systems include multiple <REFERENCE>(xmp) processors, which use the 100 Mbyte/second <REFERENCE>(XMI) system bus to communicate with memory. Features of the module are shown in Figure 3-1.

Figure 3-1: <REFERENCE>(xmp) Module



msb-0711-90

**Table 3–1: <REFERENCE>(xmp) Specifications**

---

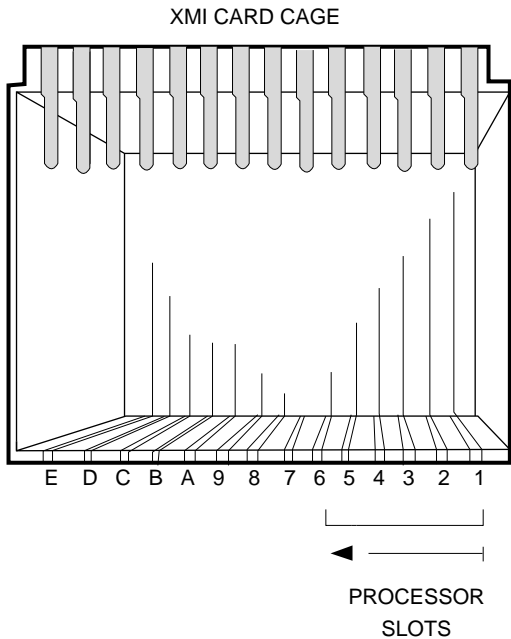
<b>Parameter</b>	<b>Description</b>
<b>Module Number:</b>	<REFERENCE>(Txxxx)
<b>Dimensions:</b>	23.3 cm (9.2") H x 0.23 cm (0.093") W x 28.0 cm (11.0") D
<b>Temperature:</b>	
Storage Range	-40°C to 66°C (-40°F to 151°F)
Operating Range	10°C to 40°C (41°F to 122°F)
<b>Relative Humidity:</b>	
Storage	10% to 95% noncondensing
Operating	10% to 95% noncondensing
<b>Altitude:</b>	
Storage	Up to 4.8 km (16,000 ft)
Operating	Up to 2.4 km (8000 ft)
<b>Current:</b>	4.5A at +5.1V 3.7A at +3.3V
<b>Power:</b>	41W
<b>Cables:</b>	VIB cable, 17-02240-03 (only when FV64A vector processor is attached)
<b>Diagnostics:</b>	ROM-based diagnostics 0, 1, 4, and 5 VAX/DS diagnostics, see Section 3.7

---

### 3.2 <REFERENCE>(xmp) Configuration Rules

<REFERENCE>(xmp) modules will operate in any slot of the XMI card cage; however, processors usually go on the right, beginning with slot 1. Special rules apply if the <REFERENCE>(xmp) has an attached vector processor; see Section 4.3.

Figure 3-2: Typical <REFERENCE>(xmp) Configuration



msb-0054-88

Processor modules are configured after I/O adapters. (I/O adapters are installed left to right in slots E through A and 5 through 1.) In a system with only scalar processors, the KA65A modules are installed right to left, beginning with the first available slot on the right.

In a system with vector processors, each scalar/vector pair requires three consecutive slots. An attached vector processor must be in the slot to the left of the <REFERENCE>(xmp) module. The slot to the left of the vector processor can be used *only* for a memory module. Installing another kind of module in that slot can damage the vector module.

For performance reasons, the scalar processor of a scalar/vector pair should not be made the primary processor when other scalar processors are in the system.

If the system is an H9657-CX upgrade, the T2019 module must be in slot 2, and slot 1 must be empty. Since slot 1 does not hold a processor module in this type of upgrade, the progress trace will not print as part of the self-test printout. (See Section 2.2.1.)

A processor module should be replaced if it consistently fails self-test, or if it causes the operating system to crash. However, you can leave the module in the system temporarily, since the console program prevents the operating system from using that processor. If a processor module fails intermittently, you should prevent the operating system from using it by doing the following:

1. Enter console mode.
2. Use the command SET CPU/NOENABLE to remove the processor from the software configuration.
3. Reboot the operating system.





The CPU section includes:

- The processor chip, which implements the VAX base instruction group and data types. It contains a 64-entry, fully associative translation buffer for both process and system-space mappings. The processor chip includes a 2-Kbyte, direct-mapped, write-through instruction and data cache with a quadword block and fill size.
- A floating-point accelerator chip that enhances the computation phase of floating-point and some integer instructions. This chip receives operands from the processor chip, computes the result, and passes the result and status back to the processor chip to complete the instruction.

The backup cache is a 512-Kbyte, direct-mapped, writeback instruction and data cache. It is implemented in 64-Kbyte x 4 data RAMs. The backup cache contains 4-Kbyte tags, organized to provide a hexword fill size and a 4-hexword block size.

The <REFERENCE>(XMI) interface performs the following tasks:

- Translates memory and I/O space references from the processor chip to the appropriate XMI transactions.
- Implements an eight-hexword writeback buffer to handle writeback requests from the cache.
- Supports control of cache fills and cache invalidates.
- Supports XMI-required interrupt logic.
- Implements all XMI-required registers.

The system support section includes support for external ROM/EEPROM, 1 Kbyte of battery-backed-up RAM, console terminal UARTs, bus reset logic, interval and programmable timers, time-of-year clock, bus timeout, and halt arbitration logic.

### Example 3-1: ROM and EEPROM Version Numbers

```

#123456789 0123456789 0123456789 0123456789 012345#
F  E  D  C  B  A  9  8  7  6  5  4  3  2  1  0  NODE #
      A  A  .  .  .  M  M  M  M  .  .  P  P  P      TYP
      O  +  .  .  .  +  +  +  +  .  .  +  +  +      STF
      .  .  .  .  .  .  .  .  .  .  .  E  E  B      BPD
      .  .  .  .  .  .  .  .  .  .  .  +  +  +      ETF
      .  .  .  .  .  .  .  .  .  .  .  E  E  B      BPD
.  .  .  .  .  .  .  +  +  +  .  +  .  .  +  .  XBI E +
      .  .  .  .  .  A4 A3 A2 A1  .  .  .  .  .      ILV
      .  .  .  .  .  64 64 64 64  .  .  .  .  .      256 Mb
Console = V1.00  RBDs = V1.00  EEPROM = 1.00/1.00  SN = SG01234567
  ①                ②                ③

```

The console and diagnostics sections include:

- A console read-only memory (ROM), which contains the code for initialization, executing console commands, and bootstrapping the system. The last line of the self-test display shows the ROM version. In this example, callout ❶ indicates that the console ROM is version V1.00.
- A diagnostic ROM, which contains the <REFERENCE>(vax\_mod\_xxx) ROM-based diagnostics (RBDs). The diagnostic ROM has the same version number as the console ROM. Callout ❷ indicates that the diagnostic ROM is version V1.00.
- An electrically-erasable, programmable ROM (EEPROM), which contains system parameters and boot code. You can modify the parameters with the console SET commands. Patching console and diagnostic code in the ROMs is accomplished by reading the patches into a special area of the EEPROM. The EVUCA utility, which patches the EEPROM, is described in Section 3.14.

Callout ❸ indicates two EEPROM version numbers separated by a slash (/). The first number is the format version of the EEPROM. This version is changed only when the internal structure of the EEPROM is modified.

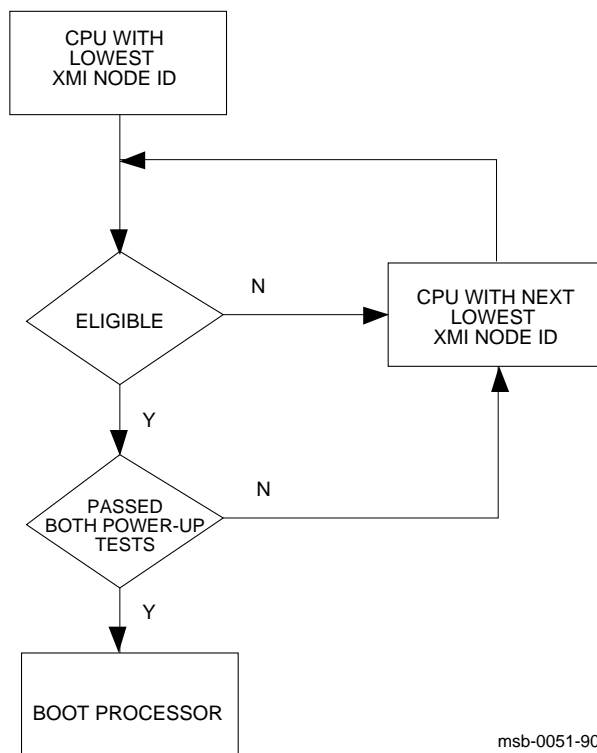
The second number is the revision of ROM patches that have been applied to the EEPROM. The major number in this revision (before the decimal point) corresponds to the major number of the ROM revision ❶. The minor number indicates the actual patch revision. In this example the EEPROM has not been patched for console ROM V1.00.

- A system support chip (MSSC) that includes support for external ROM/ EEPROM, 1 Kbyte of battery-backed-up RAM, console terminal UARTs, bus reset logic, interval timer, programmable timers, time-of-year (TOY) clock, bus timeout, and halt arbitration logic.

### 3.4 Boot Processor

In the <REFERENCE>(VAX\_mod\_XXX) system all <REFERENCE>(xmp) processors share system resources equally. The processor controlling the console at any given time is designated as the primary or boot processor. The others are called secondary processors. The boot processor is selected during the power-up sequence.

Figure 3-4: Selection of Boot Processor



Using boot code stored in its ROM or EEPROM, the boot processor reads the boot block from a specified device. Booting may be triggered by a command issued to the boot processor from the console, or by a system reset with the bottom key switch in the Auto Start position.

The boot processor also communicates with the system console, using the common console lines on the backplane. When you change system parameters in the EEPROM using SET commands, the boot processor automatically copies the new values to the EEPROMs on the secondary processors. If you swap in a new <REFERENCE>(xmp) module, it should be configured as a secondary processor. Then you can use the UPDATE command to copy the boot processor's entire EEPROM to the new secondary. See the <REFERENCE>(VAX\_XXXX) *Series Owner's Manual* for a description of the UPDATE command.

Usually the processor with the lowest <REFERENCE>(XMI) node number (which is also the lowest slot number) is selected as the boot processor. However, if this processor does not pass all its power-up tests, the next higher-numbered processor is selected. This is one way the boot processor can change.

The user also has control over boot processor selection with the SET CPU command. This command may declare a processor ineligible for selection. SET CPU can also select a boot processor explicitly.

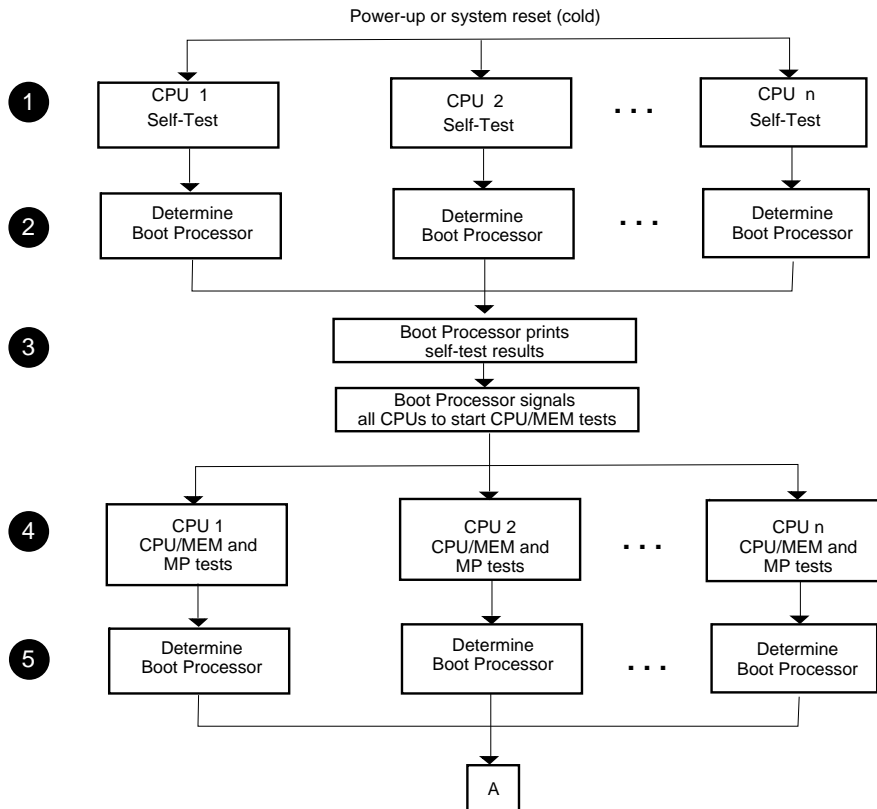
You can see the boot processor selection three ways:

- In the self-test display, the boot processor is indicated by a **B** on the second line labeled **BPD**.
- In console mode, the command SHOW CPU displays the boot processor as "Current primary."
- The bottom red LED is off on the boot processor module. It is lit on secondary processors.

### 3.5 Power-Up Sequence

**Figure 3-5 shows the power-up sequence for <REFERENCE>(xmp) processors. All processors execute two phases of self-test, and a boot processor is selected. The boot processor tests the VAXBI adapters and prints the self-test display.**

**Figure 3-5: <REFERENCE>(xmp) Power-Up Sequence, Part 1 of 2**



NOTE: The second determination of the boot processor occurs even if the original boot processor passes all memory and multiprocessor tests.

msb-0047A-90

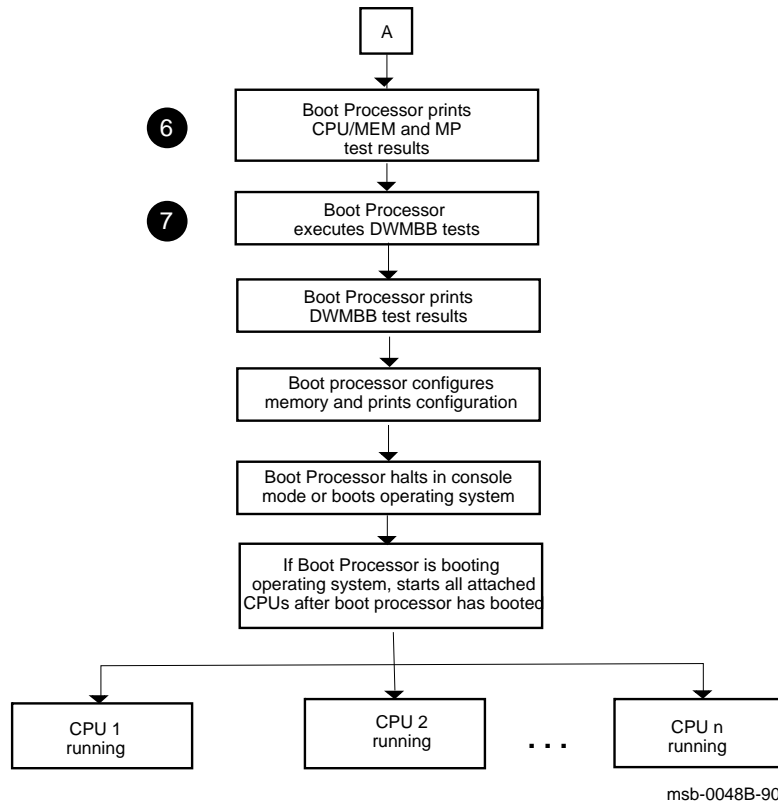
- ① All CPUs execute their on-board self-tests at the beginning of the power-up tests. On line **STF** of the self-test display, a plus sign (+) is shown for every module whose self-test passes (see Section 2.2.1).
- ② The boot processor is determined as described in Section 3.4. On the first **BPD** line, the letter **B** corresponds to the processor selected as boot processor. Because the processors have not yet completed their power-up tests, the designated processor may later be disqualified from being boot processor. For this reason, line **BPD** appears twice in the self-test display.
- ③ The boot processor prints the results of self-test, lines **NODE**, **TYP**, **STF**, and **BPD** on the self-test display. The boot processor then signals all CPUs to start running the extended test.
- ④ All CPUs execute an extended test using the memories. On line **ETF** of the self-test display, a plus sign (+) is shown for every module that passes extended test.
- ⑤ If all CPUs pass the extended test, the original boot processor selection is still valid. Lines **STF** and **ETF** would be identical for all the processors.

The yellow LED and the top two red LEDs are lit on all processor modules that pass both power-up tests. On the secondary processors, the bottom red LED is also lit. On the boot processor, this red LED is off (see Figure 2-3).

If the original boot processor fails the extended test (indicated by a minus sign (-) on line **ETF**), a new boot processor is selected. On the second **BPD** line, the letter **B** corresponds to the processor finally selected as boot processor.



Figure 3-6: <REFERENCE>(xmp) Power-Up Sequence, Part 2 of 2



- ⑥ The boot processor prints line **ETF** and the second **BPD** line of the self-test display. If none of the processors is successfully selected as the boot processor, no self-test results are displayed and the console hangs. You can identify this hung state by examining the LEDs on the processor modules (see Section 2.2.3). All yellow LEDs will be OFF. The group of eight red LEDs indicate the failing test number in binary-coded decimal.
- ⑦ The boot processor tests the DWMBB. Test results are indicated on the lines labeled **XBI** on the self-test display. A plus sign (+) at the extreme right means that the adapter test passed; a minus sign (-) means that the test failed.

## 3.6 ROM-Based Diagnostics

The ROM-based diagnostics that test the KA65A are listed in Table 3-2. See Sections 2.3 and 2.4 for instructions on running RBDs.

**Table 3-2: KA65A ROM-Based Diagnostics**

<b>Diagnostic</b>	<b>Name</b>
0	KA65A/FV64A self-test
1	KA65A/FV64A CPU/memory interaction tests
4	KA65A cache tests
5	KA65A/FV64A multiprocessor interaction tests

The KA65A diagnostic ROM contains six diagnostics, four of which test the KA65A. (The other two test the DWMBB I/O adapter and MS65A memory.) You run these diagnostics using the boot processor's RBD monitor program described in Section 2.3. Descriptions of these diagnostics are in Section 2.4.

## 3.7 VAX/DS Diagnostics

The <REFERENCE>(xmp) software diagnostics that run under the VAX Diagnostic Supervisor (VAX/DS) are listed in Table 3-3. An example follows. See Section 2.5 for instructions on running the supervisor.

**Table 3-3: <REFERENCE>(xmp) VAX/DS Diagnostics**

<b>Program</b>	<b>Description</b>
EVSBAs	VAX Standalone Autosizer
EVKAQ	VAX Basic Instructions Exerciser, Part 1
EVKAR	VAX Basic Instructions Exerciser, Part 2
EVKAS	VAX Floating Point Instruction Exerciser, Part 1
EVKAT	VAX Floating Point Instruction Exerciser, Part 2
EVKAU	VAX Privileged Architecture Instruction Test, Part 1
EVKAV	VAX Privileged Architecture Instruction Test, Part 2
EVUCA	VAX 6000 EEPROM Update Utility
EMKAX	Manual Tests

**Example 3-2: VAX/DS Commands for Running Standalone Processor Diagnostics**

```
DS> RUN EVSBA ①  
DS> SEL KA0 ②  
DS> RUN EMKAX ③  
DS> EXIT ④
```

The callouts in Example 3-2 are explained below:

- ① Run the standalone autosizer; then you do not need to attach devices to the supervisor explicitly. However, if you want to know how to use the ATTACH command for a specific diagnostic, enter:

```
DS> HELP diagnostic_name ATTACH
```

- ② The instruction and manual tests run on the boot processor. If the boot processor is the CPU with the lowest <REFERENCE>(XMI) node number (which is usually the case), issue the command to select KA0. The Diagnostic Supervisor numbers the processors consecutively. For example, if the <REFERENCE>(xmp) module with the second-lowest <REFERENCE>(XMI) node number were boot processor, you would select KA1.
- ③ This example runs the manual tests (EMKAX), which include powerfail, machine check, restart, and EEPROM functions. The diagnostic prints messages, and you must manually intervene using console switches.
- ④ Exit from VAX/DS.

## 3.8 Machine Checks

Figure 3-7 and Table 3-4 show parameters for machine checks. The <REFERENCE>(xmp) machine check parse tree appears in Appendix D along with parse trees for hard and soft error interrupts.

Figure 3-7: The Stack in Response to a Machine Check

Table 3-4: Machine Check Parameters

Parameter	Value (hex) or Bit	Description
Byte Count (SP)	18	Size of stack frame in bytes, not including PSL, PC, or byte count longword
Machine check code (SP+4)	01	Floating-point operand or result transfer error
	02	Floating-point reserved instruction
	03	Floating-point operand parity error
	04	Floating-point unknown status error

**Table 3–4 (Cont.): Machine Check Parameters**

<b>Parameter</b>	<b>Value (hex) or Bit</b>	<b>Description</b>
	05	Floating-point returned result parity error
	08	Translation buffer miss in ACV/TNV microflow
	09	Translation buffer hit in ACV/TNV microflow
	0A	Undefined INT.ID value
	0B	Undefined MOVCx state
	0C	Undefined instruction trap code
	0D	Undefined control store address
	10	Cache read tag/data parity error
	11	DAL bus or data parity read error
	12	DAL bus error on write or clear write buffer
	13	Undefined bus error microtrap
	14	Vector module error
	15	Instruction stream read error
Virtual address (SP+8)	<31:0>	Current contents of VAP register
Virtual instruction buffer address (SP+C)	<31:0>	Current virtual instruction buffer address
Interrupt state (SP+10)	<22>	ICCS bit <6>
	<15:1>	SISR bits <15:1>
Internal state (SP+14)	<31:24>	Difference between current PC and opcode PC
	<20:18>	Address of last memory reference
	<17:16>	Data length of last memory reference
	<15:8>	Opcode
	<3:0>	Last GPR referenced by E-box
Internal register (SP+18)	<31:0>	
Program counter (SP+1C)	<31:0>	PC at the start of the current instruction
Processor status longword (SP+20)	<31:0>	Current contents of PSL



## 3.9 Console Commands

**Table 3-5 summarizes the console commands. The VAX 6000 Series Owner's Manual gives a full description of these commands, their qualifiers, and examples.**

**Table 3-5: Console Commands**

<b>Command</b>	<b>Function</b>
BOOT	Initializes the system, causing a self-test, and begins the boot program.
CLEAR EXCEPTION	Cleans up error state in XBER and RCSR registers.
CONTINUE	Begins processing at the address where processing was interrupted by a CTRL/P console command.
DEPOSIT	Stores data in a specified address.
EXAMINE	Displays the contents of a specified address.
FIND	Searches main memory for a page-aligned 256-Kbyte block of good memory or for a restart parameter block.
HALT	Null command; no action is taken since the processor has already halted in order to enter console mode.
HELP	Prints explanation of console commands.
INITIALIZE	Performs a system reset, including self-test.
REPEAT	Executes the command passed as its argument.
RESTORE EEPROM	Copies the TK tape's EEPROM contents to the EEPROM of the processor executing the command. (Valid only for systems that have a TK tape.)
SAVE EEPROM	Copies to the TK tape the contents of the EEPROM of the processor executing the command. (Valid only for systems that have a TK tape.)
SET BOOT	Stores a boot command by a nickname.
SET CPU	Specifies eligibility of processors to become the boot processor or whether the vector processor is to be included in the system configuration.
SET LANGUAGE	Changes the output of the console error messages between numeric code only (international mode) and code plus explanation (English mode).

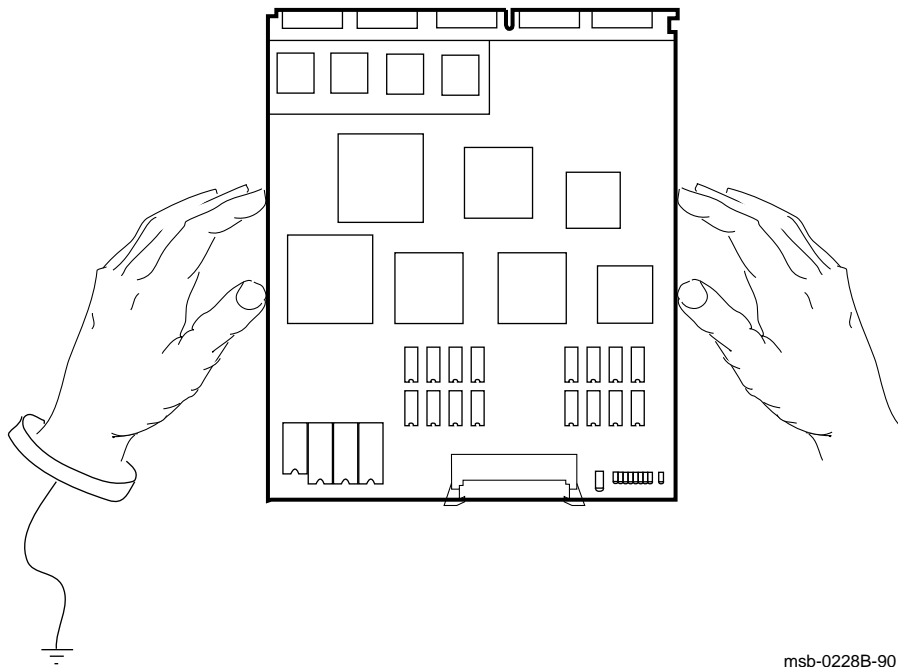
**Table 3–5 (Cont.): Console Commands**

<b>Command</b>	<b>Function</b>
SET MEMORY	Designates the method of interleaving the memory modules; supersedes the console program's default interleaving.
SET TERMINAL	Sets console terminal characteristics.
SHOW ALL	Displays the current value of parameters set.
SHOW BOOT	Displays all boot commands and nicknames that have been saved using SET BOOT.
SHOW CONFIGURATION	Displays the hardware device type and revision level for each XMI and VAXBI node and indicates self-test status.
SHOW CPU	Identifies the primary processor and the status of other processors.
SHOW ETHERNET	Locates all Ethernet adapters on the system and displays their addresses.
SHOW FIELD	Displays saved boot commands, console terminal parameters, console language mode, memory configuration, type of power system, and system serial number.
SHOW LANGUAGE	Displays the mode currently set for console error messages, international or English.
SHOW MEMORY	Displays the memory lines from the system self-test, showing interleave and memory size.
SHOW TERMINAL	Displays the baud rate and terminal characteristics functioning on the console terminal.
START	Begins execution of an instruction at the address specified in the command string.
STOP	Halts the specified node.
TEST	Passes control to the self-test diagnostics; /RBD qualifier invokes ROM-based diagnostics.
UPDATE	Copies contents of the EEPROM on the processor executing the command to the EEPROM of another processor.
Z	Logically connects the console terminal to another processor on the <REFERENCE>(XMI) bus or to a VAXBI node.
!	Introduces a comment.

### 3.10 <REFERENCE>(xmp) Handling Procedures

**Handle the KA65A modules with care. The technology used on this module is more vulnerable to static than past technology. Also, these modules have 25 mil leads to the chips; these leads are very small, close together, and easily bent.**

Figure 3-8: Holding the <REFERENCE>(xmp) Module



msb-0228B-90

The <REFERENCE>(xmp) module requires careful handling. Prepare yourself and the work area before handling these modules. Roll up your sleeves and remove any jewelry. Figure 3–8 shows the proper way to hold the module.

Follow these handling procedures to avoid damaging the <REFERENCE>(xmp) module:

1. Always wear an antistatic wrist strap.
2. Before removing the module from its ESD box, place the box on a clean, stable surface.

Be sure the box will not slide or fall. **Never** place the box on the floor. And be sure no tools, papers, manuals, or anything else that might damage the module is near it. Some components on this module can be damaged by a 600-volt static charge; paper, for example, can carry a charge of 1000 volts.

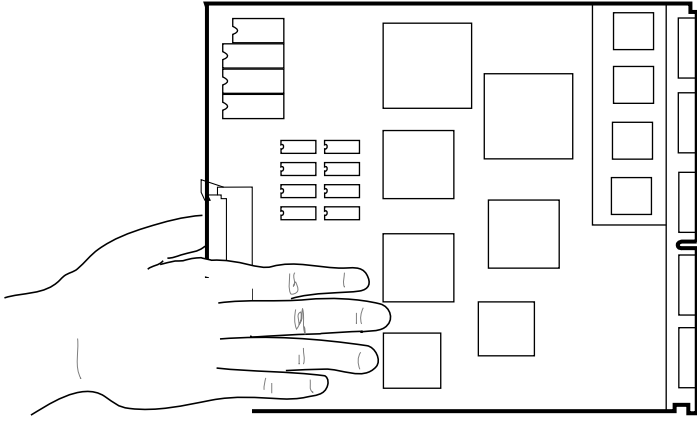
3. Hold the module only by the edges, as shown in Figure 3–8.

Do not hold the module so that your fingers touch any 25 mil devices, leads, or XMI fingers. Be sure you do not bend the module as you are holding it.

4. Be sure nothing touches the module surface or any of its components.

If anything touches the module, components or leads can be damaged. This includes the antistatic wrist strap, clothing, jewelry, cables, components on other modules, and anything in the work area (such as tools, manuals, or loose papers).

**Figure 3-9: Inserting the <REFERENCE>(xmp) Module in an XMI Card Cage**



msb-0219A-90

You must take special precautions when moving the <REFERENCE>(xmp) module in or out of the XMI card cage.

1. Be sure, when inserting the module in or removing it from the XMI card cage, that no part of the module comes in contact with another module or a cable.
2. When you swap out a module, temporarily place it in an unused XMI slot, if one is available, or place the module in an ESD box or on an ESD mat before you install the new module.

An unused XMI slot is the best place to leave a module that is being swapped out until it can be placed in the ESD box.

**CAUTION:** *If you temporarily leave a module in an unused XMI slot, be sure to remove the module before powering up the system.*

If you place the module on an ESD mat, make sure the mat is on a stable, uncluttered surface, with side 1 of the module facing up (the side with the heat sinks). Do not put it on the top of the system cabinet. And never slide the module across any surface. The leads on the components are fragile and can be damaged by contact with fingers or any surface.

3. Hold the XMI card cage handle while removing or inserting the module. If it is not held in place, the handle can spring down and damage the module.
4. When inserting the module in the card cage, grasp it as shown in Figure 3–9, being careful not to touch any 25 mil devices, and slide it slowly and gently into the slot.
5. **Do not attach the repair tag to the module.**

Place the repair tag in the plastic bag attached to the bottom of the ESD box. Allowing the repair tag to come in contact with the module can cause damage to a component.

## 3.11 How to Replace the Only Processor

**When replacing the processor module in a single-processor system, you must set a number of parameters. This ensures a correct EEPROM image in the new processor.**

**CAUTION:** *Special care must be taken when handling a <REFERENCE>(xmp) module. Review Section 3.10 before replacing this module.*

### Example 3–3: Replacing a Single Processor

```
#123456789 0123456789 0123456789 0123456789 012345#
F E D C B A 9 8 7 6 5 4 3 2 1 0 NODE #
      A A . . . M M . . . . . P TYP 6
      O + . . . + + . . . . . + STF 7
      . . . . . . . . . . . B BPD
      . . . . . . . . . . . + ETF 7
      . . . . . . . . . . . B BPD
. . . . . . . + + + . + . . + . XBI E +
      . . . . . A2 A1 . . . . . ILV
      . . . . . 64 64 . . . . . 128 Mb

Console = V1.00 RBDs = V1.00 EEPROM = 1.00/1.00 SN = 0000000000

?004F System serial number has not been initialized
>>> SET TERMINAL/SPEED:9600 9
>>> SET LANGUAGE ENGLISH
>>> [ESC][DEL]SET POWER
Power system>>> C
Power system read as: C

Update EEPROM? (Y or N) >>> Y
?011B Power system identification updated
>>> [ESC][DEL]SET SYSTEM SERIAL
System Serial Number>>> SG01234567
Serial number read as: SG01234567

Update EEPROM? (Y or N) >>> Y
?0073 System serial number updated

>>> BOOT 12
```

1. Turn the upper key switch to the Off position (0).
2. Set the console terminal baud rate to 1200.

**CAUTION:** See Section 3.10 for <REFERENCE>(xmp) module handling procedures.

3. Remove the defective processor module and temporarily insert it in an unused XMI slot or place it on an ESD mat.
4. Remove the new processor module from the ESD box and insert it in the XMI card cage. Place the old processor module in the ESD box.
5. Turn the lower key switch to Halt. Turn the upper key switch to Enable.
6. Check the self-test display for the processor, indicated by a P on the TYP line (usually in slot 1). See 6 in Example 3-3.
7. If the processor shows a plus sign (+) on both lines STF and ETF, it passed self-test. See 7.
8. Turn the lower key switch to Update.
9. Using the appropriate SET commands, enter the information displayed by the SHOW FIELD command (see 9). This command should have been issued at the end of the installation procedure, and a hard copy of the output should have been saved in the *Site Management Guide*. If it was not, you will need to research this information. The SHOW FIELD output for this system is shown here:

```
>>> SHOW FIELD
Saved boot specifications:

Console terminal parameters:
 /SCOPE      /SPEED: 9600 /BREAK

Console error message language mode:
 English

Memory configuration:
F  E  D  C  B  A   9  8  7  6  5  4  3  2  1  0  NODE #
.  .  .  .  .  .  A2 A1 .  .  .  .  .  .  .  .  ILV
.  .  .  .  .  .  64 64 .  .  .  .  .  .  .  .  128 Mb
 /INTERLEAVE:DEFAULT

Power system: C

System serial number: SG01234567
```

10. Install any console patches. See Section 3.14.
11. Turn the lower key switch to the Auto Start position.
12. Boot the operating system. See 12.



## 3.12 How to Replace the Boot Processor

**The boot processor is indicated by the letter B on the self-test BPD line (slot 1 in Example 3-4). If they have the same version ROMs, you can update the new processor's EEPROM from one of the secondaries.**

**CAUTION:** *Special care must be taken when handling a <REFERENCE>(xmp) module. Review Section 3.10 before replacing this module.*

### Example 3-4: Replacing Boot Processor

```
#123456789 0123456789 0123456789 0123456789 012345#
F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0   NODE #
      A   A   .   .   .   M   M   M   M   .   P   P   P   P           TYP 6
      o   +   .   .   .   +   +   +   +   .   +   +   +   +           STF 7
      .   .   .   .   .   .   .   .   .   .   .   E   D   E   B           BPD
      .   .   .   .   .   .   .   .   .   .   .   +   +   +   +           ETF 7
      .   .   .   .   .   .   .   .   .   .   .   E   D   E   B           BPD
.   .   .   .   .   .   .   +   +   +   .   +   .   .   +   .   XBI E +
      .   .   .   .   .   A4  A3  A2  A1  .   .   .   .   .           ILV
      .   .   .   .   .   64  64  64  64  .   .   .   .   .           256 Mb

Console = V1.00  RBDs = V1.00  EEPROM = 1.00/1.00  SN = SG01234567

>>> SET CPU/NOPRIMARY 1 10
>>> SHOW CPU
  Current Primary: 1
  /NOENABLED-
  /NOVECTOR_ENABLED-
  /NOPRIMARY- 1

>>> SET CPU 2 11
>>> SHOW CPU
  Current Primary: 2
  /NOENABLED-
  /NOVECTOR_ENABLED-
  /NOPRIMARY- 1

>>> UPDATE 1 13
```

1. Turn the upper key switch straight up to the Off position (0).

**CAUTION:** *See Section 3.10 for <REFERENCE>(xmp) module handling procedures.*

2. Remove the defective processor module and temporarily insert it in an unused XMI slot or set it on a static pad.
3. Remove the new processor module from the ESD box and insert it in the XMI card cage. Place the old processor module in the ESD box.
4. Turn the lower key switch to Halt.
5. Turn the upper key switch to Enable.
6. Check the self-test display for the new processor, indicated by a P on the TYP line (usually in slot 1). See 6 in Example 3-4.
7. If the processor shows a plus sign (+) on both lines STF and ETF, it passed self-test. See 7.
8. You will see the following message:

```
?0050 System serial number not initialized on primary processor
```

9. If you see the error message ?0052 (ROM revision mismatch. Secondary processor has revision *x.xx*), the new module will not be able to function as the boot processor. If you don't see this error message, go to Step 11.
10. Make the new module ineligible to be boot processor—use the console command SET CPU/NOPRIMARY. See 10. The new processor will operate as a secondary processor without problems, but you may continue to see error messages ?002D, ?0052, and ?0054 when the system is powered on or booted.  
Go to Step 15.
11. Make one of the secondary processors the boot processor temporarily, because the UPDATE command copies the boot processor's EEPROM. Then you can update the new processor. This command immediately makes the processor at node 2 the boot processor: SET CPU 2. See 11.
12. Turn the lower key switch to Update.
13. Now update the EEPROM of the new module from the temporary boot processor, using the UPDATE command. See 13. UPDATE takes several minutes to complete.
14. Turn the lower key switch to the Auto Start position.
15. Press the Restart button.

### 3.13 How to Add a New Processor or Replace a Secondary Processor

**Add a new secondary processor in a slot to the left of the boot processor.**

**CAUTION:** *Special care must be taken when handling a <REFERENCE>(xmp) module. Review Section 3.10 before replacing this module.*

#### Example 3-5: Adding or Replacing Secondary Processor

```
#123456789 0123456789 0123456789 0123456789 012345#
F  E  D  C  B  A  9  8  7  6  5  4  3  2  1  0  NODE #
      A  A  .  .  .  M  M  M  M  .  P  P  P  P      TYP 5
      +  +  .  .  .  +  +  +  +  .  +  +  +  +      STF 6
      .  .  .  .  .  .  .  .  .  .  .  E  D  E  B      BPD
      .  .  .  .  .  .  .  .  .  .  .  +  +  +  +      ETF 6
      .  .  .  .  .  .  .  .  .  .  .  E  D  E  B      BPD
      .  .  .  .  .  A4 A3 A2 A1 .  .  .  .  .  ILV
      .  .  .  .  .  64 64 64 64 .  .  .  .  .  256 Mb
Console = V1.00  RBDs = V1.00  EEPROM = 1.00/1.00 9  SN = SG01234567
>>> SET CPU/NOPRIMARY 3 8
>>> SHOW CPU
Current Primary: 1
/NOENABLED-
/NOVECTOR_ENABLED-
/NOPRIMARY- 3
>>> UPDATE 3 11
```

1. Turn the upper key switch straight up to the Off position (0).

**CAUTION:** *You must wear an antistatic wrist strap attached to the cabinet when you handle any modules.*

2. Either remove the defective secondary processor module, or find an empty slot where you can add the new processor. If you are removing a defective module, temporarily insert it in an unused slot.
3. Remove the new processor module from the ESD box and insert it in the XMI card cage. If you are replacing a processor module, place the old module in the ESD box.

4. Turn the lower key switch to Halt and the upper key switch to Enable.
5. Check the self-test display for the new processor, indicated by a P on the TYP line (in this example: slot 3). See ⑤ in Example 3–5.)
6. If the processor shows a plus sign (+) on both lines STF and ETF, it passed self-test. See ⑥.
7. If you see the error messages ?002D and ?0052, the new module will not be able to function as the boot processor. If you do not see these error messages, go to Step 9.
8. Make the new module ineligible to be boot processor—use the console command SET CPU/NOPRIMARY. See ③. The new processor will operate as a secondary processor without problems, but you may continue to see error messages ?002D, ?0052, and ?0054 when the system is powered on or booted.

Go to Step 13.

9. If you see error messages ?002D and ?0054, the EEPROM revision levels of the boot processor and new secondary processor do not match. Compare the boot processor's EEPROM revision numbers (see ⑨) to the numbers given in error message ?0054.

If the new secondary processor has a *higher* revision number than the boot processor, patch the boot processor's EEPROM (see Section 3.14).

10. Turn the lower key switch to Update.
11. Now update the EEPROM of the new module. See ⑩. UPDATE takes several minutes to complete.
12. Turn the lower key switch to the Auto Start position.
13. Press the Restart button.

## 3.14 Using EVUCA to Patch the EEPROM

**Use the EVUCA utility to patch the EEPROM in systems that have a CD server. EVUCA is run under VAX/DS in standalone mode.**

### Example 3-6: Patching the EEPROM with EVUCA

```
>>> BOOT /XMI:A /R5:110 EX0 ①
      [Self-test display prints]
Filename: ISL_LVAX
Follow Prompts
      [Diagnostic Supervisor Banner]
DS> LOAD EVUCA ②
DS> ATTACH KA65A HUB KA0 1
DS> ATTACH KA65A HUB KA1 2
DS> ATTACH KA65A HUB KA2 5
DS> ATTACH KA65A HUB KA3 6
DS> SELECT ALL
DS> SET TRACE
DS> START

.. Program: EVUCA - VAX 6000 EEPROM Update Utility, revision 0.5, 5
tests,
Testing: _KA0 _KA1 _KA2 _KA3
Booting secondary CPU 02.
Booting secondary CPU 05.
Booting secondary CPU 06.
Test 2: Load data from media
Data file? <EMUCA.BIN> ③
Searching for data file...
Data file loaded.
Looking for patch for CPU 01 - ROM 03.00 EEPROM 03.00
No patch image was found for CPU 01 - ROM 03.00 EEPROM 03.00
.
.
.
Looking for patch for CPU 05 - ROM 02.00 EEPROM 02.00
Patch image is revision 02.10
Do you really want to apply this patch [(No), Yes] YES ④

Test 3: Determine Typecodes Updated
Test 4: Update EEPROM data
Getting selectable boot primitives for CPU 05, ROM 02.00
      [Boot primitives identified]
      [I/O device types identified]
```

**Example 3-6 Cont'd on next page**

### Example 3–6 (Cont.): Patching the EEPROM with EVUCA

```
Available boot primitive space is 27F4
Please enter what boot primitive to delete by number. [1-3(D)] 2 ⑤
Boot primitives fit into allotted EEPROM area.
Secondary cpus are being updated, please wait a maximum of 20 seconds.
Updating CPU 05
Test 5: Show Boot primitives
.
.
.
The primary cpu was not updated. ⑥
Secondary CPU 05, was successfully updated.

.. End of run, 0 errors detected, pass count is 1,
   time is 24-SEP-1990 17:06:57.88
DS>
```

1. Load the latest diagnostic CD in the CD drive. The CD is labled 6000\_  
DIAG\_\*. (\* is the revision.) Then boot VAX/DS (see ① in Example 3–6).
2. Load EVUCA, attach the processors, and run EVUCA (see ②).
3. Test 2 of EVUCA selects the correct patch file for the system (see ③).  
Press Return.
4. When EVUCA finds a patch, it asks if it should be applied (see ④).
5. Test 4 creates a new updated EEPROM image in memory. Several boot  
primitives are available; they reside in either ROM or the EEPROM.  
Since all primitives may not fit, the user may be prompted to choose  
those primitives **not** wanted (see ⑤). When enough space is available,  
EVUCA continues.
6. EVUCA then prints a list of boot primitives for each processor and,  
if necessary, updates EEPROMs. It then prints a status message  
indicating which EEPROMs it updated (see ⑥).

## 3.15 <REFERENCE>(xmp) Registers

The <REFERENCE>(xmp) registers consist of the processor status longword, internal processor registers, <REFERENCE>(xmp) registers in <REFERENCE>(XMI) private space, <REFERENCE>(XMI) required registers, and 16 general purpose registers.

**Table 3–6: <REFERENCE>(xmp) Internal Processor Registers**

Register	Mnemonic	Address	Type	Class
Kernel Stack Pointer	KSP	IPR0	R/W	1
Executive Stack Pointer	ESP	IPR1	R/W	1
Supervisor Stack Pointer	SSP	IPR2	R/W	1
User Stack Pointer	USP	IPR3	R/W	1
Interrupt Stack Pointer	ISP	IPR4	R/W	1
Reserved		IPR5–IPR7		3
P0 Base	P0BR	IPR8	R/W	1
P0 Length	P0LR	IPR9	R/W	1
P1 Base	P1BR	IPR10	R/W	1
P1 Length	P1LR	IPR11	R/W	1

Key to Types:

R–Read  
W–Write  
R/W–Read/write

Key to Classes:

1–Implemented by the <REFERENCE>(xmp) (as specified in the *VAX Architecture Reference Manual*).  
2–Implemented uniquely by the <REFERENCE>(xmp).  
3–Not implemented. Read as zero; NOP on write.  
4–Access not allowed; accesses result in a reserved operand fault.  
5–Accessible, but not fully implemented; accesses yield UNPREDICTABLE results.  
6–Implemented by the FV64A vector module.  
I–The register is initialized on <REFERENCE>(xmp) reset (power-up, system reset, and node reset).

**Table 3–6 (Cont.): <REFERENCE>(xmp) Internal Processor Registers**

<b>Register</b>	<b>Mnemonic</b>	<b>Address</b>	<b>Type</b>	<b>Class</b>
System Base	SBR	IPR12	R/W	1
System Length	SLR	IPR13	R/W	1
Reserved		IPR14–IPR15		3
Process Control Block Base	PCBB	IPR16	R/W	1
System Control Block Base	SCBB	IPR17	R/W	1
Interrupt Priority Level	IPL	IPR18	R/W	1 I
Reserved		IPR19		
Software Interrupt Request	SIRR	IPR20	W	1
Software Interrupt Summary	SISR	IPR21	R/W	1 I
Reserved		IPR22–IPR23		3
Interval Clock Control and Status	ICCS	IPR24	R/W	2 I
Reserved		IPR25–IPR26		3
Time-of-Year Clock	TODR	IPR27	R/W	1
Reserved		IPR28–IPR31		
Console Receiver Control/Status	RXCS	IPR32	R/W	2 I
Console Receiver Data Buffer	RXDB	IPR33	R	2 I
Console Transmitter Control/Status	TXCS	IPR34	R/W	2 I
Console Transmitter Data Buffer	TXDB	IPR35	W	2 I
Reserved		IPR36–IPR37		3
Machine Check Error Summary	MCESR	IPR38	W	2
Reserved		IPR39		3
Accelerator Control and Status	ACCS	IPR40	R/W	2 I
Reserved		IPR41		3
Console Saved PC	SAVPC	IPR42	R	2
Console Saved PSL	SAVPSL	IPR43	R	2
Reserved		IPR44–IPR46		3



**Table 3–6 (Cont.): <REFERENCE>(xmp) Internal Processor Registers**

<b>Register</b>	<b>Mnemonic</b>	<b>Address</b>	<b>Type</b>	<b>Class</b>
Translation Buffer Tag	TBTAG	IPR47	W	2
Reserved		IPR48–IPR54		3
I/O Reset	IORESET	IPR55	W	2
Memory Management Enable	MAPEN	IPR56	R/W	1 I
Translation Buffer Invalidate All	TBIA	IPR57	W	1
Translation Buffer Invalidate Single	TBIS	IPR58	W	1
Translation Buffer Data	TBDATA	IPR59	W	2
Reserved		IPR60–IPR61		3
System Identification	SID	IPR62	R	1
Translation Buffer Check	TBCHK	IPR63	W	1
Reserved		IPR64–IPR111		3
Backup Cache Index	BCIDX	IPR112	R/W	2
Backup Cache Status	BCSTS	IPR113	R/W	2 I
Backup Cache Control	BCCTL	IPR114	R/W	2 I
Backup Cache Error Address	BCERA	IPR115	R	2
Backup Cache Tag Store	BCBTS	IPR116	R/W	2
Backup Cache Deallocate Tag	BCEDET	IPR117	W	2
Backup Cache Error Tag	BCERT	IPR118	R	2
Reserved		IPR119–IPR122		
Vector Interface Error Status	VINTSR	IPR123	R/W	2
Primary Cache Tag Store	PCTAG	IPR124	R/W	2
Primary Cache Index	PCIDX	IPR125	R/W	2
Primary Cache Error Address	PCERR	IPR126	R/W	2
Primary Cache Status	PCSTS	IPR127	R/W	2 I
Reserved		IPR128–IPR143		3
Vector Processor Status	VPSR	IPR144	R/W	6

**Table 3–6 (Cont.): <REFERENCE>(xmp) Internal Processor Registers**

<b>Register</b>	<b>Mnemonic</b>	<b>Address</b>	<b>Type</b>	<b>Class</b>
Vector Arithmetic Exception	VAER	IPR145	R	6
Vector Memory Activity Check	VMAC	IPR146	R	6
Vector Translation Buffer Invalidate All	VTBIA	IPR147	W	6
Reserved		IPR148–IPR156		5
Vector Indirect Register Address	VIADR	IPR157	R/W	6
Vector Indirect Data Low	VIDLO	IPR158	R/W	6
Vector Indirect Data High	VIDHI	IPR159	R/W	6

The IPRs are explicitly accessible to software only by the Move To Processor Register (MTPR) and Move From Processor Register (MFPR) instructions, which require kernel mode privileges. From the console, EXAMINE/I and DEPOSIT/I commands read and write the IPRs.

**Table 3–7: <REFERENCE>(XMI) Registers for the <REFERENCE>(xmp)**

<b>Register</b>	<b>Mnemonic</b>	<b>Address</b>
XMI Device	XDEV	BB + 00
XMI Bus Error	XBER	BB + 04
XMI Failing Address	XFADR	BB + 08
XMI General Purpose	XGPR	BB + 0C
Node-Specific Control and Status	NSCSR	BB + 1C
XMI Control 0	XCR0	BB + 24
XMI Failing Address Extension	XFAER	BB + 2C
XMI Bus Error Extension	XBEER	BB + 34
Writeback Failing Address 0	WFADR0	BB + 40
Writeback Failing Address 1	WFADR1	BB + 44

**Note:** "BB" = base address of an XMI node, which is the address of the first location in nodespace.

**Table 3–8: <REFERENCE>(xmp) Registers in <REFERENCE>(XMI) Private Space**

<b>Register</b>	<b>Mnemonic</b>	<b>Address</b>
Control Register 0	CREG0	none
Control Register 1	CREG1	none
Control Register Write Enable	CREGWE	E000 0000
Console ROM (halt protected)		E004 0000 to E009 FFFF
Console EEPROM (halt protected)		E00A 0000 to E00A 7FFF
Console ROM (not halt protected)		E00C 0000 to E011 FFFF
Console EEPROM (not halt protected)		E012 0000 to E012 7FFF
MSSC Base Address	SSCBAR	E014 0000
MSSC Configuration	SSCCNR	E014 0010
MSSC Bus Timeout Control	SSCBTR	E014 0020
MSSC Output Port	OPORT	E014 0030
MSSC Input Port	IPORT	E014 0040
Control Register Base Address	CRBADR	E014 0130
Control Register Address Decode Mask	CRADMR	E014 0134
EEPROM Base Address	EEBADR	E014 0140
EEPROM Address Decode Mask	EEADMR	E014 0144
Timer 0 Control	TCR0	E014 0160
Timer 0 Interval	TIR0	E014 0164
Timer 0 Next Interval	TNIR0	E014 0168
Timer 0 Interrupt Vector	TIVR0	E014 016C
Timer 1 Control	TCR1	E014 0170
Timer 1 Interval	TIR1	E014 0174
Timer 1 Next Interval	TNIR1	E014 0178
Timer 1 Interrupt Vector	TIVR1	E014 017C
MSSC Interval Counter	SSCICR	E014 01F8
MSSC Internal RAM		E014 0400 to E014 07FF

**Table 3–8 (Cont.): <REFERENCE>(xmp) Registers in <REFERENCE>(XMI) Private Space**

<b>Register</b>	<b>Mnemonic</b>	<b>Address</b>
MAXMI DAL Diagnostic	DCSR	E100 0000
MAXMI Failing DAL 0	FDAL0	E100 0020
MAXMI Failing DAL 1	FDAL0	E100 0028
MAXMI Failing DAL 2	FDAL0	E100 0030
MAXMI Failing DAL 3	FDAL0	E100 0038
MAXMI RAM		E100 8000 to E100 9FFF
IP IVINTR Generation	IPINTR	E101 0000 to E101 FFFF
WE IVINTR Generation	WEINTR	E102 0000 to E102 FFFF

# FV64A Vector Processor

---

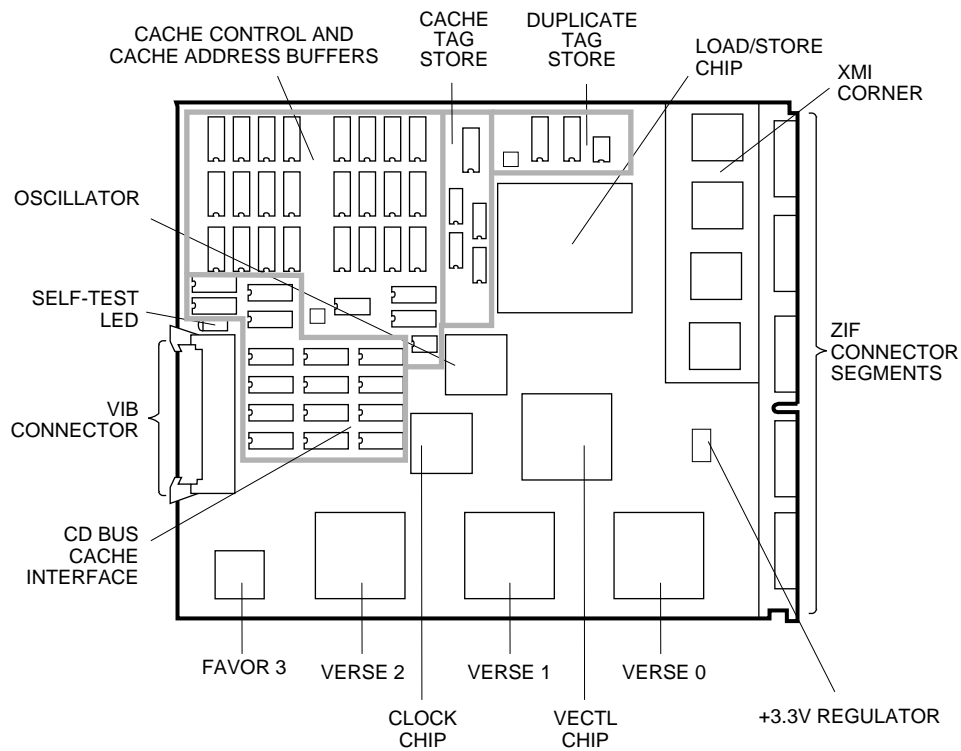
This chapter contains the following sections:

- <REFERENCE>(xrv) Physical Description and Specifications
- <REFERENCE>(xmp)/FV64A Coprocessors
- Configuration Rules
- Functional Description
- Self-Test Results: Console Display and Self-Test LED
- Self-Test Results: Scalar XGPR Register
- ROM-Based Diagnostics
- VAX/DS Diagnostics
- Machine Checks
- Vector Console Commands
- FV64A Handling Procedures
- How to Replace a Vector Module
- Vector Processor Registers

## 4.1 <REFERENCE>(xrv) Physical Description and Specifications

The <REFERENCE>(XRV) is a vector processor used with the <REFERENCE>(xmp) scalar processor. The module designation is T2017. The two processor modules are connected with a VIB cable. Figure 4-1 shows side 1 of the module, and Figure 4-2 shows side 2.

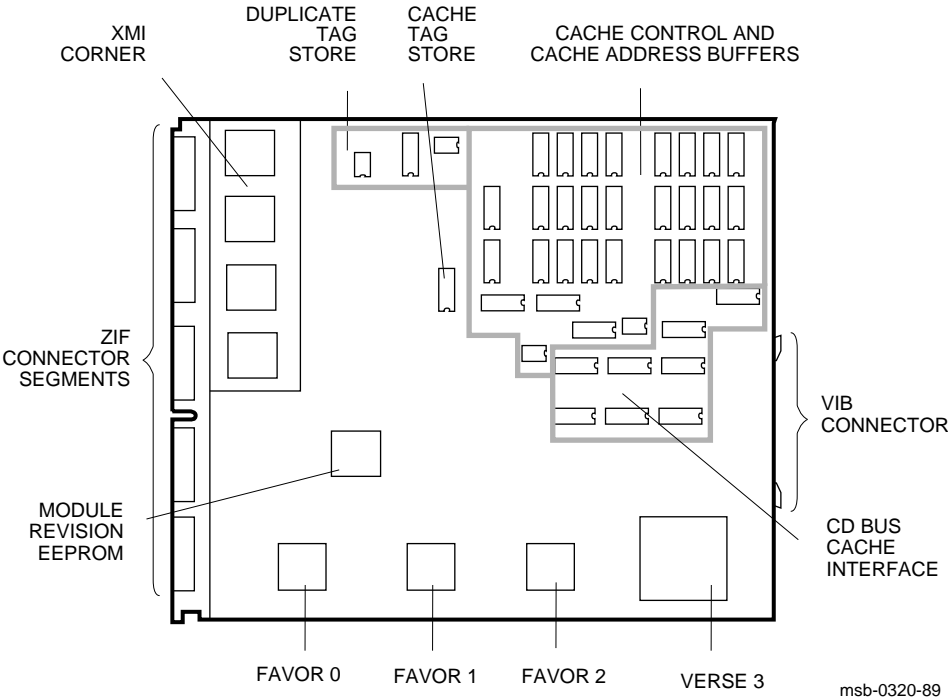
Figure 4-1: <REFERENCE>(XRV) Module (Side 1)



msb-0319-89

**Because the vector module has components on side 2, only memory modules can be installed next to side 2 (see Figure 4-2).**

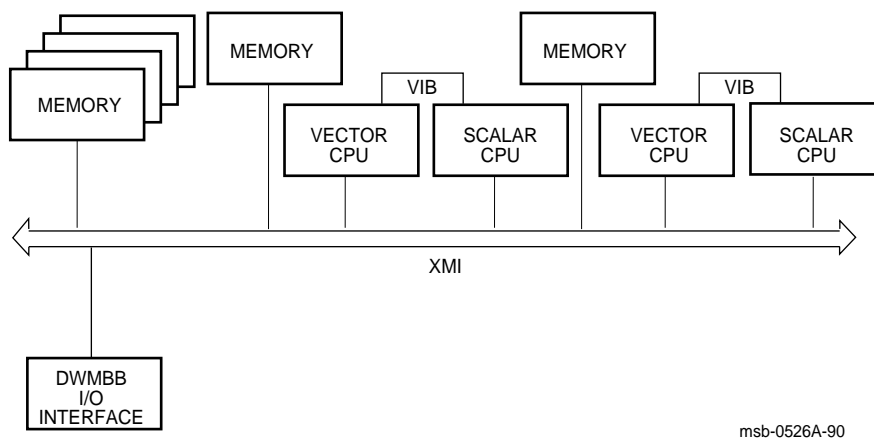
**Figure 4-2: <REFERENCE>(XRV) Module (Side 2)**



## 4.2 <REFERENCE>(xmp)/FV64A Coprocessors

The <REFERENCE>(XX) uses a high-speed system bus, called the <REFERENCE>(XMI) bus, to interconnect its processors and its memory modules. In Figure 4-3 all I/O devices connect to the VAXBI bus. The <REFERENCE>(XX) supports multiprocessing with up to six scalar processors or one or two scalar/vector pairs.

Figure 4-3: VAX 6000 Model 500 Vector Processing System





**Table 4–1: <REFERENCE>(XRV) Specifications**

<b>Parameter</b>	<b>Description</b>
<b>Module Number:</b>	T2017
<b>Dimensions:</b>	23.3 cm (9.2") H x 0.23 cm (0.093") W x 28.0 cm (11.0") D
<b>Temperature:</b>	
Storage Range	-40°C to 66°C (-40°F to 151°F)
Operating Range	10°C to 40°C (41°F to 122°F)
<b>Relative Humidity:</b>	
Storage	10% to 95% noncondensing
Operating	10% to 95% noncondensing
<b>Altitude:</b>	
Storage	Up to 4.8 km (16,000 ft)
Operating	Up to 2.4 km (8000 ft)
<b>Current:</b>	14A at +5V
<b>Power:</b>	70W
<b>Cables:</b>	VIB cable, 17-02240-03
<b>Diagnostics:</b>	ROM-based diagnostics 0, 1, and 5 VAX/DS diagnostics, see Section 4.8

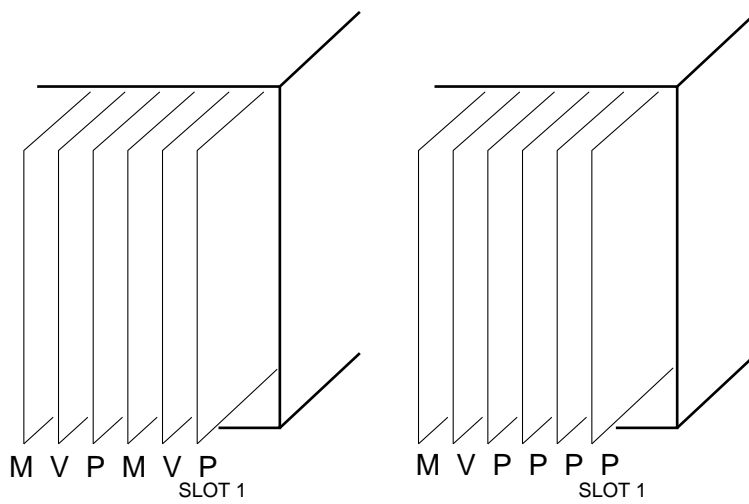
The FV64A vector processor is an integrated vector processor; that is, the vector processor module performs as a coprocessor that is tightly coupled with a host scalar processor. The two processors are physically connected by an intermodule cable, the VIB. The scalar processor is specifically designed to support its vector coprocessor, and the vector instruction set is implemented as part of the host native instruction set. Both the scalar and vector processors are on the XMI bus, and they share a common memory.

A <REFERENCE>(vax\_mod\_xxx) system can have one or two scalar/vector pairs. If the system has only one pair, it can also have additional scalar processors. For optimal performance, two memory modules of the same size are required for one scalar/vector pair, and four memory modules are required for two scalar/vector pairs.

### 4.3 <REFERENCE>(xrv) Configuration Rules

**A vector processor must be installed to the left of its companion scalar processor. An intermodule cable connects the two modules. A memory module or an empty slot must be to the left of the vector processor. Any other configuration may damage the vector module.**

Figure 4-4: Scalar/Vector Configurations



TWO SCALAR/VECTOR PAIRS

ONE SCALAR/VECTOR PAIR

KEY:  
M= MEMORY  
V= VECTOR PROCESSOR  
P= SCALAR PROCESSOR

msb-0373-90

Table 4–2 shows the maximum number of scalar and vector processors supported in a <REFERENCE>(vax\_mod\_xxx) system.

**Table 4–2: Processor Module Combinations**

Maximum Scalar Processors	Maximum Vector Processors	Configuration (Slot 1 at Right)
6	0	P P P P P P
4	1	M V P P P P
2	2	M V P M V P

Figure 4–4 shows system configurations for a <REFERENCE>(vax\_mod\_xxx) system with one or two vector processors. The left side of the figure indicates the configuration for two scalar/vector pairs with a memory module in the slot to the left of the vector processor. The right side of the figure shows a single scalar/vector pair with additional scalar processors.

Processor modules are configured after I/O adapters. (I/O adapters are installed, from left to right, in slots E to A and then 5 to 1.) Processors are configured from right to left, filling available slots starting with slot 1. Memories are configured last, from left to right, filling available slots from 9 to 1. However, in a system with a vector processor, the modules should be installed as shown in Figure 4–4. These configurations must be followed to avoid damage to the modules and for performance reasons:

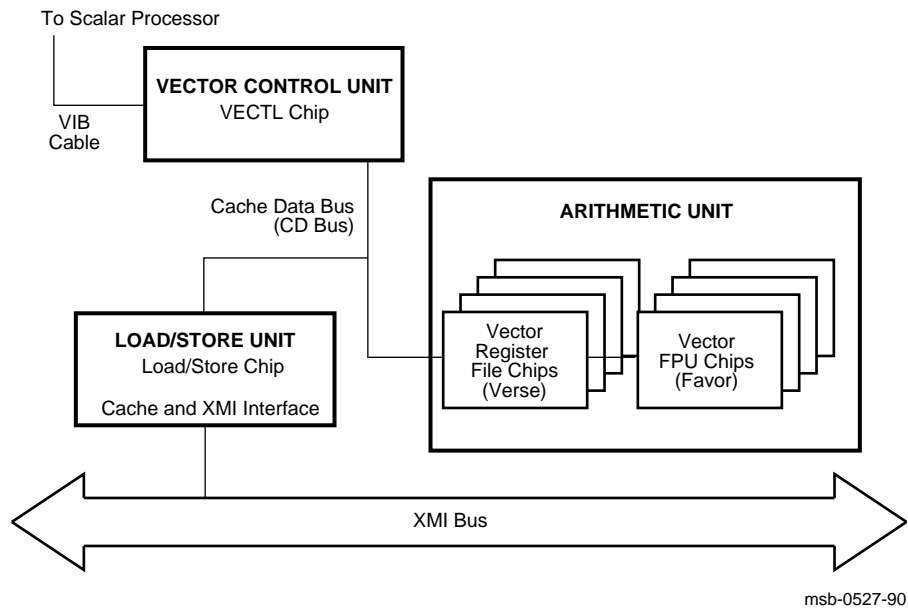
- Because the <REFERENCE>(xrv) module has VLSI components with heat sinks protruding from both sides, only a memory module, with its low components, can be placed next to side 2 of the <REFERENCE>(xrv) module.
- In a system with one scalar/vector pair and one or more additional scalar processors, the scalar processor of the pair should be prevented from being the boot processor for performance reasons.

If the scalar/vector pair is to the left of other scalar processors, then the processor of the scalar/vector pair will not become the boot processor unless other processors fail self-test or have been disabled with the SET CPU console command. Alternatively, you can issue the SET CPU/NOPRIMARY command and give the node number of the attached scalar processor that you do not want to be the boot processor.

## 4.4 <REFERENCE>(xrv) Functional Description

**Figure 4-5 shows the three main functional units of the <REFERENCE>(XRV) processor: the vector control unit, the arithmetic unit, and the load/store unit, which includes the XMI interface and cache control.**

**Figure 4-5: <REFERENCE>(XRV) Block Diagram**



The <REFERENCE>(xrv) is an integrated vector processor, tightly coupled to the <REFERENCE>(xmp) scalar processor. The vector instructions are issued from the scalar processor, and the vector processor then dispatches them internally. All communication between the scalar and vector modules takes place across the intermodule VIB cable. All communication with memory is over the XMI bus.

The vector processor has 16 vector data registers, each 64 quadwords long. It also has a 1-megabyte direct-mapped cache and a 136-entry translation buffer.

The <REFERENCE>(xrv) is an XMI module with the standard XMI Corner. The module has a cable connector at the rear edge of the module that connects to the rear edge of a <REFERENCE>(xmp) module. The instructions are issued over the VIB bus and pass to the VECTL chip, which then controls the operations on the module. It passes instructions to the load/store unit over the CD bus. The load/store unit then issues XMI memory transactions. The VECTL chip also issues instructions to the four pairs of Verse and Favor chips that make up the arithmetic unit. The vector data registers are in the Verse chips. The Favor chips perform the arithmetic operations on the data held in the Verse chips.

The vector processor module uses the standard XMI Corner interface, but it functions only as an XMI commander. The vector processor does not issue transactions to I/O space, nor does it respond to XMI transactions directed to it. All error reporting is done by the scalar processor.

## 4.5 Self-Test Results: Console Display and Self-Test LED

You can check the vector processor self-test results in three ways: the self-test display if the <REFERENCE>(xrv) module is attached to the <REFERENCE>(xmp) module in node 1, the yellow self-test LED on the <REFERENCE>(xrv) module, and the contents of the XGPR register of the attached <REFERENCE>(xmp) module. If self-test passes, the large yellow LED on the <REFERENCE>(xrv) module lights. If the <REFERENCE>(xrv) module fails self-test, the light remains unlit.

### Example 4–1: Self-Test Results

```
#123456789 0123456789 0123456789 0123456789 0123456789 01234567# ①
F  E  D  C  B  A  9  8  7  6  5  4  3  2  1  0  NODE #
      A  A  .  .  .  M  M  .  M  V- -P  M  V- -P  TYP ②
      +  +  .  .  .  +  +  .  +  +  +  +  +  +  STF ③
      .  .  .  .  .  .  .  .  .  D  E  .  E  B  BPD ④
      .  .  .  .  .  .  .  .  .  +  +  +  +  +  ETF ⑤
      .  .  .  .  .  .  .  .  .  D  E  .  E  B  BPD ⑥
      .  .  .  .  .  .  A4  A3  .  A2  .  .  A1  .  .  ILV
      .  .  .  .  .  .  32  32  .  32  .  .  32  .  .  128Mb
Console = V1.00  RBDs = V1.00  EEPROM = 1.00/1.00  SN = SG01234567 ⑦
>>>
```

Example 4–1 shows the self-test results for a system with two scalar/vector pairs. Each <REFERENCE>(xmp) runs its self-test and then tests any attached vector processor.

- ① The first line of the self-test printout is the progress trace. This line shows the self-test progress of the <REFERENCE>(xmp) in node 1 (the baud rate must be at least 1200). The numbers correspond to tests in the system self-test. If there is an attached vector processor module and self-test passes, the line prints as in Example 4–1 ending with #. If there is no attached vector processor, testing stops after the first 45 tests. If a test fails, the failing test number is the last one printed. For example, if test 14 fails, the line is printed as follows:

```
#123456789 01234
```

- ② This line indicates the type (TYP) of module at each <REFERENCE>(XMI) node. Scalar processors are type P, and vector processors are type V. The dashes indicate that the vector processors are attached to the adjacent scalar processors.
- ③ This line shows self-test fail status (STF), which are the results of on-board self-test. Possible values for processors are:
  - + (pass)
  - (fail)

All processors passed self-test in this example.

- ④ The BPD line indicates boot processor designation and whether vector processors are enabled or disabled. When the system completes on-board self-test, the scalar processor with the lowest XMI ID number that passes self-test and that is eligible is selected as boot processor — in this example, the processor at node 1.

The results on the BPD line indicate:

- The boot processor (B)
- Scalar processors eligible (E) or ineligible (D) to become the boot processor
- Vector processors enabled (E) or disabled (D)

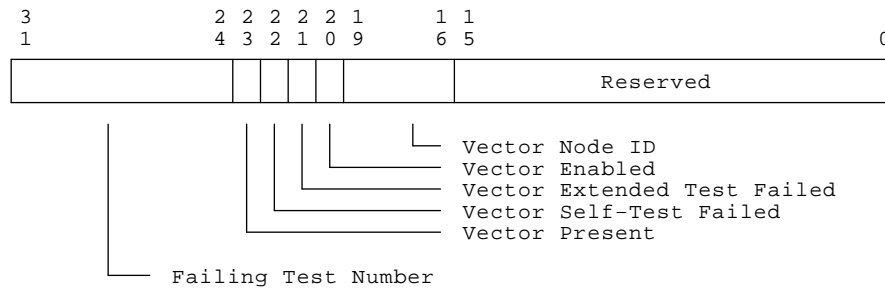
In this example the vector processor attached to the scalar processor at node 4 has been disabled. A vector processor can be disabled by the SET CPU/NOVECTOR\_ENABLED command.

- ⑤ During extended test (ETF) all processors run additional tests, which include reading and writing memory and using the cache. On line ETF, results are reported for each processor in the same way as on line STF— a plus sign indicates that extended test passed and a minus sign that extended test failed.
- ⑥ Another BPD line is displayed, because it is possible for a different CPU to be designated boot processor if the processor first designated as the boot processor fails the extended testing.
- ⑦ The last line of the self-test display shows the ROM and EEPROM version numbers and the system serial number.

## 4.6 Self-Test Results: Scalar XGPR Register

**You can check self-test results in the self-test display or in the XGPR register. The failing test number is left in the upper byte of the XGPR register of the failing <REFERENCE>(xmp) module.**

**Figure 4–6: XGPR Register**



### Example 4–2: XGPR Register After Power-Up Test Failure

```
>>> E/P/L E188000C      ! Examine the longword at physical address
                        ! E188000C, the address of the XGPR
E188000C 45F0xxxx      ! register of the processor in slot 1.
                        ! The result indicates that test 45 of
                        ! self-test failed (Load/Store Cache test).
```



Figure 4–6 shows the XGPR register of the scalar processor. Bit <23>, when set, indicates that there is a vector processor attached to this processor. Bits <22:16> give status on an attached vector processor.

The failing test number is derived from the upper byte (bits <31:24>) of the longword returned. For self-test, the upper byte contains the failing test number. If CPU/memory interaction test fails, this byte contains the failing test number plus 60. If a multiprocessor test fails, this byte contains the failing test number plus 90. All numbers are expressed in binary-coded decimal (BCD). See Table 4–3.

As shown in Example 4–2, you can examine the XGPR register of the failing <REFERENCE>(xmp) module to determine the failing test number. See Table 2–5 to determine the base address (BB) of the <REFERENCE>(xmp) processor’s node. Then calculate the address of the XGPR register by adding 0C (hex) to the base address.

**Table 4–3: Interpreting XGPR Failing Test Numbers**

<b>Failing Diagnostic</b>	<b>XGPR &lt;31:24&gt; (BCD)</b>	<b>Test Numbers</b>
Self-test	1–57	1–57
CPU/memory interaction test	61–82	1–22
Additional memory test	83–89	3
Multiprocessor tests	91–97	1–7

## 4.7 ROM-Based Diagnostics

The ROM-based diagnostics that test the FV64A are listed in Table 4-4. See Sections 2.3 and 2.4 for instructions on running RBDs.

**Table 4-4: FV64A ROM-Based Diagnostics**

<b>Diagnostic</b>	<b>Name</b>
0	KA65A and FV64A self-test
1	KA65A and FV64A CPU/memory interaction tests
5	KA65A and FV64A multiprocessor interaction tests

The KA65A diagnostic ROM contains six diagnostics, three of which test the FV64A. (The other three test the KA65A cache, the DWMBB I/O adapter, and the MS65A memory.) You run these diagnostics using the boot processor's RBD monitor program described in Section 2.3. Descriptions of these diagnostics are in Section 2.4.

## 4.8 VAX/DS Diagnostics

The <REFERENCE>(XRv) diagnostics that run under the VAX Diagnostic Supervisor (VAX/DS) are listed in Table 4-5. Example 4-3 lists VAX/DS commands used in testing vector processors. See Section 2.5 for instructions on running the supervisor.

**Table 4-5: <REFERENCE>(XRv) VAX/DS Diagnostics**

Program	Description
EVKAG	VAX Vector Instruction Exerciser, Part 1 (1 1/2 min—quick) (16 min—default)
EVKAH	VAX Vector Instruction Exerciser, Part 2 (1 min—quick) (18 min—default)

### Example 4-3: VAX/DS Commands for Testing Vector Processors

```
DS> SET QUICK           ! Abbreviated version of the VAX Vector
                        ! Instruction Exerciser will be run.
DS> DESELECT KA1       ! Removes the second scalar/vector pair
                        ! from testing.
DS> RUN EVKAG           ! Part 1 of VAX Vector Instruction Exerciser.
DS> RUN EVKAH           ! Part 2.

DS> BOOT n             ! If more than one vector, make the scalar
DS> DESELECT KA0       ! processor of the second scalar/vector pair
DS> SELECT KA1         ! the boot processor. Run EVKAG and EVKAH
                        ! on the second vector processor.
                        ! Restore original boot processor.
DS> RUN EVKAG
DS> RUN EVKAH
DS> BOOT n
DS> EXIT
```

## 4.9 Machine Checks

**A machine check is an exception that indicates a processor-detected internal error. Figure 4-7 and Table 4-6 show these parameters. The <REFERENCE>(xrv) machine check parse tree appears in Appendix D along with parse trees for hard and soft error interrupts and disable faults.**

Figure 4-7: The Stack in Response to a Machine Check

Table 4-6: <REFERENCE>(xrv) Machine Check Parameters

Parameter	Value (hex)	Description
Machine check code (SP+4)	14	Vector module error

Machine checks are taken regardless of the current IPL. If the machine check exception vector bits (<1:0>) are not both one, the operation of the processor is undefined. The exception is taken on the interrupt stack and the IPL is raised to 1F (hex). See Table 3-4 for the complete list of machine check codes.

## 4.10 Vector Console Commands

**Table 3–5 gives the console commands specific to the vector processor.**

**Table 4–7: Vector Console Commands**

<b>Command</b>	<b>Function</b>
DEPOSIT	Stores data in a specified address. Additional addresses can be VMR, VCR, and VLR (for Vector Mask Register, Vector Count Register, and Vector Length Register).
/M	Defines the address space as a vector indirect register; accesses addresses 400 and higher.
/Q	Quadword is the default data size for vector registers (except for VCR and VLR).
/VE	Defines the address space as the vector register set.
EXAMINE	Displays the contents of a specified address. Additional addresses can be VMR, VCR, and VLR (for Vector Mask Register, Vector Count Register, and Vector Length Register).
/M	Defines the address space as a vector indirect register; accesses addresses 400 and higher.
/Q	Quadword is the default data size for vector registers (except for VCR and VLR).
/VE	Defines the address space as the vector register set.
SET CPU	Specifies attributes of processors, such as eligibility to become the boot processor or whether a vector processor is enabled.
/NOVECTOR_ENABLED	Prevents a vector module from being recognized in the system configuration.
/VECTOR_ENABLED	Specifies that a vector module will be recognized in the system configuration; the default.

## DEPOSIT Examples

1. >>> DEPOSIT/VE V12 0 ! Deposits zero into all 64 elements  
! of vector register V12.
  
2. >>> DEPOSIT V6:2C/n:2 0 ! Deposits zero into V6 beginning at  
! element 2C (hex) and also in the next  
! two elements.
  
3. >>> DEPOSIT VLR 1 ! Deposits one in the Vector Length  
! Register.
  
4. >>> DEPOSIT/Q/P 200 FFFFFFFF45370201  
! Deposits FFFFFFFF45370201, a quadword  
! of data into physical memory at address  
! 200.
  
5. >>> DEPOSIT/M 440 0 ! Deposits zeros to vector indirect  
! register with address 440 (hex).

## EXAMINE Examples

1. >>> EXAMINE VLR                   ! Examines the Vector Length  
          M 00000001 0E               ! Register.
  
2. >>> EXAMINE/Q/P 200               ! Examines the quadword in  
                                      ! physical memory at address 200.
  
3. >>> EXAMINE/VE V12:2E             ! Examines element 2E (hex)  
                                      ! (which is 41 decimal) of vector  
                                      ! data register V12.
  
4. >>> EXAMINE/M 440                 ! Examines the vector indirect  
                                      ! register at hex address 440.  
          M 440 FFFFFFFF 00000000   ! /M is used to access vector  
                                      ! indirect registers.



```

5. >>> EXAMINE/VE V0          ! Examines vector register V0; system
                                ! displays all 64 elements of register V0.

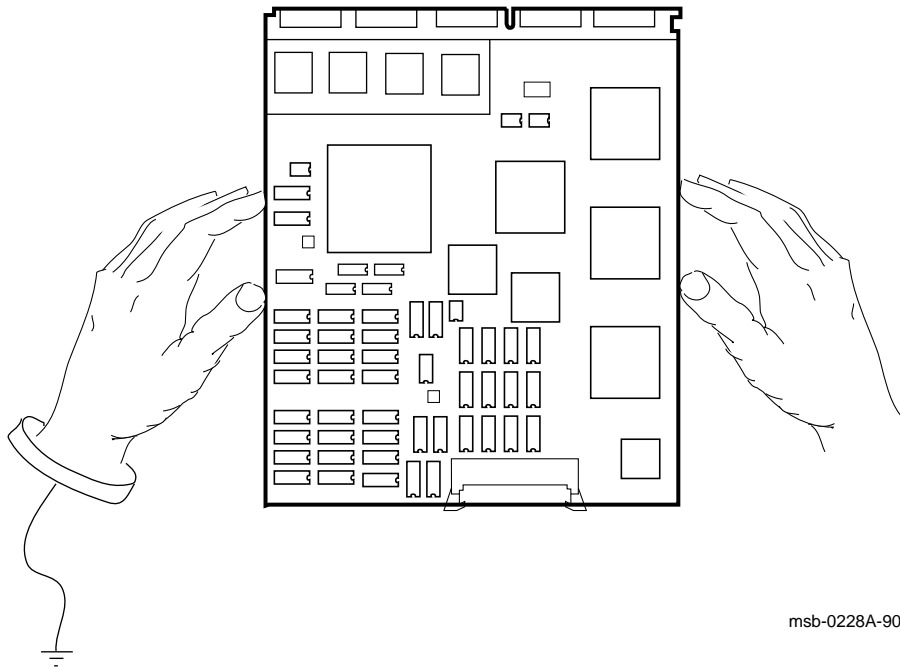
VE V00:00 00000000 00000002    VE V00:01 00000000 00000002
VE V00:02 00000000 00000002    VE V00:03 00000000 00000002
VE V00:04 00000000 00000002    VE V00:05 00000000 00000002
VE V00:06 00000000 00000002    VE V00:07 00000000 00000002
VE V00:08 00000000 00000002    VE V00:09 00000000 00000002
VE V00:0A 00000000 00000002    VE V00:0B 00000000 00000002
VE V00:0C 00000000 00000002    VE V00:0D 00000000 00000002
VE V00:0E 00000000 00000002    VE V00:0F 00000000 00000002
VE V00:10 00000000 00000002    VE V00:11 00000000 00000002
VE V00:12 00000000 00000002    VE V00:13 00000000 00000002
VE V00:14 00000000 00000002    VE V00:15 00000000 00000002
VE V00:16 00000000 00000002    VE V00:17 00000000 00000002
VE V00:18 00000000 00000002    VE V00:19 00000000 00000002
VE V00:1A 00000000 00000002    VE V00:1B 00000000 00000002
VE V00:1C 00000000 00000002    VE V00:1D 00000000 00000002
VE V00:1E 00000000 00000002    VE V00:1F 00000000 00000002
VE V00:20 00000000 00000002    VE V00:21 00000000 00000002
VE V00:22 00000000 00000002    VE V00:23 00000000 00000002
VE V00:24 00000000 00000002    VE V00:25 00000000 00000002
VE V00:26 00000000 00000002    VE V00:27 00000000 00000002
VE V00:28 00000000 00000002    VE V00:29 00000000 00000002
VE V00:2A 00000000 00000002    VE V00:2B 00000000 00000002
VE V00:2C 00000000 00000002    VE V00:2D 00000000 00000002
VE V00:2E 00000000 00000002    VE V00:2F 00000000 00000002
VE V00:30 00000000 00000002    VE V00:31 00000000 00000002
VE V00:32 00000000 00000002    VE V00:33 00000000 00000002
VE V00:34 00000000 00000002    VE V00:35 00000000 00000002
VE V00:36 00000000 00000002    VE V00:37 00000000 00000002
VE V00:38 00000000 00000002    VE V00:39 00000000 00000002
VE V00:3A 00000000 00000002    VE V00:3B 00000000 00000002
VE V00:3C 00000000 00000002    VE V00:3D 00000000 00000002
VE V00:3E 00000000 00000002    VE V00:3F 00000000 00000002

```

## 4.11 <REFERENCE>(xrv) Handling Procedures

**Handle the processor modules with care. The CMOS2 technology used on the later VAX 6000 series modules is more vulnerable to static than past technology. Also, these modules have 25 mil leads to the chips; these leads are very small, close together, and easily bent.**

Figure 4-8: Holding the <REFERENCE>(xrv) Module



The later VAX 6000 series modules require careful handling. Prepare yourself and the work area before handling these modules. Roll up your sleeves and remove any jewelry. Figure 4-8 shows the proper way to hold these modules.

Follow these handling procedures to avoid damaging the processor modules:

1. Always wear an antistatic wrist strap.
2. Before removing the module from its ESD box, place the box on a clean, stable surface.

Be sure the box will not slide or fall. **Never** place the box on the floor. And be sure no tools, papers, manuals, or anything else that might damage the module is near it. Some components on this module can be damaged by a 600-volt static charge; paper, for example, can carry a charge of 1000 volts.

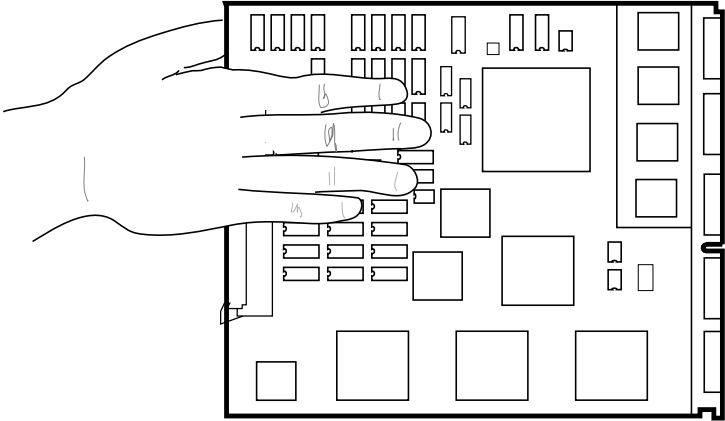
3. Hold the module only by the edges, as shown in Figure 4-8.

Do not hold the module so that your fingers touch any 25 mil devices, leads, or XMI fingers. Be sure you do not bend the module as you are holding it.

4. Be sure nothing touches the module surface or any of its components.

If anything touches the module, components or leads can be damaged. This includes the antistatic wrist strap, clothing, jewelry, cables, components on other modules, and anything in the work area (such as tools, manuals, or loose papers).

**Figure 4-9: Inserting the <REFERENCE>(xrv) Module in an XMI Card Cage**



msb-0372-90

You must take special precautions when moving the processor modules in or out of the XMI card cage.

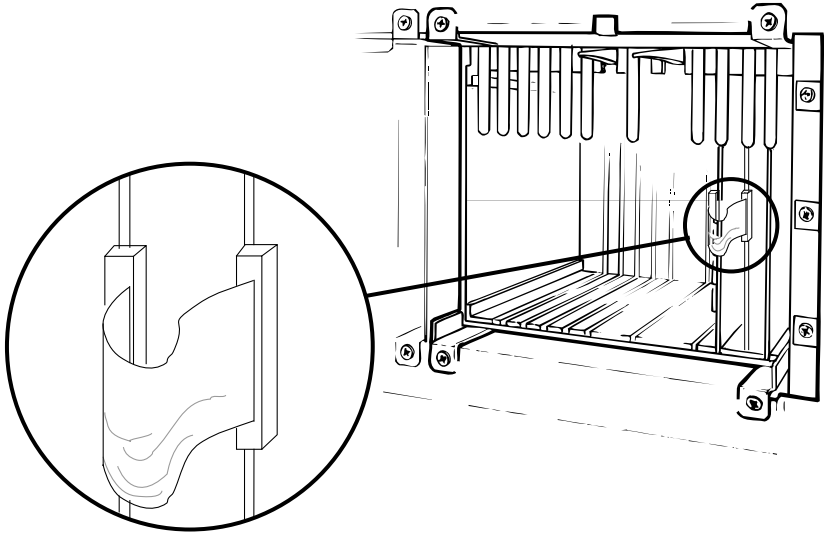
1. Be sure, when inserting the module in or removing it from the XMI card cage, that no part of the module comes in contact with another module or a cable. The leads on the components are fragile and can be damaged by contact with fingers or any surface.
2. When you swap out a module, place it in the correct ESD box before you install the new module.
3. Hold the XMI card cage handle while removing or inserting the module. If it is not held in place, the handle can spring down and damage the module.
4. When inserting the module in the card cage, grasp it as shown in Figure 4-9, being careful not to touch any 25 mil devices, and slide it slowly and gently into the slot.
5. **Do not attach the repair tag to the module.**

Place the repair tag in the plastic bag attached to the bottom of the ESD box. Allowing the repair tag to come in contact with the module can cause damage to a component.

## 4.12 How to Replace a Vector Module

**If a vector module is defective, you can replace it with a new one. If you install an additional one, see the complete installation instructions in the VAX 6000 Series Upgrade Manual.**

**Figure 4-10: Replacing a Vector Module in an XMI Card Cage**



msb-0407-90

**CAUTION:** *Special care must be taken when handling processor modules. See Section 4.11 before replacing this module. Also review the configuration rules in Section 4.3.*

*While removing or inserting a module in the XMI card cage, you must hold the XMI card cage lever. Failure to do so may result in damage to the module.*

1. Turn the upper key switch straight up to the Off position (0).
2. Open the cabinet door and remove the plastic door in front of the XMI card cage.

**CAUTION:** *You must wear an antistatic wrist strap attached to the cabinet when you handle any modules.*

3. Disconnect the VIB cable (17-02240-03) from the vector module.
4. Remove the defective vector processor module.
5. Take the new vector processor module from the ESD box and insert it in the XMI card cage. Place the defective module in the ESD box.
6. Attach the connecting VIB (vector interface bus) cable. The keyed end of the cable attaches to the vector module.
7. Press the lever down to close the connector.
8. Replace the plastic door and shut the cabinet door.
9. Turn the lower key switch to Halt and the upper key switch to Enable.
10. Check the self-test display for the new vector processor, indicated by a V on the TYP line.
11. If the processor shows a plus sign (+) on both lines STF and ETF, it passed self-test.

## 4.13 Vector Processor Registers

The <REFERENCE>(XRV) internal processor registers are listed in Table 4-8. See Chapter 3 for the complete list of IPR registers. The console program allows you to access the vector registers. Software accesses the vector registers with MTPR/MFPR and MTVP/MFVP instructions.

**Table 4-8: <REFERENCE>(XRV) Internal Processor Registers**

Register	Mnemonic	Address	Type	Class
Vector Interface Error Status	VINTSR	IPR123	R/W	1
Vector Processor Status	VPSR	IPR144	R/W	2
Vector Arithmetic Exception	VAER	IPR145	R	2
Vector Memory Activity Check	VMAC	IPR146	R	2
Vector Translation Buffer Invalidate All	VTBIA	IPR147	W	2
Vector Indirect Register Address	VIADR	IPR157	R/W	2
Vector Indirect Data Low	VIDLO	IPR158	R/W	2
Vector Indirect Data High	VIDHI	IPR159	R/W	2

Key to Types:

R-Read  
W-Write  
R/W-Read/write

Key to Classes:

1-Implemented by the <REFERENCE>(xmp) CPU module.  
2-Implemented by the <REFERENCE>(XRV) vector module.



The IPRs listed in Table 4–8 are explicitly accessible to software only by the Move To Processor Register (MTPR) and Move From Processor Register (MFPR) instructions, which require kernel mode privileges. (The vector indirect registers are also accessed with MTPR and MFPR instructions. These registers are described in the *System Technical User's Guide*.)

From the console, EXAMINE/I and DEPOSIT/I commands read and write the IPRs. EXAMINE/M and DEPOSIT/M commands provide access to the vector indirect registers above hex address 400. EXAMINE/VE and DEPOSIT/VE provide access to the vector data registers.

Other instructions, the Move To/From Vector Processor (MTVP/MFVP) instructions, are used by software to access the Vector Length, Vector Count, and Vector Mask control registers. From the console, these registers are specified as VLR, VCR, and VMR after DEPOSIT and EXAMINE commands, with no qualifiers.

For more information on accessing the vector module registers, see the *VAX 6000 Series Vector Processor Owner's Manual*.

## Chapter 5

# MS65A Memory

---

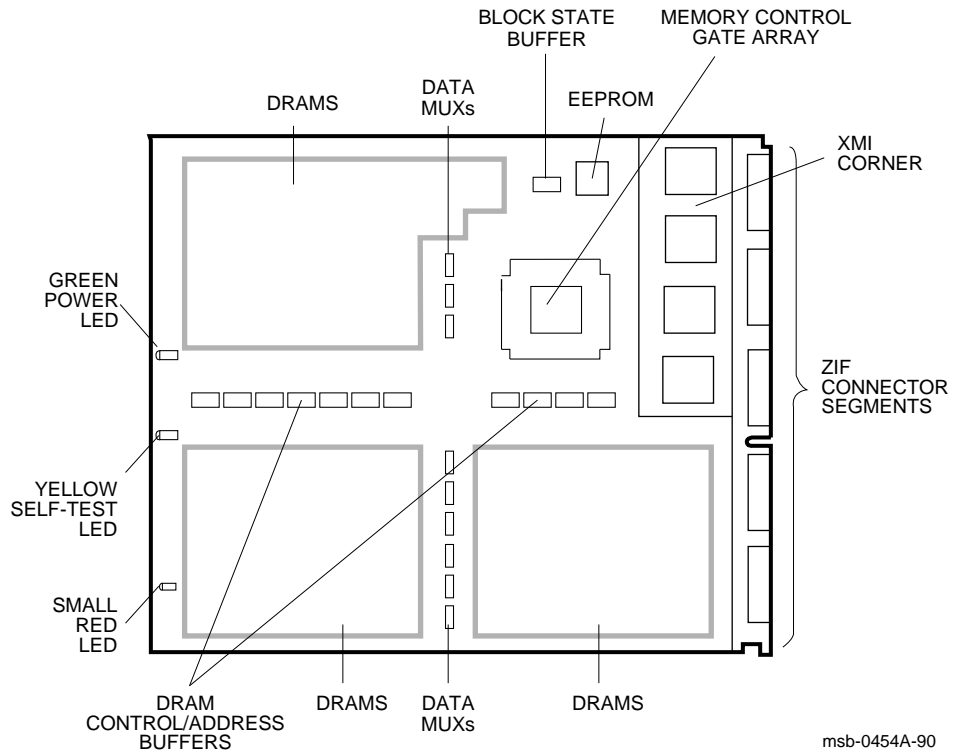
This chapter discusses the <REFERENCE>(XMA) memory module. Sections include:

- Physical Description
- Configuration Rules
- Specifications
- Functional Description
- Interleaving
- Console Commands for Interleaving
- Addressing
- Memory Self-Test
- Memory Self-Test Errors
- Control and Status Registers

## 5.1 MS65A Physical Description

The <REFERENCE>(XMA) memory module is a metal-oxide semiconductor (MOS), dynamic random access memory (DRAM), which provides up to 128 Mbytes of data storage. The memory module is designed for use with the VAX 6000 through the XMI bus.

Figure 5-1: MS65A Module



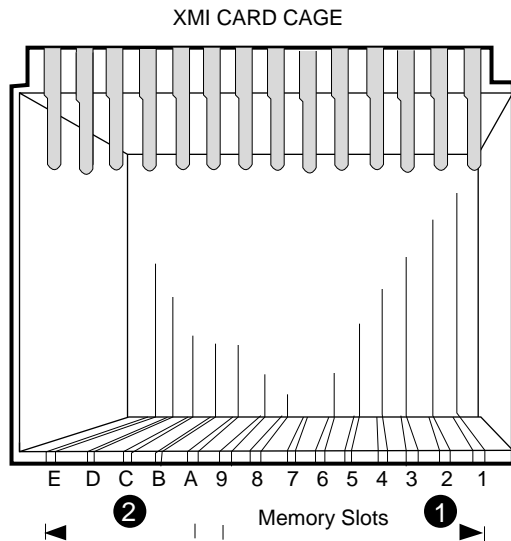
The <REFERENCE>(XMA) memory module has the following features:

- The memory module contains MOS dynamic RAM (DRAM) arrays; a CMOS memory control gate array that contains error correction code (ECC) logic and control logic; an EEPROM storage element; and an XMI interface known as the XMI Corner.
- Storage arrays are made up of two or four banks, either 155 or 299 DRAMs.
- ECC logic detects single-bit and double-bit errors and corrects single-bit errors on 64-bit words.
- Memory self-test checks all RAMs, the data path, and control logic on power-up.
- Quadwords, octawords, and hexwords can be read from or written to memory.
- Memory is configured by the console program for 2-, 4-, 8-way or no interleaving.

## 5.2 MS65A Configuration Rules

**Figure 5-2 shows the order of placement of MS65A modules in the XMI backplane.**

**Figure 5-2: MS65A Configuration**



msb-0133D-90

Memory modules are configured after I/O adapter and processor modules. Install memory modules next to vector processors first, then install additional memories as follows:

- 1 Install the first memory module in slot 9. Fill all available slots left to right from slot 9 to slot 1.
- 2 Install any additional memory modules right to left in available slots from slot A to slot E.

## 5.3 MS65A Specifications

**Table 5-1 gives the <REFERENCE>(XMA) memory module specifications.**

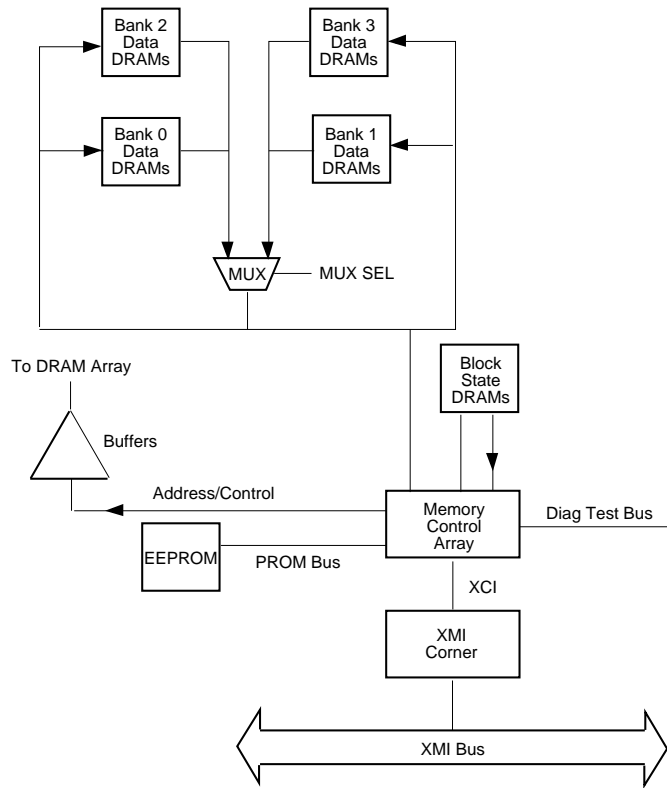
**Table 5-1: <REFERENCE>(XMA\_TITLE) Specifications**

<b>Parameter</b>	<b>Description</b>
<b>Module Number:</b>	T2053
<b>Dimensions:</b>	23.3 cm (9.2") H x 0.23 cm (0.093") W x 28.0 cm (11.0") D
<b>Memory Size:</b>	MS65A-BA 32 Mbytes MS65A-CA 64 Mbytes MS65A-DA 128 Mbytes
<b>Addresses:</b>	16-Mbyte boundaries
Starting Address	0 to 512 Gbytes
Ending Address	0 to 512 Gbytes
<b>Technology:</b>	
DRAMS	1 or 4 Mbit dynamic RAMs
Gate Arrays	CMOS gate array
<b>Interleave:</b>	2-, 4-, 8-way or none
<b>Error Correction Code:</b>	Detects single- and double-bit errors and corrects single-bit errors
<b>Temperature:</b>	
Storage Range	-40°C to 66°C (-40°F to 151°F)
Operating Range	5°C to 50°C (41°F to 122°F)
<b>Relative Humidity:</b>	
Storage and Operating	10 to 95% noncondensing
<b>Altitude:</b>	
Storage	Up to 4.8 km (16,000 ft)
Operating	Up to 2.4 km (8000 ft)
<b>Current:</b>	10A active, 3.8A standby, max.
<b>Power:</b>	50W active, 19W standby, max.

## 5.4 MS65A Functional Description

The <REFERENCE>(XMA) module consists of an XMI Corner, a memory control gate array, address and control drivers, block state DRAMs, DRAM arrays, and an EEPROM.

Figure 5-3: MS65A Block Diagram



msb-0730-90

The XMI Corner is located on the <REFERENCE>(XMA) module and contains interface logic.

The memory control gate array transfers data between the XMI Corner and the DRAMs. The memory control gate array also controls address multiplexing, command decoding, arbitration, and CSR logic functions.

Address and control logic modifies address bits received from the XMI Corner. These modified address bits are used to control the selection of the DRAMs during reading and writing.

Memory is arranged in two or four banks of DRAMs. Each bank contains either 155 or 299 DRAMs on each memory module.

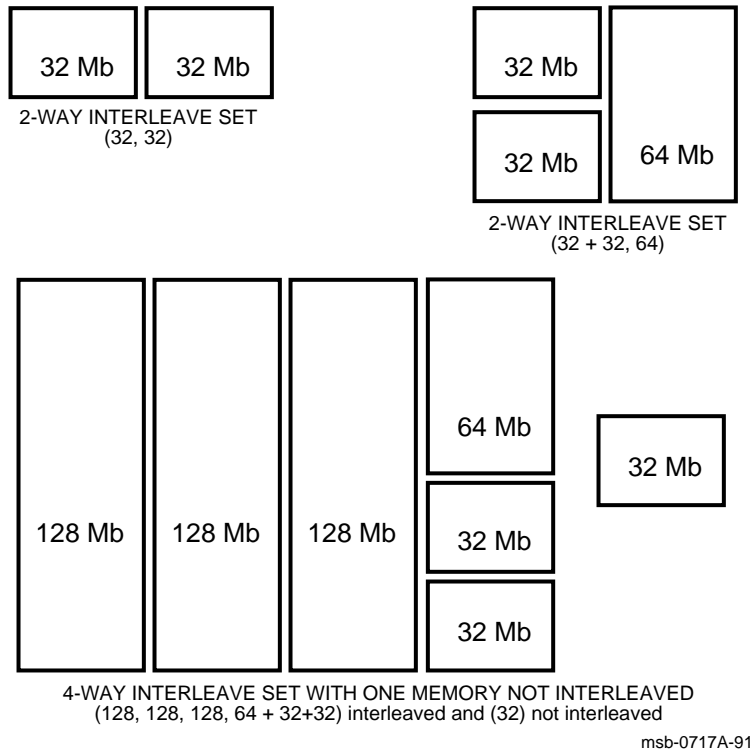
The data in the EEPROM is used to initialize the memory control gate array. After a power-up or system reset, the data in the EEPROM is loaded into the memory control gate array.



## 5.5 MS65A Interleaving

**Interleaving optimizes memory access time and increases the effective memory transfer rate by operating memory modules in parallel.**

**Figure 5-4: MS65A Interleaving**



Memory supports 2-, 4-, 8-way or no interleaving. Up to eight memory modules of the same size can be interleaved. Memory modules of different sizes can also be interleaved. Figure 5-4 shows three examples of interleaving. The first is a two-way set (32, 32); two arrays of the same size are interleaved. The second two-way set (32 + 32, 64) consists of different size arrays. The interleave set at the bottom of Figure 5-4 is a four-way set consisting of several array sizes.

Interleaving is done on hexword boundaries. Interleaving addresses are set in the Starting and Ending Address Register by the console program (see Section 5.7). The <REFERENCE>(XMA) does not check for valid or invalid interleaving configurations.

**NOTE:** *Memory modules that fail self-test due to multiple bit errors are not included in the interleave set.*

When different sizes of memory modules are installed in a Model 500 system, the console interleaves the memory modules according to size and sets as follows.

- Sorts memory modules into groups by size.
- Interleaves the largest size memory modules first.
- Stacks remaining sets of modules into sets that equal the largest size memory modules and interleaves them with the largest size memory modules.
- Stacks remaining modules into sets of the next largest size memory modules and interleaves them.
- Continues stacking and interleaving memory modules until all memory modules have been configured (including noninterleaved modules).

Unless the system requires a specific, dedicated memory use, you should run the default interleave rather than setting interleaving manually. In default, the console program chooses the optimal configuration for the system. Manual interleaving requires more operator attention.

## 5.6 Console Commands for Interleaving

The **SET MEMORY** and **SHOW MEMORY** commands are useful for setting the interleave to a memory configuration other than the default interleave. This is not usually advisable, but occasional customer use will warrant overriding the original console setting of the interleave. The **INITIALIZE** command causes the <REFERENCE>(XX) system to execute <REFERENCE>(XMA) self-tests.

### Example 5-1: SET MEMORY and INITIALIZE Commands

```
>>> SET MEMORY /INTERLEAVE:DEFAULT ❶
! For a system with one 64-Mbyte and two
! 32-Mbyte memory modules, it creates a 2-way
! interleave of 64-Mbyte memory modules
! (1x64-Mbyte and 2x32-Mbyte memory modules)
! located at XMI nodes 9, 8, and 7.

>>> SHOW MEMORY ❷ ! Displays the memory lines from self-test.
F E D C B A 9 8 7 6 5 4 3 2 1 0 NODE #
. . . . . A3 A2 A1 . . . . . ILV
. . . . . 32 32 64 . . . . . 128Mb

/INTERLEAVE:DEFAULT

>>> SET MEMORY /INTERLEAVE:(7, 8+9) ❸
! Explicitly specifies what is created
! as requested by the user (two interleave
! sets with modules in nodes 7, 8, and 9).

>>> INITIALIZE ❹ ! Initializes the system.

>>> SHOW MEMORY ❺ ! Displays the memory lines from self-test.
F E D C B A 9 8 7 6 5 4 3 2 1 0 NODE #
. . . . . B2 B1 A1 . . . . . ILV
. . . . . 32 32 64 . . . . . 128Mb

/INTERLEAVE:(7, 8+9)
>>>
```

The callouts in Example 5–1 are explained below.

- ❶ Shows the SET MEMORY command that configures interleaving with the console program. This command invokes the default interleaving configuration. It is recommended that this default be used, rather than trying to interleave memory manually.
- ❷ The SHOW MEMORY command displays the node number (node #), interleave (ILV), and total usable memory (xxMb) lines from the self-test results.
- ❸ Shows the SET MEMORY command that creates a 2-way interleave as requested by the user. In this example the user explicitly specified how to interleave the memory modules. Each interleaving set must contain the node number of the memory module. If there is more than one memory module in a set, they are joined by a + sign. Each set of interleaved memory modules must be separated by a comma.
- ❹ The system is initialized, self-test is run, and the >>> prompt returns. Section 5.8 describes the memory self-test and shows test results.
- ❺ The SHOW MEMORY command displays the configuration set in ❸.

**NOTE:** Refer to Chapter 5 of the VAX 6000 Series Owner's Manual for detailed information on the SET MEMORY and SHOW MEMORY commands.

The SET MEMORY command does not change memory interleaving; it just modifies the memory configuration in the EEPROM. The memory configuration specified by the SET MEMORY command takes place when the system is initialized (by a power-up or INITIALIZE command).

## 5.7 MS65A Addressing

**Memory addressing is set on hexword boundaries and depends on the interleaving sets organized by the console. Starting and ending addresses are determined by the console regardless of how interleaving is done (by the user or by the console).**

**Figure 5-5: MS65A Addressing**

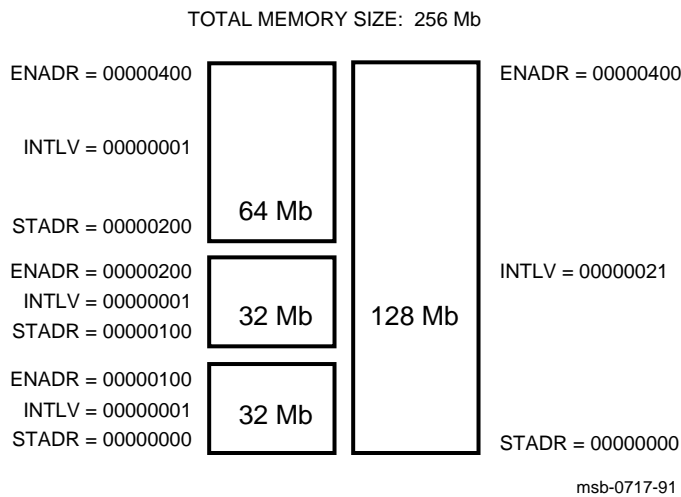


Figure 5-5 shows the starting address (STADR), ending address (ENADR), and interleave (INTLV) registers of a sample interleave set. The contents of these registers are set by the console.

The memory shown in Figure 5-5 is divided into two interleaving sets and totals 256 Mbytes. Set 0 consists of two 32-Mbyte arrays and one 64-Mbyte array. Set 1 consists of one 128-Mbyte array.

The starting address of the first array is 0. The ending address is determined by multiplying the density of the array by the interleave factor (number of sets). For example, the starting address of the first array in set 0 is 0, and the ending address is 100 hex (64 decimal, which is equal to 32 multiplied by 2). The starting address of the second array is the same as the ending address of the first.

Each array's interleave register indicates the set it belongs to (bits <7:5>) and the total number of interleave sets (bits <1:0>). The interleave register for the 128-Mbyte array indicates that the array is set 1 (bits <7:5>=001) of two interleave sets (bits <1:0>=01).

## 5.8 Memory Self-Test

The <REFERENCE>(XMA) performs an initialization and self-test sequence on power-up or when the sequence is requested by a console command. During memory self-test the array chip is initialized, all memory locations are tested, and the control and status registers are initialized.

### Example 5–2: MS65A Memory Module Results in Self-Test

```
#123456789 0123456789 0123456789 0123456789 012345#
F  E  D  C  B  A  9  8  7  6  5  4  3  2  1  0  NODE #
  A  A  .  .  .  M  M  M  M  .  .  P  P  P  TYP  ①
  +  +  .  .  .  +  +  +  +  .  .  +  +  +  STF  ②
  .  .  .  .  .  .  .  .  .  .  .  .  E  E  B  BPD
  .  .  .  .  .  .  .  .  .  .  .  +  +  +  ETF
  .  .  .  .  .  .  .  .  .  .  .  .  E  E  B  BPD
  .  .  .  .  .  A4 A3 A2 A1 .  .  .  .  ILV  ③
  .  .  .  .  .  64 64 64 64 .  .  .  .  256Mb ④

Console = V1.00  RBDs = V1.00  EEPROM = 1.00/1.00  SN = SG01234567
>>>
```

The callouts in Example 5-2 are explained below.

- ❶ The **TYP** line shows that memory modules are installed in XMI slots 6 through 9 as indicated by the M in this row.
- ❷ The **STF** line shows if memory modules pass self-test, as indicated by the + in this row. If a module fails self-test, a - is indicated, but the console still tests all pages within the module. The failing module is included in the configuration, and the addresses that fail self-test are not used by the system.
- ❸ The **ILV** line indicates the memory array modules are 4-way interleaved.
- ❹ This system contains a total usable memory of 256 Mbytes (four 64-Mbyte memory modules).

If all <REFERENCE>(XMA) nodes pass self-test, the CPU/memory test is performed on the <REFERENCE>(XMA) by the CPU. The console executes a simple read/write test to a small portion of memory. Since there are no errors from the self-test, the memory bitmap is set with all pages as good.



## 5.9 Memory Self-Test Errors

**If an <REFERENCE>(XMA) node fails self-test, an explicit memory test is run on the failing module and console error messages are displayed. The failing module is still included in the memory configuration.**

### Example 5–3: MS65A Memory Module Node Exclusion

```
>>> SET MEMORY /INTERLEAVE:(7+8, 9)
>>> INITIALIZE
      [Self-test display prints]
>>> SHOW MEMORY

F  E  D  C  B  A  9  8  7  6  5  4  3  2  1  0  NODE #
.  .  .  .  .  .  B1 A2 A1 -  .  .  .  .  .  ILV
.  .  .  .  .  .  64 64 64 .  .  .  .  .  192Mb
/INTERLEAVE:(7+8, 9)
```

If an <REFERENCE>(XMA) node fails self-test, then the console executes an explicit memory test during the building of the bitmap. Failing memory modules are included in the configuration, although they are interleaved by themselves. The only way to exclude a memory module from interleaving is to use the SET MEMORY command without designating the node you want to exclude. Example 5–3 shows how to exclude the memory module at node 6.

During the explicit memory test, any number of the following console messages might be displayed to aid the customer service engineer in diagnosing the problem.

```
?0037 Explicit interleave list is bad. Configuring
      all arrays uninterleaved.
```

This means that the explicit set of memory arrays for the explicit interleave includes no nodes that contain memory array. All memory arrays found in the system are unconfigured (the SET MEMORY command may have specified nodes that did not contain memory modules).

```
?0046 Memory interleave set is inconsistent: n n ...
```

This means that the listed nodes (*nn*) do not form a valid memory interleave set. One or more of the nodes might not be a memory array or the set contains an invalid number of memory arrays. Each listed memory array

that is valid will be configured uninterleaved; any memory array that is not included in the set will not be interleaved.

?0047 Insufficient working memory for normal operation.

This means that less than 256 Kbytes per processor of working memory were found. There may be insufficient memory for the console to function or for the operating system to boot.

?011E Uncorrectable memory errors discovered -- long memory test must be performed on node *n*

This means that a memory array contains an unrecoverable error. The console must perform a slow test to locate all the failing locations.

?004A Memories not interleaved due to uncorrectable errors.

This means that the listed arrays would normally have been interleaved (by default or an explicit request). Because one or more arrays contained unrecoverable errors, this interleave set will not be constructed.

**NOTE:** *Refer to Appendix A for a list of console error messages. See also Section 6.6 in the VAX 6000 Series Owner's Manual for more information on these errors.*

When self-test has finished running on the module, the yellow LED (located at the center of the module's edge farthest from the XMI backplane) lights. After self-test, starting and ending addresses are set by the boot processor.

## 5.10 MS65A Control and Status Registers

**The memory contains 19 control and status registers (CSRs) to control the memory and log errors. All CSRs are 32 bits long and respond only to longword read and write transactions. Only full writes are performed to the CSRs. If a parity error occurs during a write operation, the operation is aborted and the contents of the CSRs are unchanged.**

The CSRs start at an address dependent upon the node ID. All CSR addresses are designated as BB + *n*, where *n* is the relative offset of the register.

**Table 5–2: MS65A Control and Status Registers**

<b>Register</b>	<b>Mnemonic</b>	<b>Address</b>
Device Register	XDEV	BB <sup>1</sup> + 00
Bus Error Register	XBER	BB + 04
Starting and Ending Address Register	SEADR	BB + 10
Memory Control Register 1	MCTL1	BB + 14
Memory ECC Error Register	MECER	BB + 18
Memory ECC Address Register	MECEA	BB + 1C
Memory Control Register 2	MCTL2	BB + 30
TCY Tester Register	TCY	BB + 34
Block State ECC Error Register	BECER	BB + 38
Block State ECC Address Register	BECEA	BB + 3C
Starting Address Register	STADR	BB + 50
Ending Address Register	ENADR	BB + 54
Segment/Interleave Control Register	INTLV	BB + 58
Memory Control Register 3	MCTL3	BB + 5C
Memory Control Register 4	MCTL4	BB + 60
Block State Control Register	BSCTL	BB + 68

<sup>1</sup>"BB" refers to the base address of an <REFERENCE>(XMI) node (2180 0000 + (node ID x 8000))

**Table 5–2 (Cont.): MS65A Control and Status Registers**

<b>Register</b>	<b>Mnemonic</b>	<b>Address</b>
Block State Address Register	BSADR	BB + 6C
EEPROM Control Register	EECTL	BB + 70
Timeout Control/Status Register	TMOER	BB + 74

## Chapter 6

# DWMBB I/O Adapter

---

This chapter discusses the DWMBB adapter, the interface to an optional VAXBI I/O channel. Sections include:

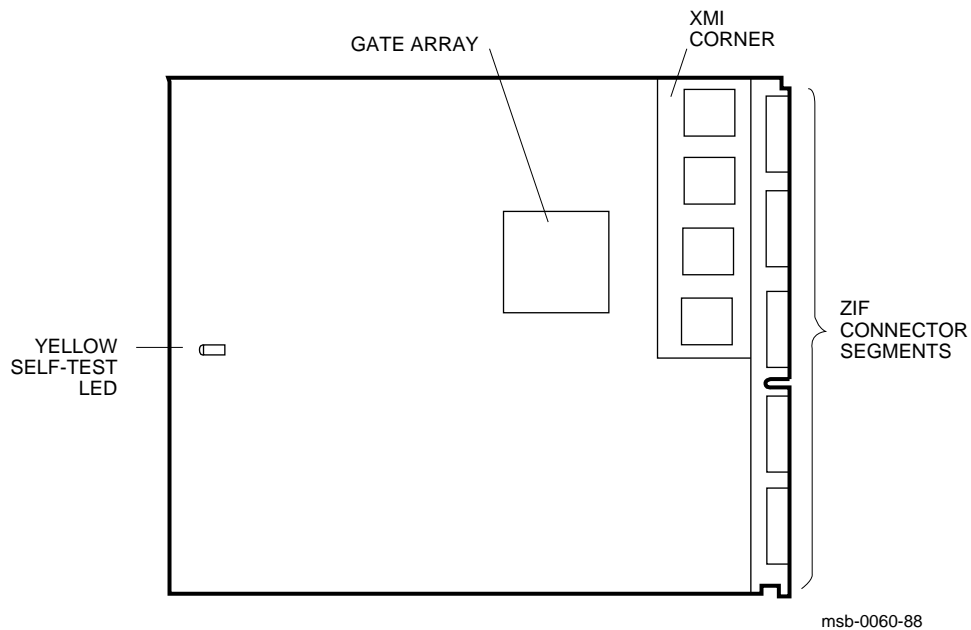
- Physical Description
  - Physical Layout
  - Specifications
- Configuration Rules
- Functional Description
- Registers

## 6.1 DWMBB Physical Description

### 6.1.1 Physical Layout

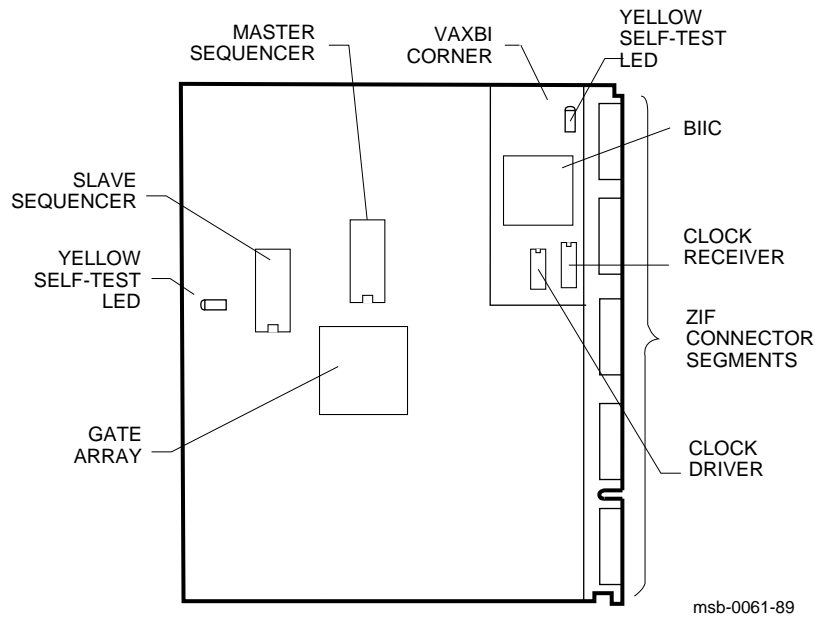
The <REFERENCE>(xbia) is an XMI module (T2018) with the standard XMI Corner, an XMI self-test OK LED indicator, IBUS drivers/receivers and transceivers, timeout logic, and a gate array that controls the <REFERENCE>(xbia). Most of the components on the <REFERENCE>(xbia) are surface-mounted.

Figure 6-1: <REFERENCE>(XBIA\_TITLE)



The <REFERENCE>(xbib) is a standard VAXBI (T1043) module with a VAXBI Corner, including a BIIC interface chip, the primary interface between the VAXBI bus and the <REFERENCE>(xbib) node logic, a clock driver, and a clock receiver. The <REFERENCE>(xbib) gate array is used mostly for data path logic. The VAXBI self-test OK LED is on the VAXBI Corner, and the module self-test OK LED is at the module edge opposite the connector edge.

Figure 6-2: <REFERENCE>(XBIB\_TITLE)



## 6.1.2 Specifications

The following specifications apply to the DWMBB modules.

**Table 6–1: DWMBB/A Specifications**

<b>Parameter</b>	<b>Description</b>
<b>Module Number:</b>	T2018
<b>Dimensions:</b>	23.3 cm (9.2") H x 0.23 cm (0.093") W x 28.0 cm (11.0") D
<b>Temperature:</b>	
Storage Range	-40°C to 66°C (-40°F to 151°F)
Operating Range	5°C to 50°C (41°F to 122°F)
<b>Relative Humidity:</b>	
Storage and operating	10% to 95% noncondensing
<b>Altitude:</b>	
Storage	Up to 4.8 km (16,000 ft)
Operating	Up to 2.4 km (8000 ft)
<b>Current:</b>	6A at +5V
<b>Power:</b>	16W



**Table 6–2: DWMBB/B Specifications**

<b>Parameter</b>	<b>Description</b>
<b>Module Number:</b>	T1043
<b>Dimensions:</b>	20.3 cm (8") H x 0.23 cm (0.093") W x 23.3 cm (9.2") D
<b>Temperature:</b>	
Storage Range	-40°C to 66°C (-40°F to 151°F)
Operating Range	5°C to 50°C (41°F to 122°F)
<b>Relative Humidity:</b>	
Storage and operating	10% to 95% noncondensing
<b>Altitude:</b>	
Storage	Up to 4.8 km (16,000 ft)
Operating	Up to 2.4 km (8000 ft)
<b>Current:</b>	6A at +5V 10mA at -12V
<b>Power:</b>	30W

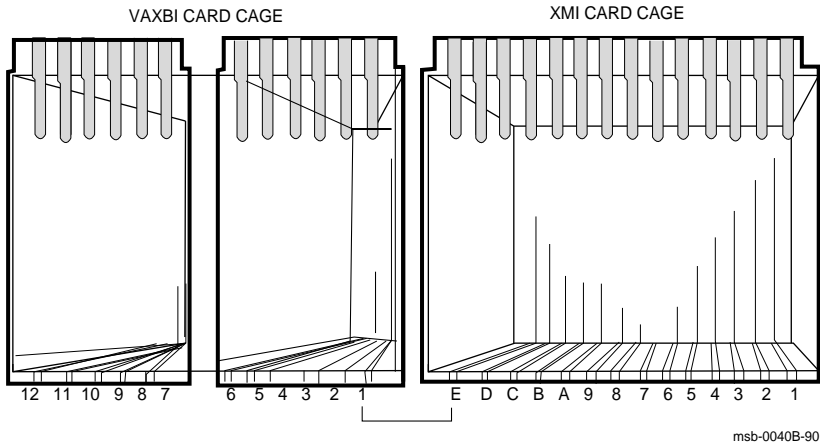
**Table 6–3: <REFERENCE>(xbi) Cables**

<b>Part Number</b>	<b>Description</b>
17-01569-01	DWMBB to H7206-B power OK cable
17-01897-01	15' DWMBB cables for expander cabinet, from XMI slots 1, 2, 3, and 4 as needed (segments D and E) to VAXBI cages 2, 3, 4, and 5 (segments D and E). Two per DWMBB.
17-01897-02	7" DWMBB cables, from XMI slot E (segments D and E) to VAXBI cage 1 slot 1 (segments D and E). Two per DWMBB.

## 6.2 <REFERENCE>(xbi) Configuration Rules

**This section describes the configuration rules for the DWMBB/A module in the XMI card cage and for the DWMBB/B module in the VAXBI card cage.**

Figure 6-3: <REFERENCE>(VAX\_XXXX) Slot Numbers



<REFERENCE>(XBIA) modules are placed in the order shown in Table 6-4.

**Table 6-4: <REFERENCE>(xbi) Configuration**

<b>&lt;REFERENCE&gt;(XMI) Node No.</b>	<b>VAXBI Channel</b>	<b>Location</b>
E	1	System cabinet
1	2	Expander cabinet
2	3	Expander cabinet
3	4	Expander cabinet
4	5	Expander cabinet

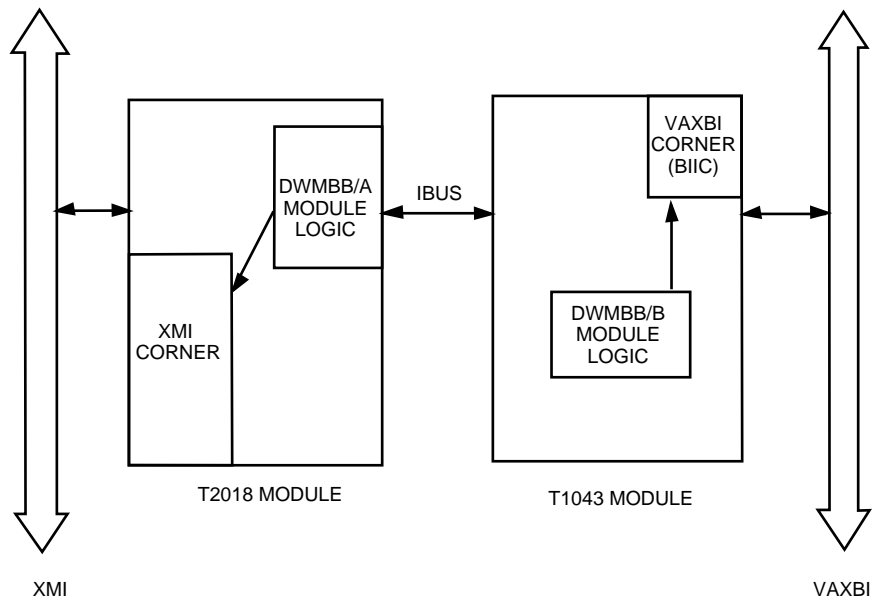
Configuration rules are as follows:

- The first VAXBI channel is the 12-slot channel in the system cabinet. The DWMBB/A module is placed in XMI slot E; the corresponding DWMBB/B module is placed in the system VAXBI cage, slot 1 (the rightmost slot). See Figure 6-3.
- Any additional VAXBI channels are 6-slot channels in the expander cabinet. The DWMBB/B module is placed in slot 1 of each. The corresponding DWMBB/A module is placed in the XMI slot listed in Table 6-4.

## 6.3 DWMBB Functional Description

The <REFERENCE>(XBI) adapter provides an information path between the <REFERENCE>(XMI) bus and I/O devices on the VAXBI bus. The <REFERENCE>(XBI) consists of two modules: the <REFERENCE>(XBIA) and the <REFERENCE>(XBIB). The <REFERENCE>(XBIA) resides on the <REFERENCE>(XMI) bus, and the <REFERENCE>(XBIB) resides on the VAXBI bus. Four 30-pin cables, which make up the IBUS, connect the two modules.

Figure 6-4: <REFERENCE>(XBI\_TITLE) Block Diagram



msb-0062A-90

The <REFERENCE>(XBIA) contains the <REFERENCE>(XMI) Corner, the register files, <REFERENCE>(XMI) required registers, <REFERENCE>(XBIA)-specific registers, page map registers, and the control sequencers for the <REFERENCE>(XMI) interface.

The <REFERENCE>(XBIB) contains the BIIC interface chip, interconnect drivers, control sequencers to handle the control of the data transfer, status bits to and from the <REFERENCE>(XBIA) module's register files and the BIIC, <REFERENCE>(XBIB)-specific registers, decode logic for direct memory access (DMA) operation, and VAXBI clock-generation circuitry.

The <REFERENCE>(XBIA) and <REFERENCE>(XBIB) modules are connected by four cables of 30 wires each. These 120 wires make up the IBUS, which transfers data and control information between the two modules.

The <REFERENCE>(XBI) uses I/O and DMA transactions to exchange information. I/O transactions originate from the <REFERENCE>(XMP) module(s) and are presented to the <REFERENCE>(XBI) from the <REFERENCE>(XMI) bus with the processor as the <REFERENCE>(XMI) commander and the <REFERENCE>(XBI) as the <REFERENCE>(XMI) responder.

DMA transactions originate from VAXBI nodes that select the <REFERENCE>(XBI) as the VAXBI slave. These are read or write transactions targeted to <REFERENCE>(XMI) memory space or are VAXBI-generated interrupt transactions that target a <REFERENCE>(XMP) module. For DMA transactions, the <REFERENCE>(XBI) is the <REFERENCE>(XMI) commander, and the <REFERENCE>(XMA) module is the <REFERENCE>(XMI) responder.

The <REFERENCE>(XBI) can be both a master and a slave on the VAXBI. As a master, it carries out transactions requested by its <REFERENCE>(XMI) devices. As a slave, it responds to VAXBI transactions that select its node.

## 6.4 <REFERENCE>(xbi) Registers

Two sets of registers are used by the <REFERENCE>(XBI) adapter: VAXBI registers (residing in the BIIC) and <REFERENCE>(XBI) registers (residing on both modules of the <REFERENCE>(XBI)). The <REFERENCE>(XBI) registers include the <REFERENCE>(XMI) required registers and <REFERENCE>(XBI)-specific registers addressed in <REFERENCE>(XBI) private space.

**Table 6–5: VAXBI Registers**

Name	Mnemonic	Address <sup>1</sup>
Device Register	DTYPE	bb+00
VAXBI Control and Status Register	VAXBICSR	bb+04
Bus Error Register	BER	bb+08
Error Interrupt Control Register	EINTRSCR	bb+0C
Interrupt Destination Register	INTRDES	bb+10
IPINTR Mask Register	IPINTRMSK	bb+14
Force-Bit IPINTR/STOP Destination Register	FIPSDDES	bb+18
IPINTR Source Register	IPINTRSRC	bb+1C
Starting Address Register	SADR	bb+20
Ending Address Register	EADR	bb+24
BCI Control and Status Register	BCICSR	bb+28
Write Status Register	WSTAT	bb+2C
Force-Bit IPINTR/STOP Command Register	FIPSCMD	bb+30
User Interface Interrupt Control Register	UINTRCSR	bb+40
General Purpose Register 0	GPR0	bb+F0
General Purpose Register 1	GPR1	bb+F4
General Purpose Register 2	GPR2	bb+F8
General Purpose Register 3	GPR3	bb+FC

<sup>1</sup>The abbreviation "bb" refers to the base address of a VAXBI node (the address of the first location of nodespace).

Table 6–5 lists the VAXBI registers. The VAXBI registers are described in Chapter 5 of the *VAXBI Options Handbook*. Table 6–6 lists the <REFERENCE>(XBI) registers.

**Table 6–6: <REFERENCE>(xbi) XMI Registers**

<b>Name</b>	<b>Mnemonic<sup>1</sup></b>	<b>Address<sup>2</sup></b>
Device Register	XDEV	BB+00
Bus Error Register	XBER	BB+04
Failing Address Register	XFADR	BB+08
Responder Error Address Register	AREAR	BB+0C
Error Summary Register	AESR	BB+10
Interrupt Mask Register	AIMR	BB+14
Implied Vector Interrupt Destination/Diagnostic Register	AIVINTR	BB+18
Diagnostic 1 Register	ADG1	BB+1C
Utility Register	AUTLR	BB+20
Control and Status Register	ACSR	BB+24
Return Vector Register	ARVR	BB+28
XMI Failing Address Extension Register	XFAER	BB+2C
VAXBI Error Address Register	ABEAR	BB+30
Control and Status Register	BCSR	BB+40
Error Summary Register	BESR	BB+44
Interrupt Destination Register	BIDR	BB+48
Timeout Address Register	BTIM	BB+4C
Vector Offset Register	BVOR	BB+50
Vector Register	BVR	BB+54
Diagnostic Control Register 1	BDCR1	BB+58
Reserved Register	BRSVD	BB+5C

<sup>1</sup>If the first letter of the mnemonic is "X" or "A," it indicates that the register resides on the <REFERENCE>(XBIA) module; a first letter of "B" indicates that the register resides on the <REFERENCE>(XBIB) module.

<sup>2</sup>The abbreviation "BB" refers to the base address of an <REFERENCE>(XMI) node (the address of the first location of nodespace).

**Table 6–6 (Cont.): <REFERENCE>(xbi) XMI Registers**

<b>Name</b>	<b>Mnemonic<sup>1</sup></b>	<b>Address<sup>2</sup></b>
Page Map Register (first location)	PMR	BB+200
.		
.		
Page Map Register (last location)	PMR	BB+401FC



## Appendix A

# Console Error Messages

---

Table A-1 lists the console error messages that appear when the processor halts and the console gains control. Most messages are followed by:

- PC = xxxxxxxx — program counter = address at which the processor halted or the exception occurred
- PSL = xxxxxxxx — processor status longword = contents of the register
- -SP = xxxxxxxx — -SP is one of the following:
  - ESP executive stack pointer
  - ISP interrupt stack pointer
  - KSP kernel stack pointer
  - SSP supervisor stack pointer
  - USP user stack pointer

Table A-2 lists standard console error messages for the Model 500.

**Table A-1: Console Error Messages Indicating Halt**

<b>Error Message</b>	<b>Meaning</b>
?0002 External halt (CTRL/P, break, or external halt).	<code>[CTRL/P]</code> or STOP command.
?0003 Power-up halt.	System has powered up, had a system reset, or an XMI node reset.
?0004 Interrupt stack not valid during exception processing.	Interrupt stack pointer contained an invalid address.
?0005 Machine check occurred during exception processing.	A machine check occurred while handling another error condition.
?0006 Halt instruction executed in kernel mode.	The CPU executed a Halt instruction.

**Table A-1 (Cont.): Console Error Messages Indicating Halt**

<b>Error Message</b>	<b>Meaning</b>
?0007 SCB vector bits <1:0> = 11.	An interrupt or exception vector in the System Control Block contained an invalid address.
?0008 SCB vector bits <1:0> = 10.	An interrupt or exception vector in the System Control Block contained an invalid address.
?000A CHMx executed while on interrupt stack.	A change-mode instruction was issued while executing on the interrupt stack.
?0010 ACV/TNV occurred during machine check processing.	An access violation or translation-not-valid error occurred while handling another error condition.
?0011 ACV/TNV occurred during kernel-stack-not-valid processing.	An access violation or translation-not-valid error occurred while handling another error condition.
?0012 Machine check occurred during machine check processing.	A machine check occurred while processing a machine check.
?0013 Machine check occurred during kernel-stack-not-valid processing.	A machine check occurred while handling another error condition.
?0019 PSL <26:24>= 101 during interrupt or exception.	An exception or interrupt occurred while on the interrupt stack but not in kernel mode.
?001A PSL <26:24>= 110 during interrupt or exception.	An exception or interrupt occurred while on the interrupt stack but not in kernel mode.
?001B PSL <26:24>= 111 during interrupt or exception.	An exception or interrupt occurred while on the interrupt stack but not in kernel mode.
?001D PSL <26:24> = 101 during REI.	An REI instruction attempted to restore a PSL with an invalid combination of access mode and interrupt stack bits.
?001E PSL <26:24> = 110 during REI.	An REI instruction attempted to restore a PSL with an invalid combination of access mode and interrupt stack bits.
?001F PSL <26:24> = 111 during REI.	An REI instruction attempted to restore a PSL with an invalid combination of access mode and interrupt stack bits.

**Table A–2: Standard Console Error Messages**

<b>Error Message</b>	<b>Meaning</b>
?0020 Illegal memory reference.	An attempt was made to reference a virtual address (V) that is either unmapped or is protected against access under the current PSL.
?0021 Illegal command.	The command was not recognized, contained the wrong number of parameters, or contained unrecognized or inappropriate qualifiers.
?0022 Illegal address.	The specified address was recognized as being invalid, for example, a general purpose register number greater than 15.
?0023 Value is too large.	A parameter or qualifier value contained too many digits.
?0024 Conflicting qualifiers.	A command specified recognized qualifiers that are illegal in combination.
?0025 Checksum did not match.	The checksum calculated for a block of X command data did not match the checksum received.
?0026 Halted.	The processor is currently halted.
?0027 Item was not found.	The item requested in a FIND command could not be found.
?0028 Timeout while waiting for characters.	The X command failed to receive a full block of data within the timeout period.
?0029 Machine check accessing memory.	Either the specified address is not implemented by any hardware in the system, or an attempt was made to write a read-only address, for example, the address of the 33rd Mbyte of memory on a 32-Mbyte system.
?002A Unexpected machine check or interrupt.	A valid operation within the console caused a machine check or interrupt.
?002B Command is not implemented.	The command is not implemented by this console.
?002C Unexpected exception.	An attempt was made to examine either a nonexistent IPR or an unimplemented register in RSSC address range (20140000—20140800).

**Table A-2 (Cont.): Standard Console Error Messages**

<b>Error Message</b>	<b>Meaning</b>
?002D For Secondary Processor <i>n</i> .	This message is a preface to second message describing some error related to a secondary processor. This message indicates which secondary processor is involved.
?002E Specified node is not an I/O adapter.	The referenced node is incapable of performing I/O or did not pass its self-test.
?0030 Write to Z command target has timed out.	The target node of the Z command is not responding.
?0031 Z connection terminated by ^P.	A CTRL/P was typed on the keyboard to terminate a Z command.
?0032 Your node is already part of a Z connection.	You cannot issue a Z command while executing a Z command.
?0033 Z connection successfully started.	You have requested a Z connection to a valid node.
?0034 Specified target already has a Z connection.	The target node was the target of a previous Z connection that was improperly terminated. Reset the system to clear this condition.
?0036 Command too long.	The command length exceeds 80 characters.
?0037 Explicit interleave list is bad. Configuring all arrays uninterleaved.	The list of memory arrays for explicit interleave includes no nodes that are actually memory arrays. All arrays found in the system are configured.
?0039 Console patches are not usable.	The console patch area in EEPROM is corrupted or contains a patch revision that is incompatible with the console ROM.
?003B Error encountered during I/O operation.	An I/O adapter returned an error status while the console boot primitive was performing I/O.
?003C Secondary processor not in console mode.	The primary processor console needed to communicate with a secondary processor, but the secondary processor was not in console mode. STOP the node or reset the system to clear this condition.

**Table A-2 (Cont.): Standard Console Error Messages**

<b>Error Message</b>	<b>Meaning</b>
?003D Error initializing I/O device.	A console boot primitive needed to perform I/O, but could not initialize the I/O adapter.
?003E Timeout while sending message to secondary processor.	A secondary processor failed to respond to a message sent from the primary. The primary sends such messages to perform console functions on secondary processors.
?003F Microcode power-up self-test failed in REX520.	Model 400 CPU chip failed its microcoded self-test.
?0040 Key switch must be at "Update" to update EEPROM.	A SET command was issued, but the key switch was not set to allow updates to the EEPROM.
?0041 Specified node is not a bus adapter.	A command to access a VAXBI node specified an XMI node that was not a bus adapter.
?0042 Invalid terminal speed.	The SET TERMINAL command specified an unsupported baud rate.
?0043 Unable to initialize node.	The INITIALIZE command failed to reset the specified node.
?0044 Processor is not enabled to BOOT or START.	As a result of a SET CPU/NOENABLE command, the processor is disabled from leaving console mode.
?0045 Unable to stop node.	The STOP command failed to halt the specified node.
?0046 Memory interleave set is inconsistent: <i>nn...</i>	The listed nodes do not form a valid memory interleave set. One or more of the nodes might not be a memory array or might be of a different size, or the set could contain an invalid number of members. Each listed array that is a valid memory will be configured uninterleaved.
?0047 Insufficient working memory for normal operation.	Less than 256 Kbytes per processor of working memory were found. There is insufficient memory for the console to function normally or for the operating system to boot.

**Table A-2 (Cont.): Standard Console Error Messages**

<b>Error Message</b>	<b>Meaning</b>
?0048 Uncorrectable memory errors—long memory test must be performed.	A Model 400 memory array contains an unrecoverable error. The console must perform a slow test to locate all the failing locations.
?0049 Memory cannot be initialized.	The specified operation was attempted and prevented.
?004A Memories not interleaved due to uncorrectable errors:	The listed arrays would normally have been interleaved (by default or explicit request). Because one or more of them contained unrecoverable errors, this interleave set will not be constructed.
?004B Internal logic error in console.	The console encountered a theoretically impossible condition.
?004C Invalid node for Z command.	The target of a Z command must be a CPU or an I/O adapter and must not be the primary processor.
?004D Invalid node for new primary.	The SET CPU command failed when attempting to make the specified node the primary processor.
?004E Specified node is not a processor.	The specified node is not a processor, as required by the command.
?004F System serial number has not been initialized.	No CPU in the system contains a valid system serial number.
?0050 System serial number not initialized on primary processor.	The primary processor has an uninitialized system serial number. All other processors in the system contain a valid serial number.
?0051 Secondary processor returned bad response message.	A secondary processor returned an unintelligible response to a request made by the console on the primary processor.
?0052 ROM revision mismatch. Secondary processor has revision <i>x.xx</i> .	The revision of console ROM of a secondary processor does not match that of the primary.
?0053 EEPROM header is corrupted.	The EEPROM header has been corrupted. The EEPROM must be restored from the TK tape drive.

**Table A-2 (Cont.): Standard Console Error Messages**

<b>Error Message</b>	<b>Meaning</b>
?0054 EEPROM revision mismatch. Secondary processor has revision <i>x.xx/y.yy</i> .	A secondary processor has a different revision of EEPROM or has a different set of EEPROM patches installed.
?0055 Failed to locate EEPROM area.	The EEPROM did not contain a set of data required by the console. The EEPROM may be corrupted.
?0056 Console parameters on secondary processor do not match primary.	The console parameters are not the same for all processors .
?0057 EEPROM area checksum error.	A portion of the EEPROM is corrupted. It may be necessary to reload the EEPROM from the TK tape drive.
?0058 Saved boot specifications on secondary processor do not match primary.	The saved boot specifications are not the same for all processors.
?0059 Invalid unit number.	A BOOT or SET BOOT command specified a unit number that is not a valid hexadecimal number between 0 and FF.
?005A System serial number mismatch. Secondary processor has xxxxxxxx.	The indicated serial number of a secondary processor does not match that of the primary.
?005B Unknown type of boot device.	The console program does not have a boot primitive to support the specified type of device or the device could not be accessed to determine its type.
?005C No HELP is available.	The HELP command is not supported when the console language is set to International.
?005D No such boot spec found.	The specified boot specification was not found in the EEPROM.
?005E Saved boot spec table full.	The maximum number of saved boot specifications has already been stored.
?005F EEPROM header version mismatch.	Processors have different versions of EEPROMs.
?0061 EEPROM header or area has bad format.	All or part of the EEPROM contains inconsistent data and is probably corrupted. Reload the EEPROM from the TK tape.
?0062 Illegal node number.	The specified node number is invalid.

**Table A-2 (Cont.): Standard Console Error Messages**

<b>Error Message</b>	<b>Meaning</b>
?0063 Unable to locate console tape device.	The console could not locate the I/O adapter that controls the TK tape.
?0064 Operation only applies to secondary processors.	The command can only be directed at a secondary processor.
?0065 Operation not allowed from secondary processor.	A secondary processor cannot perform this operation.
?0066 Validation of EEPROM tape image failed.	The image on tape is corrupted or is not the result of a SAVE EEPROM command. The image cannot be restored.
?0067 Read of EEPROM image from tape failed.	The EEPROM image was not successfully read from tape.
?0068 Validation of local EEPROM failed.	For a PATCH EEPROM operation, the EEPROM must first contain a valid image before it can be patched. For a RESTORE EEPROM operation, the image was written back to EEPROM but could not be read back successfully.
?0069 EEPROM not changed.	The EEPROM contents were not changed.
?006A EEPROM changed successfully.	The EEPROM contents were successfully patched or restored.
?006B Error changing EEPROM.	An error occurred in writing to the EEPROM. The EEPROM contents may be corrupted.
?006C EEPROM saved to tape successfully.	The EEPROM contents were successfully written to the TK tape.
?006D EEPROM not saved to tape.	The EEPROM contents were not completely written to the TK tape.
?006E EEPROM Revision = <i>x.xx/y.yy</i> .	The EEPROM contents are at revision <i>x.xx</i> with revision <i>y.yy</i> patches.
?006F Major revision mismatch between tape image and EEPROM.	The major revision of tape and EEPROM do not match. The requested operation cannot be performed.
?0070 Tape image Revision = <i>x.xx/y.yy</i> .	The EEPROM image on the TK tape is at revision <i>x.xx</i> with revision <i>y.yy</i> patches.



**Table A–2 (Cont.): Standard Console Error Messages**

<b>Error Message</b>	<b>Meaning</b>
?0073 System serial number updated.	The EEPROM has been updated with the correct system serial number.
?0074 System serial number not updated.	The EEPROM has not been changed.
?0075 /CONSOLE_LIMIT value too small for proper operation. Value ignored.	No change has been made.
?0076 Error writing to tape. Tape may be write-locked.	Tape has not been written. Check to see if tape is write-locked.
?0077 CCA not accessible or corrupted.	Attempt to find the console communications area (CCA) failed. The console then builds a local CCA, which does not allow for interprocessor communication.
?0078 Vector module configuration error at node <i>n</i>	The console detected a vector module configuration error. Problem can be that the vector node number is not one greater than the scalar CPU or that the module to the left of a vector processor is not a memory module.
?0079 Vector synchronization error.	The console could not synchronize with the vector processor on a console entry. The Busy bit in the Vector Processor Status Register remained set after a timeout, or a vector processor error occurred.
?007A No vector module associated with CPU at specified node.	No vector module is in the slot to the left of the specified CPU, or the VIB cable either is not attached or is bad.
?007B An error occurred while accessing the vector module.	Attempt to access VCR, VLR, or VMR registers failed.
?007C I/O adapter configuration error at node <i>n</i>	The I/O adapter at node <i>n</i> is configured improperly.
?007D Vector module is disabled—check KA64A revision at XMI node <i>n</i>	The vector module is attached to a KA64A module that is not at the revision level required.
?0083 Loading system software. <sup>1</sup>	The console is attempting to load the operating system in response to a BOOT command, power-up, or restart failure.

<sup>1</sup>No numbered prefix appears with these messages in English language mode. These numbers are used for these messages in International mode.

**Table A-2 (Cont.): Standard Console Error Messages**

<b>Error Message</b>	<b>Meaning</b>
?0084 Failure. <sup>1</sup>	An operation did not complete successfully. Should be issued with another message to clarify failure.
?0085 Restarting system software. <sup>1</sup>	The console is attempting to restart the in-memory copy of the operating system following a power-up or serious error.
?00A0 Initializing system. <sup>1</sup>	The console is resetting the system in response to a BOOT command.
?00A6 Console halting after unexpected machine check or exception. <sup>1</sup>	The console executed a Halt instruction to reset the console state after processing an unexpected machine check.
?00A7 RCSR <WD> is set. Local CCA must be built. <sup>1</sup>	When the <WD> bit is set, writes to memory are disabled. The Model 400 processor must then build a CCA in local memory. Main memory cannot be written to or accessed with interlocked instructions.
?00A8 Bootstrap failed due to previous error. <sup>1</sup>	The previous attempt to bootstrap the system failed.
?00A9 Restart failed due to previous error. <sup>1</sup>	The previous attempt to restart the system failed.
?0104 Filename format error.	Period and semicolon characters are improperly used within the filename specified for a MOP boot.
?0105 Illegal character(s) in filename.	For filename specified in a MOP boot.
?0106 Filename cannot contain nested blanks or tabs.	For filename specified in a MOP boot.
?0107 Filename can be no longer than 16 characters.	For filename specified in a MOP boot.
?0111 Microcode power-up self-test failed in DC595.	CPU chip failed its microcoded self-test.
?011E Uncorrectable memory errors discovered - long memory test must be performed on node <i>n</i>	Memory array in node <i>n</i> contains an uncorrectable error. The console must perform a full test to locate all the failing locations.

<sup>1</sup>No numbered prefix appears with these messages in English language mode. These numbers are used for these messages in International mode.

**Table A-2 (Cont.): Standard Console Error Messages**

<b>Error Message</b>	<b>Meaning</b>
?0120 Unsupported memory module found, will not be configured.	One or more MS62A memory modules are installed but will not be used. Only MS65A memory modules are compatible with Model 500.
?0121 Patch command no longer implemented—use the Diagnostic utility EVUCA.	An invalid PATCH command was issued; use the EVUCA program to update the EEPROM.
Node <i>n</i> : ?xxxx	Error message ?xxxx was generated on secondary processor <i>n</i> and was passed to the primary processor to be displayed.

## Appendix B

# Boot Status and Error Messages

---

This appendix lists status and error messages for Ethernet MOP, disk, tape, and CI boots. Status messages are shown in the order they would appear after the boot command is issued. Listed after each status message are the error messages that could appear during each boot subprocess.

## B.1 Ethernet MOP Boot Status and Error Messages

1. [Start boot]
  - ?002 Enode is not an I/O adapter
  - ?0100 Specified adapter failed selftest
  - ?010B Illegal adapter specified for NI boot
2. \* Initializing adapter
  - ?0119 Failure to initialize specified adapter
3. \* Specified adapter initialized successfully
4. \* "Request Program" MOP message sent—waiting for service from remote node
  - ?0115 Aborting boot process—adapter failed attempting to execute port command
  - ?0113 No traffic was detected on the net—aborting boot procedure
  - ?011F Aborting boot process—adapter failed attempting to execute boot command (DEBNA only)
5. \* Still waiting for assistance—reissuing "Request Program" message
6. \* Remote service link established
7. \* Reading boot image from remote node
  - ?010F Failed to receive image from remote server
8. \* Passing control to transfer address

## B.2 Disk Boot Status and Error Boot Messages

1. [Start Boot]
  - ?002E Specified node is not an I/O adapter
  - ?0100 Specified adapter failed selftest
  - ?010A Illegal adapter specified for disk boot
2. \* Initializing adapter
  - ?0119 Failure to initialize specified adapter
3. \* Specified adapter initialized successfully
4. \* Connecting to boot disk
  - ?0117 Specified unit offline
  - ?0118 Specified unit offline—Unit unknown, online to another controller or port disabled via A,B switches
  - ?010E Specified unit offline — No media mounted or disabled via RUN/STOP switch setting
  - ?0116 Specified unit is inoperative
  - ?0103 Drive error detected—aborting
  - ?0102 Controller error detected—aborting
  - ?0114 Serious exception reported—aborting
5. \* Reading bootblock from disk
  - ?0117 Specified unit offline
  - ?0118 Specified unit offline—Unit unknown, online to another controller or port disabled via A,B switches
  - ?010E Specified unit offline—No media mounted or disabled via RUN/STOP switch setting
  - ?0116 Specified unit is inoperative
  - ?0103 Drive error detected—aborting
  - ?0102 Controller error detected—aborting
  - ?0114 Serious exception reported—aborting
6. \* Passing control to transfer address

## B.3 Tape Status and Error Boot Messages

1. [Start boot]
  - ?002E Specified node is not an I/O adapter
  - ?0100 Specified adapter failed selftest
  - ?010C Illegal adapter specified for tape use

2. \* Initializing adapter
  - ?0119 Failure to initialize specified adapter
3. \* Specified adapter initialized successfully
4. \* Connecting to tape
  - ?0117 Specified unit offline
  - ?0118 Specified unit offline—Unit unknown, online to another controller or port disabled via A,B switches
  - ?010E Specified unit offline—No media mounted or disabled via RUN/STOP switch setting
  - ?0116 Specified unit is inoperative
  - ?0103 Drive error detected—aborting
  - ?0102 Controller error detected—aborting
  - ?0114 Serious exception reported—aborting
  - ?0101 BVP port error—aborting
5. \* Reading bootblock from tape
  - ?0117 Specified unit offline
  - ?0118 Specified unit offline—Unit unknown, online to another controller or port disabled via A,B switches
  - ?010E Specified unit offline—No media mounted or disabled via RUN/STOP switch setting
  - ?0116 Specified unit is inoperative
  - ?0103 Drive error detected—aborting
  - ?0102 Controller error detected—aborting
  - ?0114 Serious exception reported—aborting
  - ?0101 BVP port error—aborting
6. \* Rewinding tape
  - ?0117 Specified unit offline
  - ?0118 Specified unit offline—Unit unknown, online to another controller or port disabled via A,B switches
  - ?010E Specified unit offline—No media mounted or disabled via RUN/STOP switch setting
  - ?0116 Specified unit is inoperative
  - ?0103 Drive error detected—aborting
  - ?0102 Controller error detected—aborting
  - ?0114 Serious exception reported—aborting
  - ?0101 BVP port error—aborting
7. \* Passing control to transfer address

## B.4 CI Status and Error Boot Messages

1. [Start boot]
  - ?002E Specified node is not an I/O adapter
  - ?0109 Illegal adapter specified for CI boot
2. \* Initializing adapter
  - ?0119 Failure to initialize specified adapter
3. \* Specified adapter initialized successfully
4. \* Connecting to storage controller
5. \* Previous operation failed—retrying CI boot
6. \* Port received a "no path" error—retrying the init sequence
  - ?0110 Port received a "no path" error after 6 retries—aborting the boot process
7. \* Connecting to MSCP server layer
8. \* Previous operation failed—retrying CI boot
9. \* Connecting to boot disk
  - ?0117 Specified unit offline
  - ?0118 Specified unit offline—Unit unknown, online to another controller or port disabled via A,B switches
  - ?010E Specified unit offline—No media mounted or disabled via RUN/STOP switch setting
  - ?0116 Specified unit is inoperative
  - ?0103 Drive error detected—aborting
  - ?0102 Controller error detected—aborting
  - ?0114 Serious exception reported—aborting
10. \* Connecting to shadow unit
  - ?0117 Specified unit offline
  - ?0118 Specified unit offline—Unit unknown, online to another controller or port disabled via A,B switches
  - ?010E Specified unit offline—No media mounted or disabled via RUN/STOP switch setting
  - ?0116 Specified unit is inoperative
  - ?0103 Drive error detected—aborting
  - ?0102 Controller error detected—aborting
  - ?0114 Serious exception reported—aborting
11. \* Failure to connect to shadow unit—retrying on physical unit

- 12. \* Reading bootblock from disk
  - ?0117 Specified unit offline
  - ?0118 Specified unit offline—Unit unknown, online to another controller or port disabled via A,B switches
  - ?010E Specified unit offline—No media mounted or disabled via RUN/STOP switch setting
  - ?0116 Specified unit is inoperative
  - ?0103 Drive error detected—aborting
  - ?0102 Controller error detected—aborting
  - ?0114 Serious exception reported—aborting
- 13. \* Passing control to transfer address



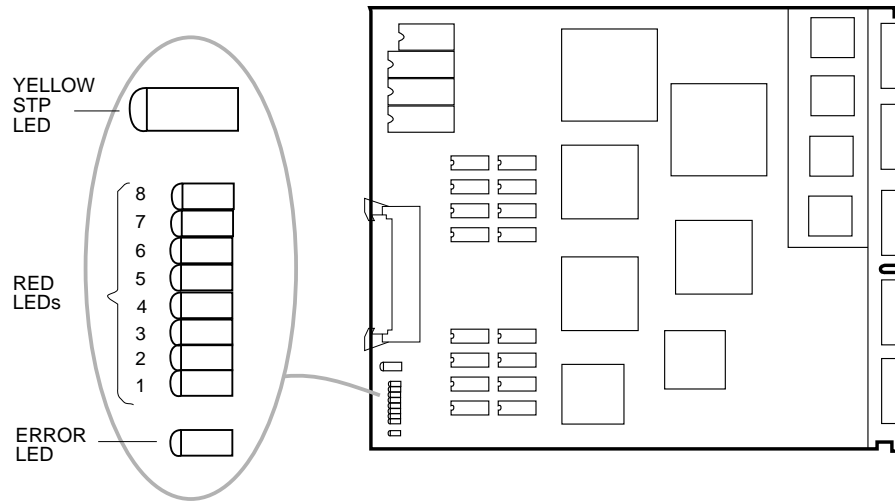
## Appendix C

# KA65A LED Patterns Indicating Console Errors

---

After self-test runs without errors, the yellow self-test passed (STP) LED remains lit, and the console lights a pattern in red LEDs 1–8. (See Figure C–1.) This pattern indicates that the console has detected either no errors or one of three console-detected errors: no primary, no memory, or no CCA. These patterns are shown in Table C–1. The state of the error LED does not affect the interpretation of these patterns.

**Figure C-1: KA65A Module LEDs**



msb-0711A-90

**Table C-1: KA65A Console LED Patterns**

<b>Red LED</b>	<b>No Errors (Primary)</b>	<b>No Errors (Secondary)</b>	<b>No Primary</b>	<b>No Memory</b>	<b>No CCA</b>
8	<b>On</b>	<b>On</b>	<b>On</b>	<b>On</b>	<b>On</b>
7	<b>On</b>	<b>On</b>	<b>On</b>	<b>On</b>	<b>On</b>
6	Off	Off	Off	Off	Off
5	Off	Off	Off	Off	Off
4	Off	Off	Off	Off	<b>On</b>
3	Off	Off	Off	<b>On</b>	Off
2	Off	Off	<b>On</b>	Off	Off
1	Off	<b>On</b>	X <sup>1</sup>	X	X

<sup>1</sup>X indicates LED can be on or off.

The LED patterns in Table C-1 are interpreted as follows:

- **No errors (primary)**—The console detected no errors. This is the primary processor.
- **No errors (secondary)**—The console detected no errors. This is a secondary processor.
- **No primary**—The console cannot locate any processor that is eligible to be the boot processor.
- **No memory**—The console cannot locate any system memory on the XMI bus.
- **No CCA**—The console cannot locate the console communications area in system memory.

## Appendix D

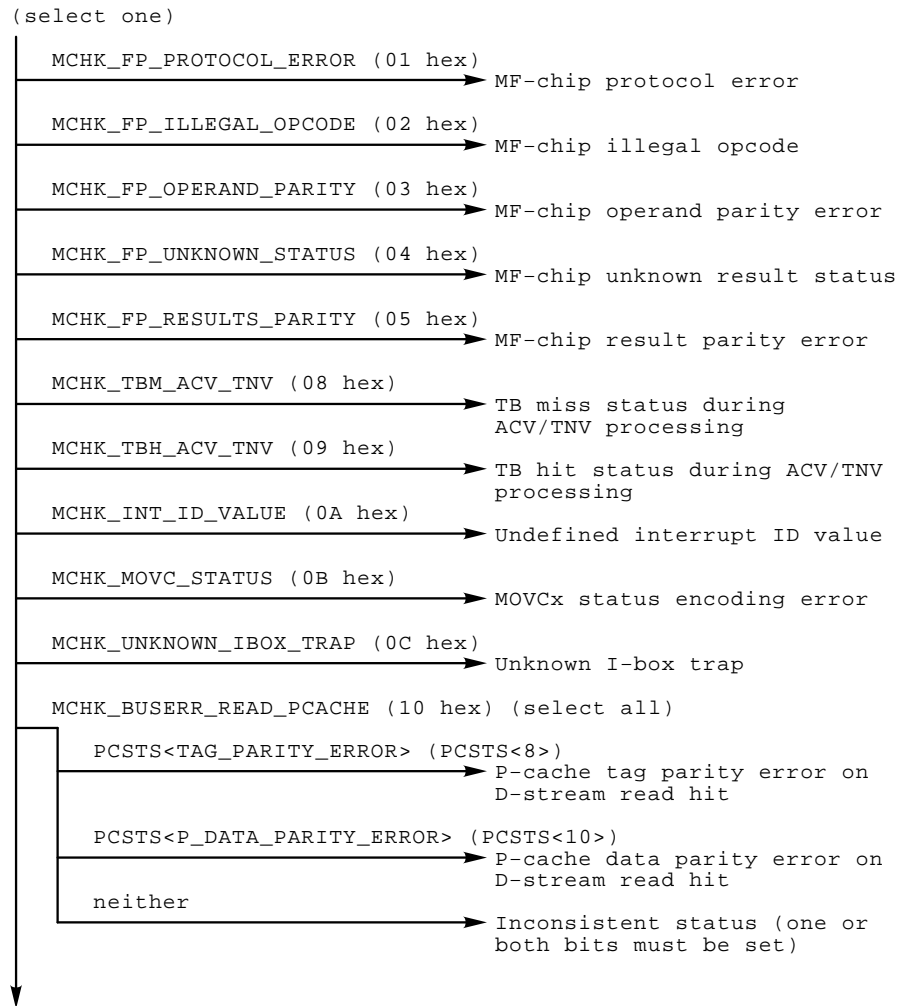
# Parse Trees

---

This appendix shows parse trees for the following:

- <REFERENCE>(XMP) Machine Checks
- <REFERENCE>(XMP) Hard Error Interrupts
- <REFERENCE>(XMP) Soft Error Interrupts
- <REFERENCE>(xrv) Machine Checks
- <REFERENCE>(xrv) Hard Error Interrupts
- <REFERENCE>(xrv) Soft Error Interrupts
- <REFERENCE>(xrv) Disable Faults

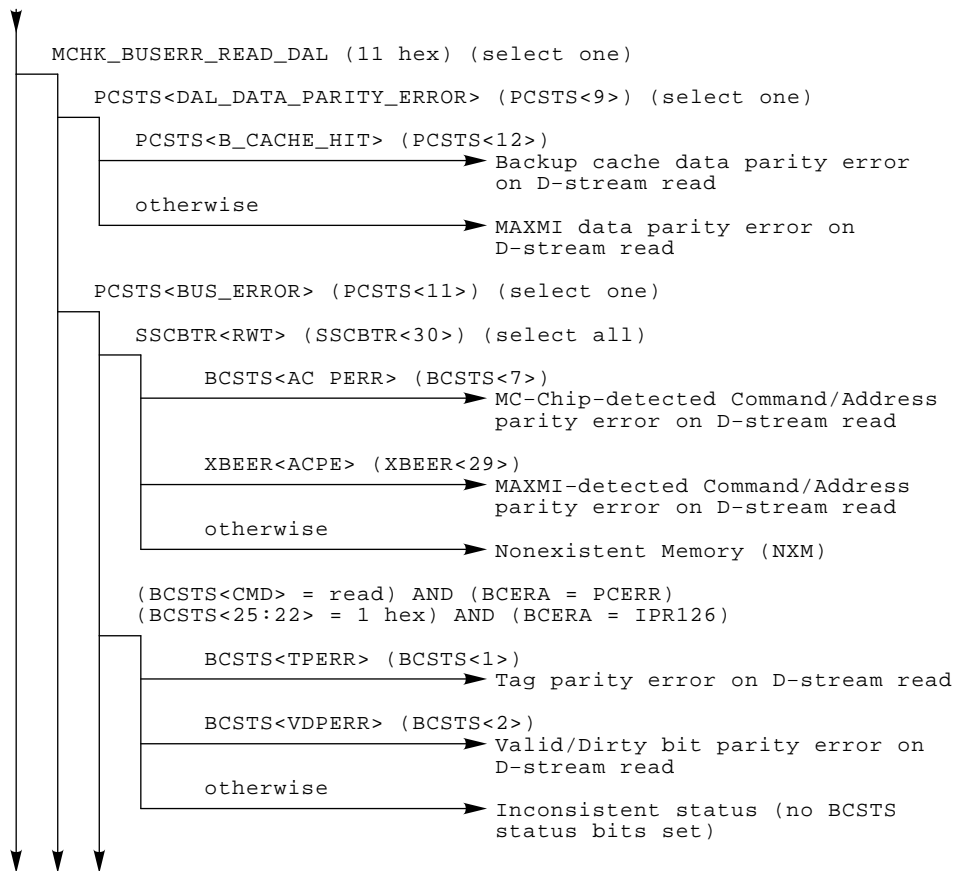
**Figure D-1: <REFERENCE>(xmp) Machine Check Parse Tree**



msb-p358-90

**Figure D-1 Cont'd on next page**

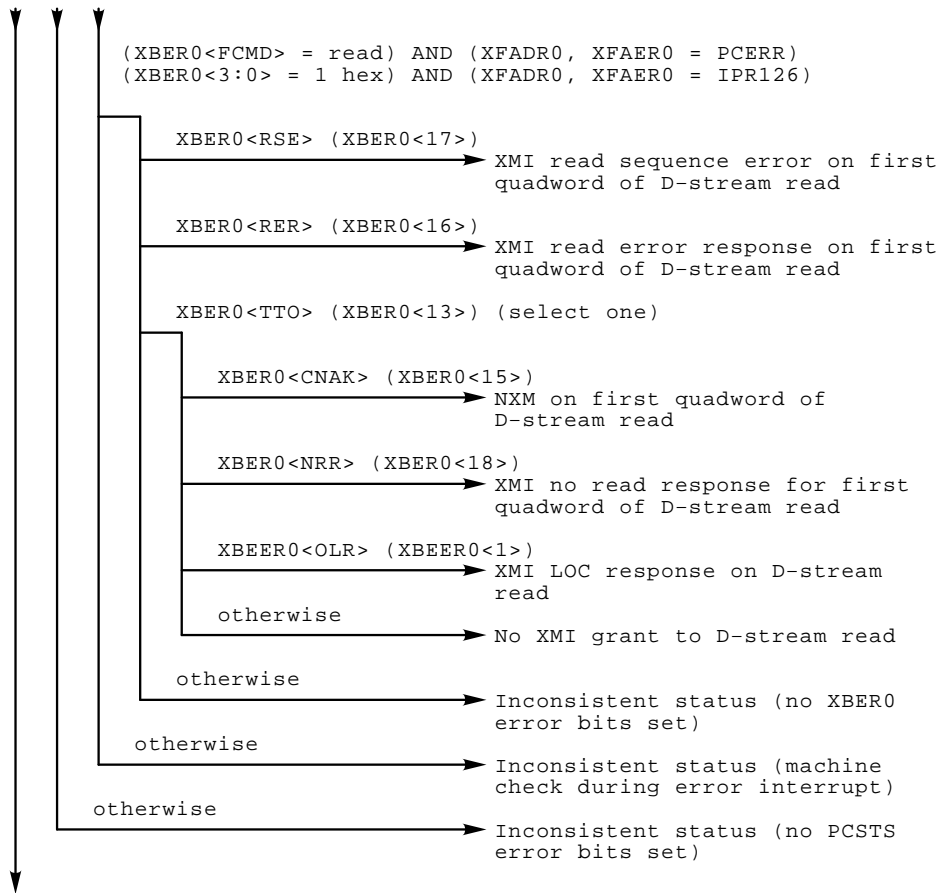
**Figure D-1 (Cont.): <REFERENCE>(xmp) Machine Check Parse Tree**



msb-p359-90

**Figure D-1 Cont'd on next page**

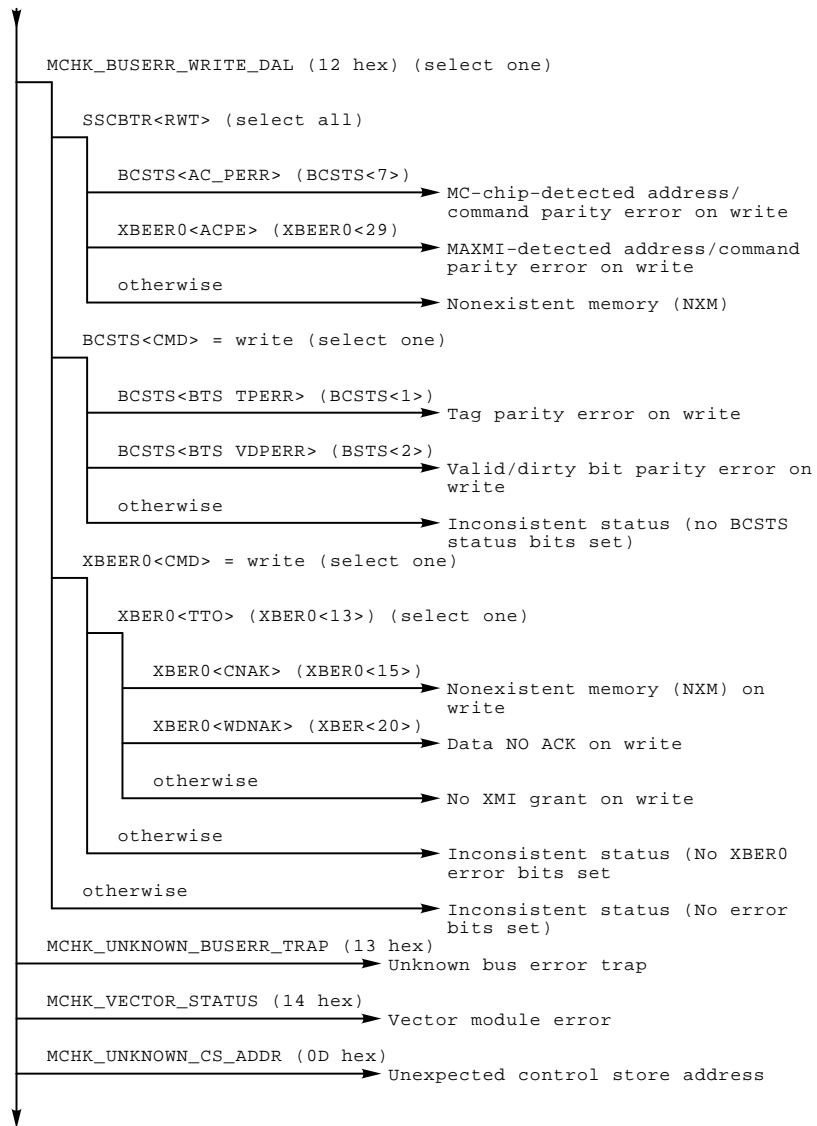
**Figure D-1 (Cont.): <REFERENCE>(xmp) Machine Check Parse Tree**



msb-p360A-90

**Figure D-1 Cont'd on next page**

**Figure D-1 (Cont.): <REFERENCE>(xmp) Machine Check Parse Tree**

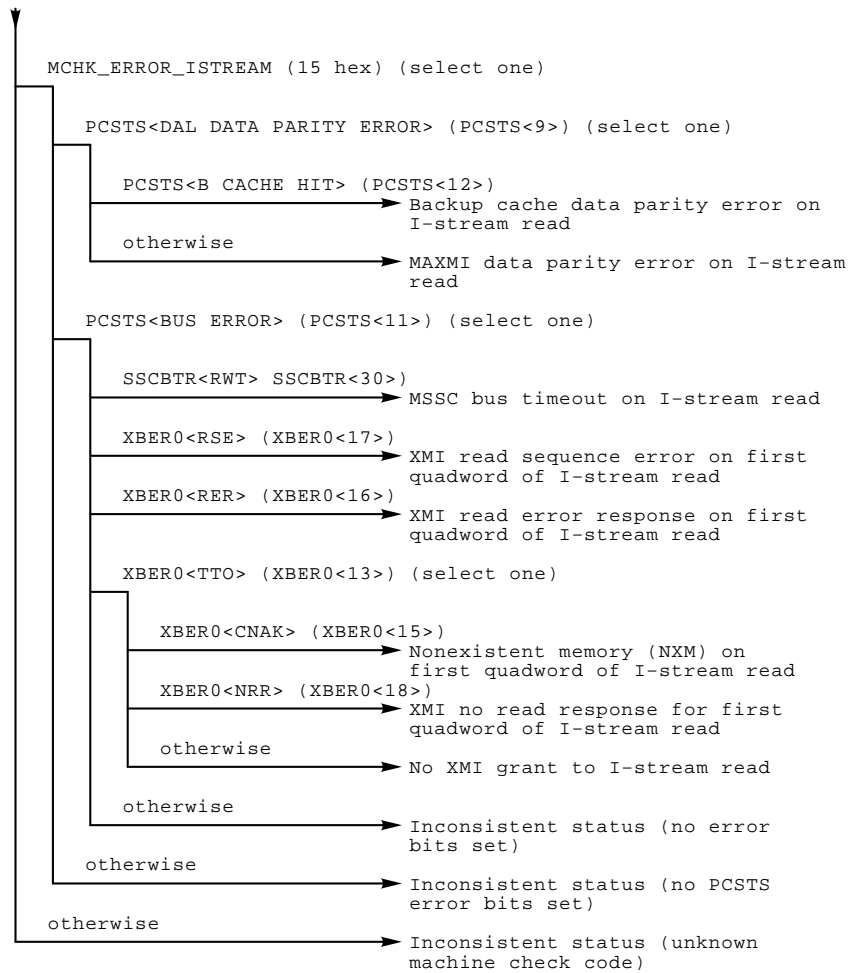


msb-p361r-90

**Figure D-1 Cont'd on next page**



**Figure D-1 (Cont.): <REFERENCE>(xmp) Machine Check Parse Tree**

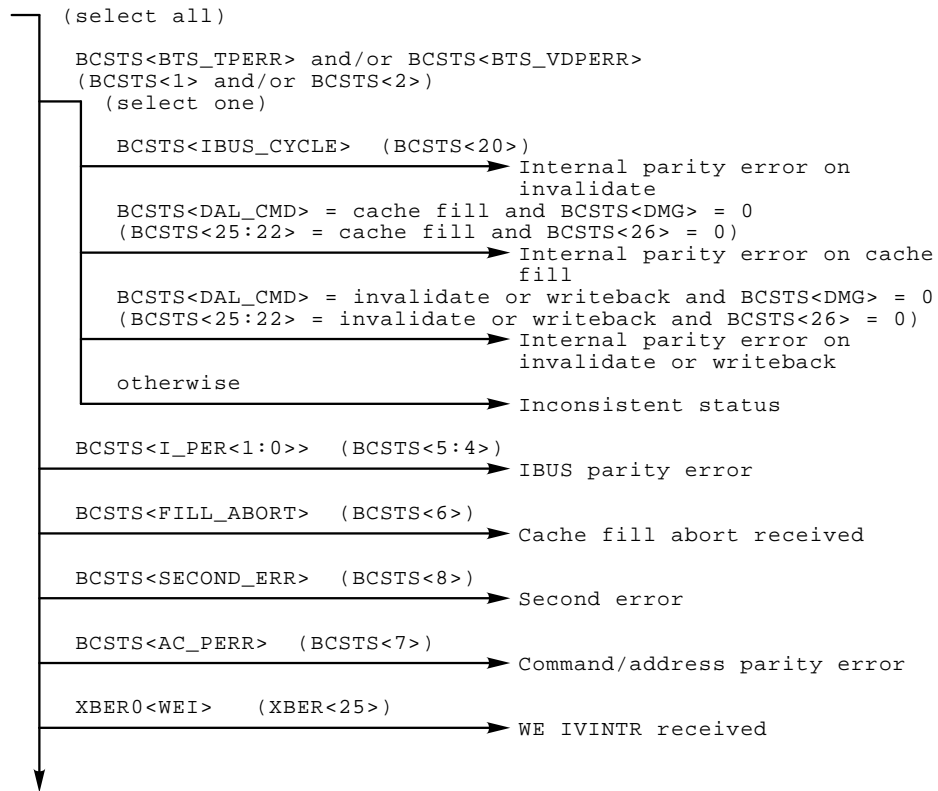


**NOTES:**

- (select one) - exactly one case must be true. If zero or more than one is true, the status is inconsistent.
- (select all) - more than one case may be true.
- otherwise - fall-through case for (select one) if no other options are true.
- neither - fall-through case for (select all) if none of the options are true.

msb-p362r-90

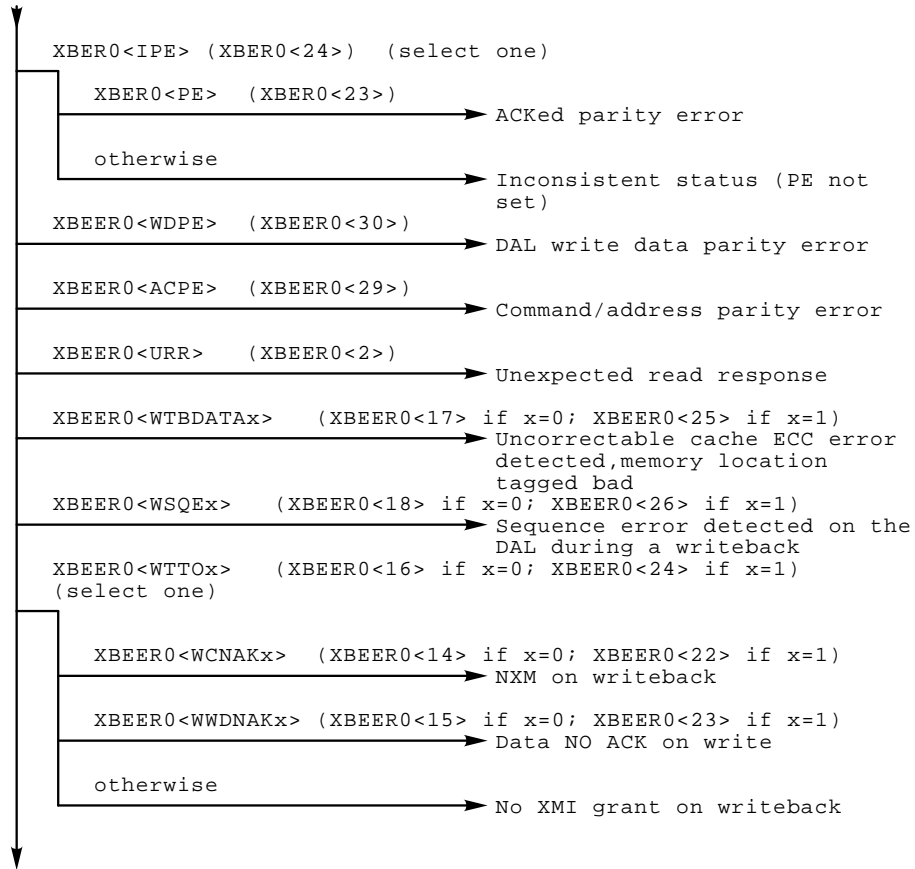
**Figure D-2: <REFERENCE>(xmp) Hard Error Interrupt Parse Tree**



msb-p363A-90

**Figure D-2 Cont'd on next page**

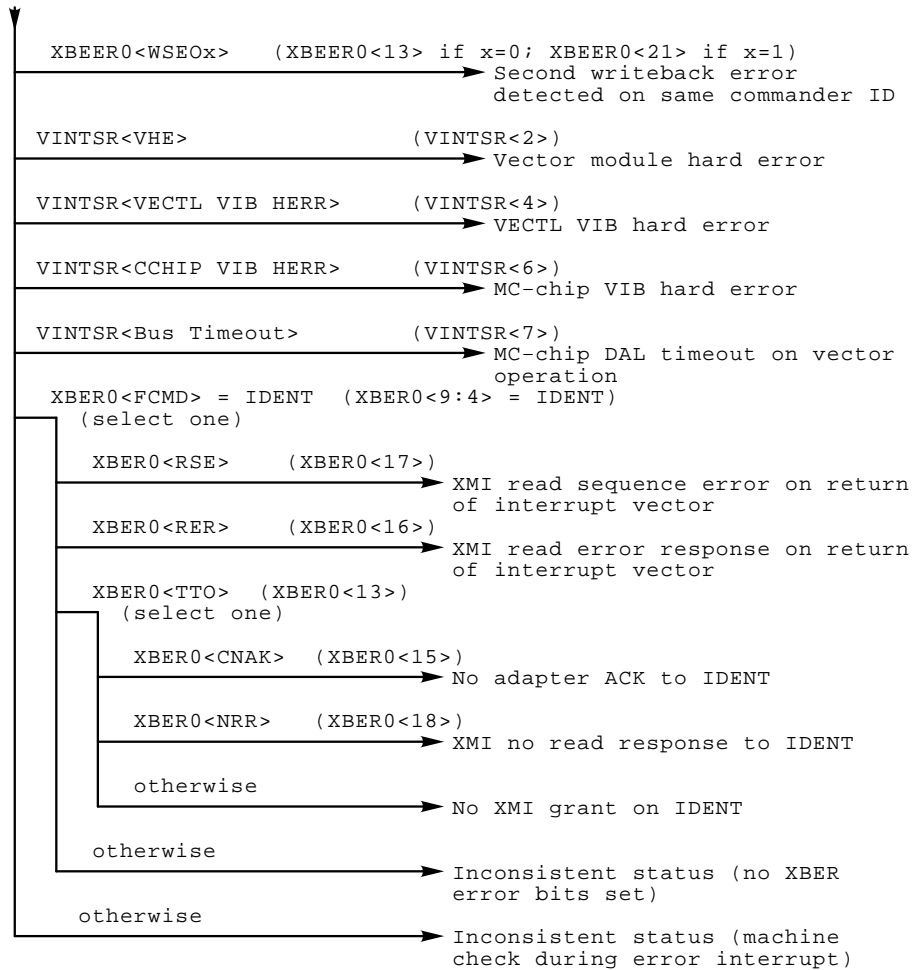
**Figure D-2 (Cont.): <REFERENCE>(xmp) Hard Error Interrupt Parse Tree**



msb-p364A-90

**Figure D-2 Cont'd on next page**

**Figure D-2 (Cont.): <REFERENCE>(xmp) Hard Error Interrupt Parse Tree**

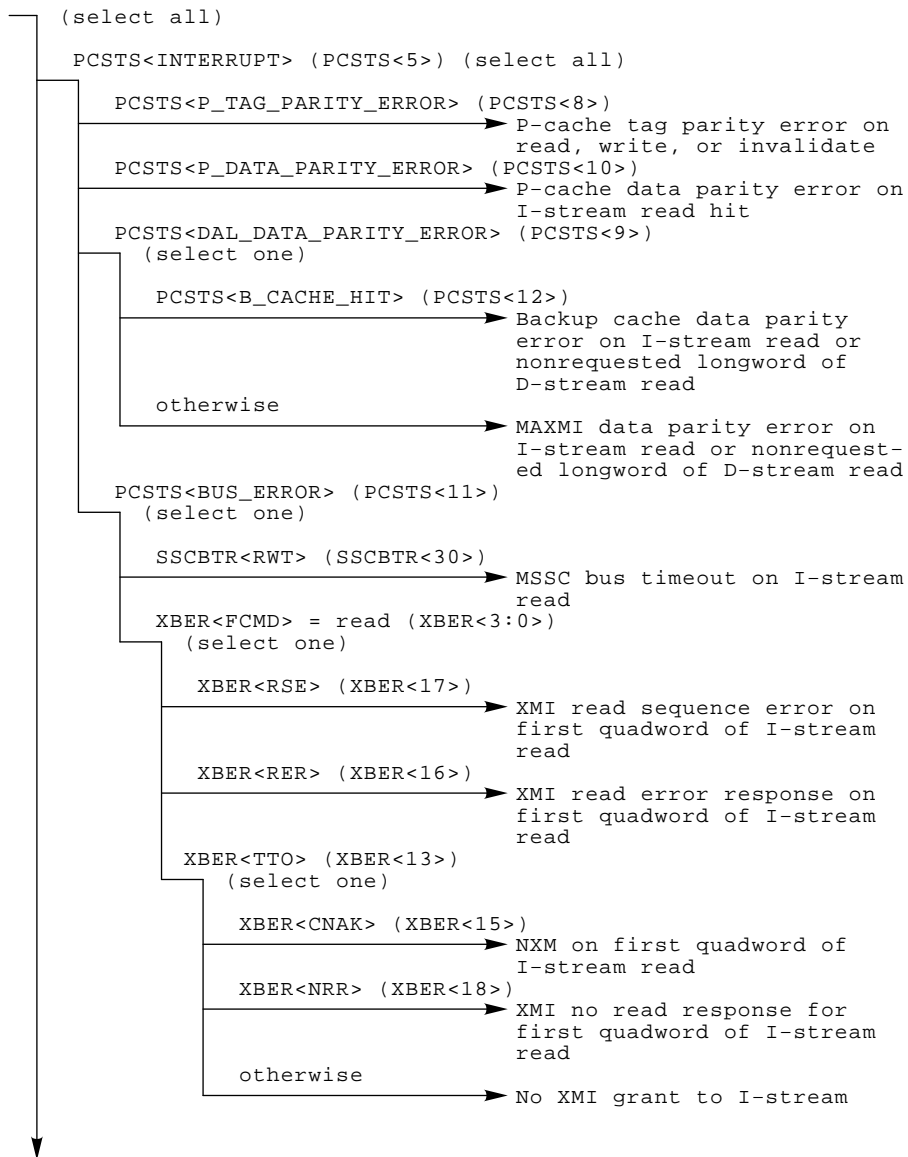


**NOTES:**

- (select one) - exactly one case must be true. If zero or more than one is true, the status is inconsistent.
- (select all) - more than one case may be true.
- otherwise - fall-through case for (select one) if no other options are true.

msb-p365A-90

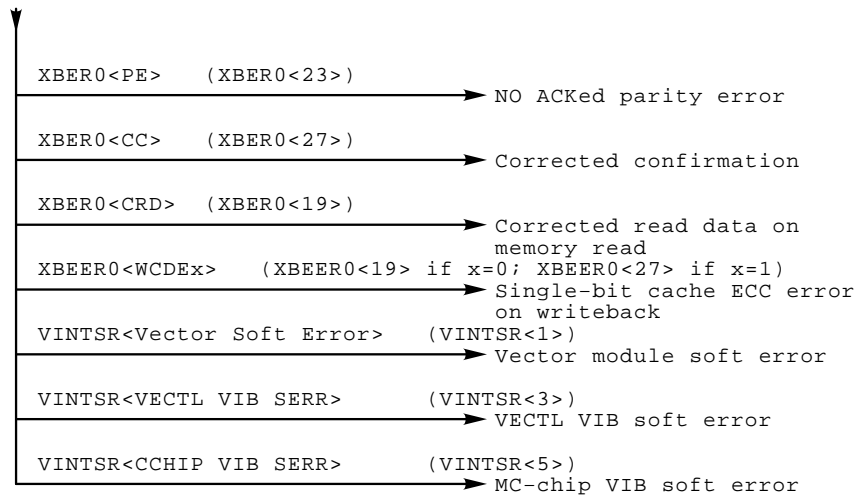
**Figure D-3: <REFERENCE>(xmp) Soft Error Interrupt Parse Tree**



msb-p366-90

**Figure D-3 Cont'd on next page**

**Figure D-3 (Cont.): <REFERENCE>(xmp) Soft Error Interrupt Parse Tree**

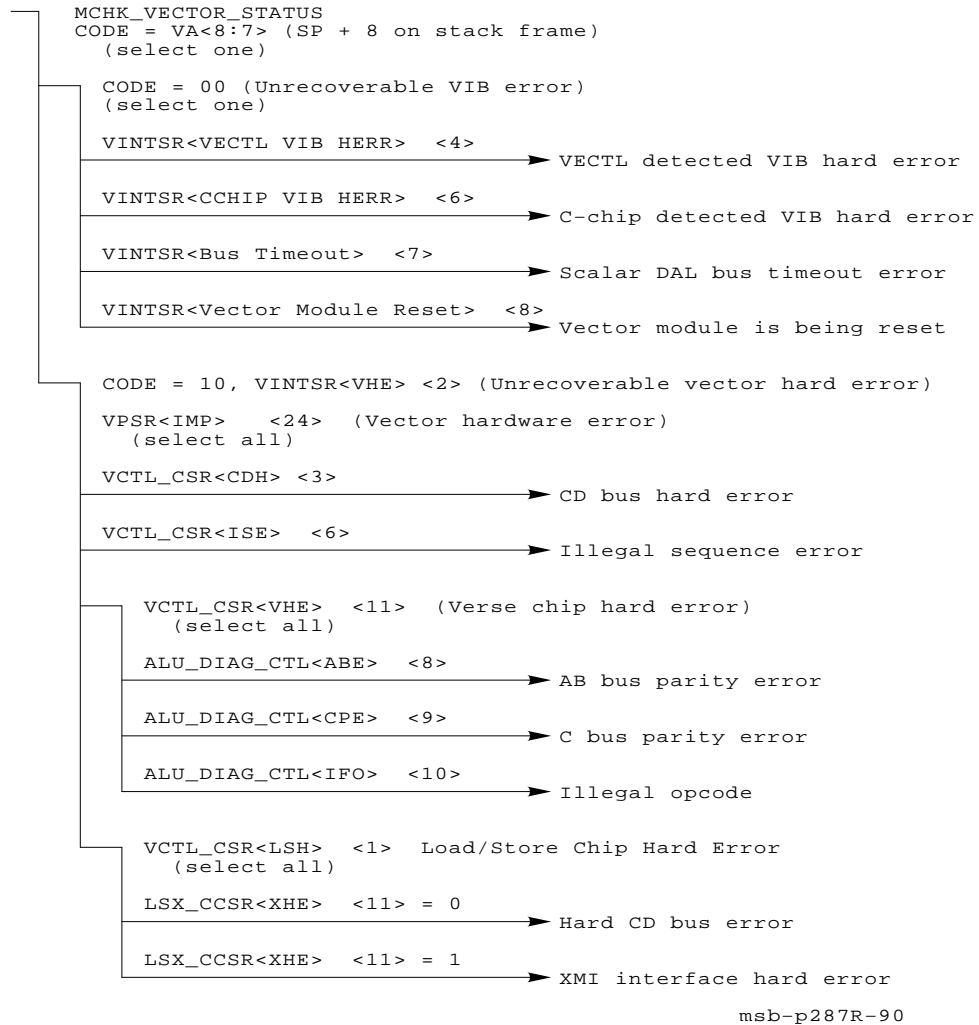


**NOTES:**

- (select one) - exactly one case must be true. If zero or more than one is true, the status is inconsistent.
- (select all) - more than one case may be true.
- otherwise - fall-through case for (select one) if no other options are true.

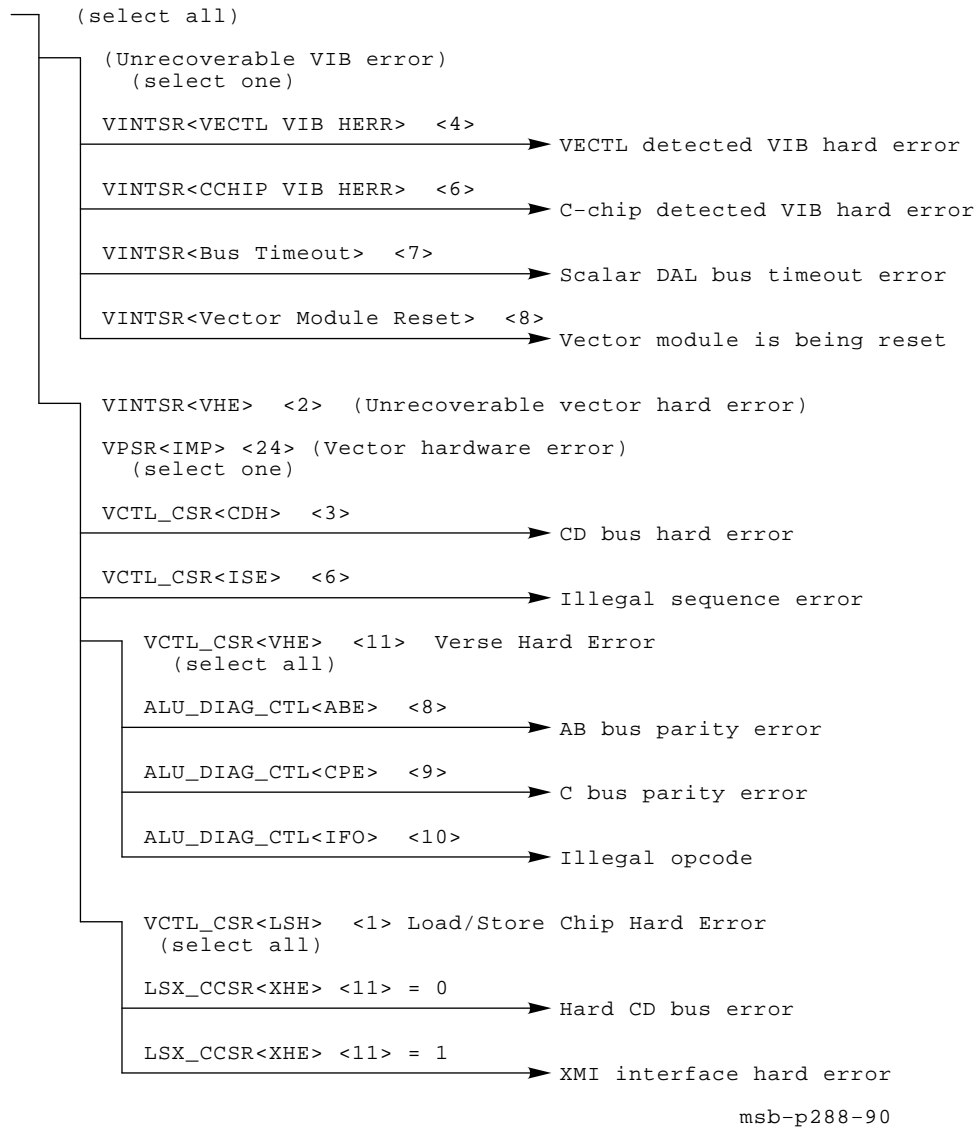
msb-p367A-90

**Figure D-4: <REFERENCE>(xrv) Machine Check Parse Tree**



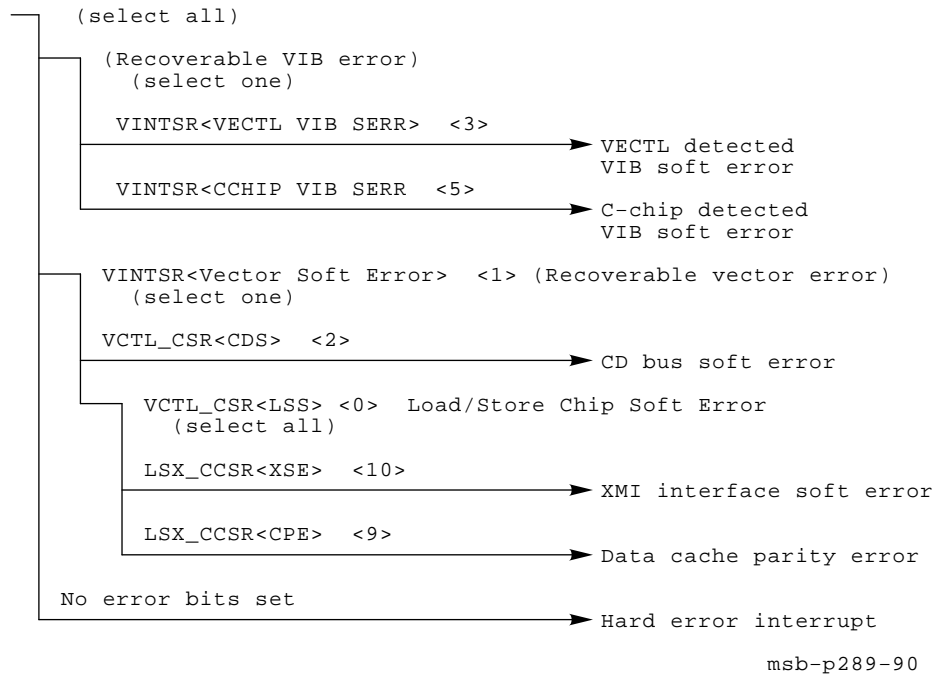
msb-p287R-90

**Figure D-5: <REFERENCE>(xrv) Hard Error Interrupt Parse Tree**

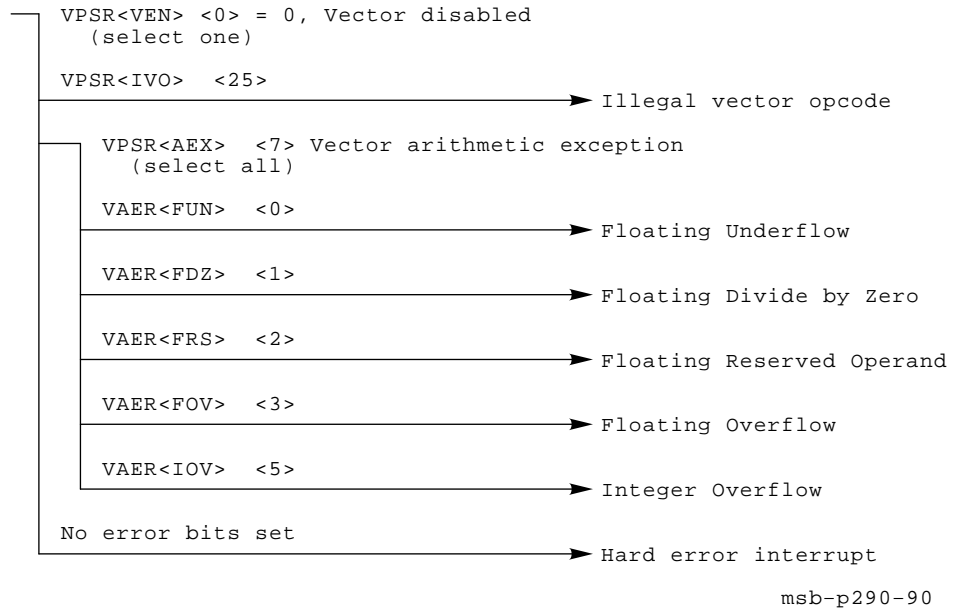




**Figure D-6: <REFERENCE>(xrv) Soft Error Interrupt Parse Tree**



**Figure D-7: <REFERENCE>(xrv) Disable Fault Parse Tree**



## Appendix E

# Restoring a Corrupted EEPROM

---

Use this procedure to restore a corrupted EEPROM. A corrupted EEPROM is indicated by any of the following console errors:

```
?0053 EEPROM header is corrupted.  
?0055 Failed to locate EEPROM area.  
?0057 EEPROM area checksum error.  
?0061 EEPROM header or area has bad format.  
?006B Error changing EEPROM.
```

**CAUTION:** *You must wear an antistatic wrist strap attached to the cabinet when you handle any modules. See Section 3.10 for processor module handling instructions.*

1. Note the module serial number and revision level of each CPU module in the system. To do this, slide each CPU module, one at a time, out of its slot so that you can see the label on the back of the module under the bar code. Return each module to its slot.
2. Turn the control panel's lower key switch to Update.
3. If the console terminal is set to a speed other than the default speed of 1200 baud, press **BREAK** until the >>> prompt prints. Alternatively, set the console terminal to 1200 baud.
4. Follow the steps shown in Examples E-1 and E-2.

## Example E-1: Restoring a Corrupted EEPROM, Part 1 of 2

```
>>> SET CPU 1 ①
>>> JSB E0040044 ②

This procedure will format the EEPROM on the primary processor,
destroying the system serial number, saved boot specifications,
terminal characteristics, console and diagnostic patches, etc.

Do you want to format the EEPROM [No]? Y ③

Zeroing EEPROM (approximately 15 seconds)

Writing data to EEPROM (approximately 15 seconds)

Move lower keyswitch from UPDATE to write-protect EEPROM ④

>>> SET CPU 2 ⑤
>>> JSB E0040044
.
.
.
>>> SET CPU 1 ⑥
>>> ESCDEL SET MANUFACTURING ⑦
Module Serial Number>>> NI000200007
Module Revision>>> D02
DC595 Revision>>>
FPU Revision>>>
SSC Revision>>>
Fields are as follows:
Module serial number:
Module revision:
DC595 revision:
FPU revision:
SSC revision:

Update EEPROM? (Y or N) >>> Y
?0071 Manufacturing parameters updated
>>> ESCDEL SET POWER ⑧
Power system>>> A
Power system read as: A

Update EEPROM? (Y or N) >>> Y
?011B Power system identification updated
>>> ESCDEL SET SYSTEM SERIAL ⑨
System Serial Number>>> AG02915081
Serial number read as: AG02915081

Update EEPROM? (Y or N) >>> Y
?0073 System serial number updated
>>> SET CPU 2 ⑩
```

- ① Make the CPU in the lowest-numbered slot the primary processor.
- ② Enter the command **JSB E0040044**. This command blasts the default EEPROM image into the current primary's EEPROM.
- ③ Type **Y** in response to the question. You do not need to press `RETURN` or `ENTER`.
- ④ Leave the key switch in the Update position until this procedure is finished.
- ⑤ Make each CPU in turn the primary processor and repeat ② and ③.
- ⑥ Again make the first CPU the primary processor.
- ⑦ The `ESC DEL` SET MANUFACTURING command prompts you for information. Enter the module serial number and module revision in response to the first two prompts. (You noted this information before starting this procedure.) Press `RETURN` in response to the rest of the prompts. Type **Y** to terminate the command.
- ⑧ The `ESC DEL` SET POWER prompts for the power system type. You can find this information in the SHOW FIELD listing that was saved for this system in the *Site Management Guide* or in another safe place. Type **Y** to terminate the command.
- ⑨ Enter the `ESC DEL` SET SYSTEM SERIAL command. The system serial number is also in the SHOW FIELD listing. Type **Y** to terminate the command.
- ⑩ Make each CPU in turn the primary processor and repeat ⑦, ⑧, and ⑨. The SET commands that are preceded by `ESC DEL` do not propagate to other processors, so they must be entered for each processor in the system.

## Example E-2: Restoring a Corrupted EEPROM, Part 2 of 2

```

>>> SET CPU 1 ⑪
>>> SET BOOT DEFAULT /XMI:E/BI:2 DU0 ⑫
>>> SET BOOT NI /XMI:C/FILENAME:ISL_LVAX EX0
>>> SET BOOT NIDI /XMI:C/FILENAME:ISL_LVAX/R5:10 EX0
>>> SET BOOT TK70 /XMI:E/BI:C MU0
>>> SET CPU /PRIMARY/ALL ⑬
>>> SET LANGUAGE INTERNATIONAL
>>> SET TERMINAL /SCOPE/SPEED:9600
>>> SET MEMORY /INTERLEAVE:DEFAULT
>>> BOOT /XMI:C/R5:110 EX0 ⑭
.
.
.
>>> INITIALIZE ⑮
#123456789 0123456789 0123456789 0123456789 012345#
F  E  D  C  B  A  9  8  7  6  5  4  3  2  1  0  NODE #
  A  .  A  .  .  M  M  .  .  .  .  P  P  P  TYP
  O  .  +  .  .  +  +  .  .  .  .  .  +  +  +  STF
  .  .  .  .  .  .  .  .  .  .  .  E  E  B  BPD
  .  .  .  .  .  .  .  .  .  .  .  +  +  +  ETF
  .  .  .  .  .  .  .  .  .  .  .  E  E  B  BPD
.  .  .  +  .  .  .  .  +  .  +  .  .  +  .  XBI E +
.  .  .  .  .  A2  A1  .  .  .  .  .  .  .  ILV
.  .  .  .  .  32  32  .  .  .  .  .  .  .  64 Mb

Console = V2.00 RBDs = V2.00 EEPROM = 2.00/2.00 SN = AG02915081

```

- ⑪ Make the first CPU the primary processor.

**NOTE:** *If the console load device is a TK70 and an image of the EEPROM has been saved on tape, you can use the RESTORE command in place of the next two steps.*

- ⑫ Use the SET BOOT command to set the boot specifications entered in the *Site Management Guide*.
- ⑬ Enter the rest of the information saved in the *Site Management Guide*.
- ⑭ Use the EVUCA utility to update the other processors' EEPROMs. See Section 3.14.
- ⑮ Initialize the system and verify there are no messages regarding console patches, corrupt EEPROMs, or system number mismatches. If the console prints any of these messages, verify that you installed the latest revision of patches. If they are the latest revision, follow the troubleshooting flowchart in Figure 1-2.

# Glossary

---

**Adapter**

A node that interfaces other buses, communication lines, or peripheral devices to the <REFERENCE>(XMI) bus or the VAXBI bus.

**Address space**

The 1 terabyte of physical address space that the XMI bus is capable of supporting; currently the XMI bus supports 1 gigabyte of physical memory.

**Asymmetric multiprocessing**

A multiprocessing configuration in which the processors are not equal in their ability to execute operating system code. In general, a single processor is designated as the primary, or master, processor; other processors are the slaves. The slave processors are limited to performing certain tasks, whereas the master processor can perform all system tasks. Contrast with *Symmetric multiprocessing*.

**Bandwidth**

The data transfer rate measured in information units transferred per unit of time (for example, Mbytes per second).

**Boot device**

Contains the bootblock and typically also contains the virtual memory boot program (VMB). A <REFERENCE>(SSS) system can be booted from one of four boot devices: the console load device, a local system disk, a disk connected to the system through a CI adapter, or a disk connected to the system through the Ethernet.

**Boot primitives**

Small programs stored in ROM on each processor with the console program. Boot primitives read the bootblock from boot devices. There is a boot primitive for each type of boot device.

**Boot processor**

The CPU module that boots the operating system and communicates with the console.



**Bootblock**

Block zero on the system disk; it contains the block number where the virtual memory boot (VMB) program is located on the system disk and contains a program that, with the boot primitive, reads VMB from the system load device into memory.

**CIBCA**

VAXBI CI port interface; connects a system to a Star Coupler.

**CIXCD**

XMI CI port interface; connects a system to a Star Coupler.

**Cold start**

An attempt by the primary processor to boot a new copy of the operating system.

**Compact disk server**

Ethernet-based CD server; provides access to CDRoms for software installation, diagnostics, and on-line documentation.

**Console communications area (CCA)**

Segment of system main memory reserved by the console program.

**Console mode**

A mode of operation allowing a console terminal operator to communicate with nodes on the XMI bus.

**DEBNI**

VAXBI adapter; Ethernet port interface.

**DEMNA**

XMI adapter; Ethernet port interface.

**DHB32**

VAXBI adapter communication device; supports up to 16 terminals.

**DMB32**

VAXBI adapter interface for 8-channel asynchronous communications for terminals, one synchronous channel, and a parallel port for a line printer.

**DRB32**

VAXBI adapter; parallel port.

**Glossary-2**

**DSB32**

VAXBI adapter communication device; provides two synchronous lines.

**DWMBB**

The XMI-to-VAXBI adapter; a 2-module adapter that allows data transfer from the XMI to the VAXBI; DWMBB/A is the module in the XMI card cage, and DWMBB/B is the VAXBI module. Every VAXBI on a <REFERENCE>(SSS) system must have a <REFERENCE>(XBI) adapter.

**Ethernet-based compact disk server**

The RRD40 compact disk drive, a console load device, functions as a server on the Ethernet.

**FV64A**

Vector processor; works in a scalar/vector processor pair.

**Interleaving memory**

See Memory interleaving.

**KDB50**

VAXBI adapter for RAxx disks; enables connection to disk drives.

**KDM70**

XMI adapter for RAxx disks; enables connection to disk drives.

**Memory interleaving**

Method to optimize memory access time; the <REFERENCE>(SSS) console program automatically interleaves the memories in the system unless the SET MEMORY command is used to set a specific interleave or no interleave (which would result in serial access to each memory module). Interleaving causes a number of memories to operate in parallel.

**Memory node**

Also called the <REFERENCE>(XMA). Memory is a global resource equally accessible by all processors on the <REFERENCE>(XMI). See also <REFERENCE>(XMA).

**Module**

A single <REFERENCE>(XMI) or VAXBI card that is housed in a single slot in its respective card cage. XMI modules (11.02" x 9.18") are larger than VAXBI modules (8.0" x 9.18").

**<REFERENCE>(XMA)**

XMI memory array; a memory subsystem of the XMI. Memory is a global resource equally accessible by all processors on the <REFERENCE>(XMI). A memory module can have 32, 64, or 128 Mbytes of memory, consisting of MOS 1-Mbit or MOS 4-Mbit dynamic RAMs, ECC logic, and control logic.

**Node**

An <REFERENCE>(XMI) node is a single module that occupies one of the 14 logical and physical slots on the <REFERENCE>(XMI) bus. A VAXBI node consists of one or more VAXBI modules that form a single functional unit.

**Node ID**

A hexadecimal number that identifies the node location. On the <REFERENCE>(XMI) bus, the node ID is the same as the physical location. On the VAXBI, the source of the node ID is an ID plug attached to the backplane.

**Pended bus**

A bus protocol in which the transfer of command/address and the transfer of data are separate operations. The <REFERENCE>(XMI) bus is a pended bus.

**Primary processor**

See *Boot processor*.

**Processor node**

A VAX processor that contains a central processor unit (CPU), executes instructions, and manipulates data contained in memory.

**RBD**

ROM-based diagnostics.

**RBV20/RBV64**

VAXBI adapter for write-once-read-many (WORM) optical disk drive. The RBV20 and RBV64 controllers use the KLESI-B adapter.

**Scalar/vector processor pair**

The FV64A vector processor functions as a coprocessor with a host scalar processor. The scalar/vector processor pair appear as one processor to an executing program.

**Secured terminal**

Console terminal in program mode while the machine is processing.

**Glossary-4**

**Shadow set**

Two disks functioning as one disk, each shadowing the information contained on the other, controlled by an HSC controller under the VMS operating system.

**Symmetric multiprocessing**

A multiprocessing system configuration in which all processors have equal access to operating system code residing in shared memory and can perform all, or almost all, system tasks.

**System root**

In a BOOT command, the argument to the /R5 qualifier.

**TBK70**

VAXBI adapter connecting the TK tape drive to the system.

**TU81E**

VAXBI adapter for a local (nonclustered) tape subsystem. The TU81E controller uses the KLESI-B adapter.

**VAX Diagnostic Supervisor (VAX/DS)**

Software that loads and runs diagnostic and utility programs.

**VAXBI bus**

The 32-bit bus used for I/O.

**VAXBI Corner**

The portion of a VAXBI module that connects to the backplane and provides an electrically identical interface for every VAXBI node.

**VMB**

The virtual memory boot program (VMB.EXE) that boots the operating system. VMB is the primary bootstrap program and is stored on the boot device. The goal of booting is to read VMB from the boot device and load the operating system.

**XBI**

Lines in the self-test display that show the status of <REFERENCE>(XBI) adapters and of VAXBI nodes. See also <REFERENCE>(xbi).

**XMI**

The 64-bit, high-speed system bus.

**<REFERENCE>(XMI) Corner**

The portion of an <REFERENCE>(XMI) module that connects to the backplane and provides an electrically identical interface for every XMI node.

## A

---

Architecture, 1-2  
    with vector processors, 4-4  
Autosizer program, 2-66

## B

---

Backup cache, 3-7  
Booting  
    boot error messages, B-1 to B-5  
    boot status messages, B-1 to B-5  
    over Ethernet, 2-62  
Boot processor, 3-10 to 3-11  
    how to replace, 3-30  
Boot processor's EEPROM, 3-33  
BPD  
    in self-test display, 2-6, 4-11

## C

---

Cache, 3-7  
    backup, 3-7  
CD server, 2-63  
Configuration rules  
    memory, 5-4  
    <REFERENCE>(xbi) adapter, 6-6 to 6-7  
    scalar processor, 3-4 to 3-5  
    vector processor, 4-6  
Console  
    error messages, A-1 to A-11  
Console commands, 3-22 to 3-23  
    for interleaving, 5-10 to 5-11  
    vector processor, 4-18  
Console display, 2-6 to 2-7

## D

---

Diagnostics  
    design, 2-2  
    overview, 2-2 to 2-3  
    ROM-based, 2-2, 2-40 to 2-59  
    self-test, 2-2  
    VAX/DS, 2-2, 3-18, 4-16  
    XGPR use, 4-12  
Diagnostic Supervisor  
    See VAX/DS  
Disable fault parse tree  
    <REFERENCE>(xrv), D-14 to D-15  
DWMBB adapter, 6-1 to 6-12  
    configuration rules, 6-6 to 6-7  
    functional description, 6-8 to 6-9  
    physical description, 6-2 to 6-3  
    registers, 6-10 to 6-12  
    specifications, 6-4 to 6-5

## E

---

EEPROM  
    restoring corrupted, E-1 to E-5  
    version number, 3-9  
EEPROM, patching, 3-34 to 3-35  
Error messages  
    console, A-1 to A-11  
ETF  
    in self-test display, 2-6, 4-11  
EVSBA, 2-66  
EVUCA, 2-61, 3-34 to 3-35  
Extended test, 3-13

## F

---

Fatal error, defined, 2-24  
Floating-point accelerator chip, 3-7  
FV64A module  
    inserting in XMI card cage, 4-25

## H

---

Hard error, defined, 2-24  
Hard error interrupt parse tree  
    <REFERENCE>(xmp), D-7 to D-9  
    <REFERENCE>(xrv), D-13

## I

---

I/O adapters, 1-3  
Interleaving, 5-8 to 5-11  
    default, 5-9  
    manual, 5-9

## L

---

LEDs  
    processor, error code in, 2-10 to 2-13, C-1 to C-3  
    processor error, 2-14 to 2-15  
    status, 2-8 to 2-9

## M

---

Machine check parse tree  
    <REFERENCE>(xmp), D-1 to D-6  
    <REFERENCE>(xrv), D-12  
Machine checks, 3-20, 4-17  
Memory  
    See MS65A memory  
Module handling, 3-24 to 3-27, 4-22 to 4-25  
MS65A memory, 1-3, 5-1 to 5-19  
    addressing, 5-12 to 5-13  
    configuration rules, 5-4  
    features, 5-3  
    functional description, 5-6 to 5-7

## MS65A memory (Cont.)

    good and bad memory pages, 5-17  
    interleaving, 5-8 to 5-11  
    physical description, 5-2 to 5-3  
    power-up, 5-14  
    registers, 5-18 to 5-19  
    self-test, 5-14 to 5-17  
    self-test errors, 5-16 to 5-17  
    specifications, 5-5  
    yellow LED, 5-17  
MSSC, 3-9  
MTPR/MFPR instructions, 3-39, 4-29  
MTVP/MFVP instructions, 4-29

## P

---

Parse trees, D-1 to D-15  
    <REFERENCE>(XMP), D-1 to D-12  
    <REFERENCE>(XRV), D-12 to D-15  
Patching the EEPROM, 3-34 to 3-35  
Power-up  
    processor, 3-12 to 3-15  
Power-up tests, 2-3  
Primary processor  
    See Boot processor  
Processor, 1-3, 3-1 to 3-41  
    See also Vector processor  
    boot, 3-10 to 3-11  
    configuration rules, 3-4 to 3-5  
    console commands, 3-22 to 3-23  
    error LED, 2-14 to 2-15  
    functional description, 3-6 to 3-9  
    handling procedures, 3-24 to 3-27  
    how to add new, 3-32  
    how to replace boot, 3-30  
    how to replace only, 3-28  
    how to replace secondary, 3-32

## Processor (Cont.)

- LEDs, 2-8 to 2-9, 2-10 to 2-13, C-1 to C-3
  - machine checks, 3-20
  - module, inserting in XMI card cage, 3-26 to 3-27
  - physical description, 3-2
  - power-up, 3-12 to 3-15
  - registers, 3-36 to 3-41
  - self-test, 3-13
  - specifications, 3-3
  - XMI interface, 3-7
- Processor chip, 3-7
- Progress trace, 2-6, 4-10

## R

---

### RBDs

- See ROM-based diagnostics
- <REFERENCE>(XBI) adapter, 1-3
- <REFERENCE>(XMI)-to-VAXBI adapter, 1-3
- See also DWMBB adapter
- <REFERENCE>(xmp) processor, 3-1 to 3-41
- See also Processor
- <REFERENCE>(XRV) processor, 4-1 to 4-29

See also Vector processor

### Registers

- MS65A memory, 5-18 to 5-19
  - processor, 3-36 to 3-41
  - <REFERENCE>(xbi) adapter, 6-10 to 6-12
  - vector processor, 4-28 to 4-29
- Repair tag, 3-27
- Reset, 2-27

- ROM-based diagnostics, 2-3, 2-40 to 2-59, 3-16, 4-14
- cache tests, 2-56 to 2-57
  - callable tests, 2-18, 2-40
  - commands, 2-18
    - QUIT, 2-27
    - START, 2-22 to 2-25
    - SUMMARY, 2-30 to 2-31

## ROM-based diagnostics (Cont.)

- control characters, 2-20 to 2-21
  - CPU/memory interaction tests, 2-46 to 2-47
  - entering RBD mode, 2-19
  - exiting RBD mode, 2-19
  - /HE, 2-19
  - I/O devices, 2-38 to 2-39
  - memory, 2-52 to 2-55
  - multiprocessor tests, 2-58 to 2-59
  - operator-invoked, 2-40 to 2-41
  - overview, 2-2
  - program, 2-18 to 2-19
  - <REFERENCE>(xbi) adapter, 2-48 to 2-51
  - run at power-up, 2-4 to 2-17
  - sample session, 2-32 to 2-37
  - self-test, 2-42 to 2-45
  - system reset in, 2-27
  - test printout
    - explanation, 2-26 to 2-29
    - failing, 2-28 to 2-29
    - passing, 2-26 to 2-27
    - sample, 2-32 to 2-37
  - tests, 2-4
  - /TR, 2-19
- ROM version number, 3-9

## S

---

### Secondary processor

- how to replace, 3-32
- Self-test, 2-3, 2-4 to 2-17
- display, 2-6, 3-13, 4-10
  - processor, 3-13
  - results, 2-6 to 2-11, 2-16 to 2-17
  - when invoked, 2-5
- Serial number, 3-28
- Single processor
- how to replace, 3-28
- Soft error, defined, 2-24
- Soft error interrupt parse tree



Soft error interrupt parse tree  
(Cont.)  
    <REFERENCE>(xmp), D-10 to  
        D-11  
    <REFERENCE>(xrv), D-14  
Specifications  
    scalar processor, 3-3  
    vector processor, 4-5  
STF  
    in self-test display, 2-6, 4-11  
System  
    architecture, 1-2, 4-4  
    functional description, 1-2 to  
        1-3  
    serial number, 3-28  
System reset, 2-27  
System support, 3-7  
System support chip  
    See MSSC

## T

---

Troubleshooting flowcharts, 1-4 to  
    1-11  
TYP  
    in self-test display, 2-6, 4-11

## V

---

VAX/DS, 2-3, 2-60 to 2-73  
    description, 2-61  
    diagnostics, 2-70 to 2-73, 3-18,  
        4-16  
    documentation, 2-60  
    exerciser tests, 2-61  
    explanation of levels, 2-60  
    function tests, 2-61  
    HELP in, 2-61  
    logic tests, 2-61  
    running in user mode, 2-64 to  
        2-65  
    running standalone, 2-62 to  
        2-63  
    sample session, 2-66 to 2-69  
    types of diagnostic programs,  
        2-61

VAXBI nodes, running RBD, 2-38  
VAX Diagnostic Supervisor  
    See VAX/DS  
Vector console commands, 4-18  
Vector processor, 4-1 to 4-29  
    configuration rules, 4-6  
    console commands, 4-18  
    functional description, 4-8  
    handling procedures, 4-22 to  
        4-25  
    how to replace, 4-26  
    machine checks, 4-17  
    memory requirements, 4-5  
    physical description, 4-2 to 4-3  
    registers, 4-28 to 4-29  
    self-test, 4-10  
    specifications, 4-5  
    XMI transactions, 4-9

## X

---

XBER register, 2-16 to 2-17  
XGPR register, 2-16 to 2-17, 4-12  
XMI interface, 3-7  
XMI nodes, running RBD, 2-38