VAX 6000 Series Owner's Manual

Order Number EK–600EB–OM.002

This manual is intended for the system manager or system operator and covers the daily operations of a VAX 6000 series system.

digital equipment corporation
maynard, massachusetts

First Printing, October 1990
Revised, November 1991

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 The VAX 6000 Series System

1.1 System Characteristics 1–2
1.2 System Architecture 1–4
1.3 Sample System 1–6
1.4 System Front View 1–8
1.5 System Rear View 1–10
1.6 Supported Adapters 1–12

Chapter 2 System Components

2.1 Console Load Devices 2–2
2.1.1 Ethernet-Based Compact Disk Server 2–4
2.1.2 In-Cabinet Tape Drive 2–6
2.2 Power System 2–8
2.3 XMI Card Cage 2–10
2.4 I/O Connections 2–12
2.5 Cooling System 2–14
2.6 Options 2–16

Chapter 3 Controls and Indicators

3.1 Control Panel 3–2
3.2 Upper Key Switch 3–4
3.3 Lower Key Switch 3–6
3.4 Restart Button 3–8
3.5 Status Indicator Lights 3–10
3.6 Circuit Breaker 3–12

Chapter 4 Booting

4.1 How Booting Works 4–2
4.2 Boot Devices 4–4
4.3 Regular Boot Procedure 4–6
4.4 Boot Device Selection 4–8
4.5 Boot Processor Selection 4–10
4.6 Booting from an HSC Disk 4–12
4.6.1 VAXcluster Boot Overview 4–12
4.6.2 Sample VAXcluster Boot 4–14
4.7 Booting from an Ethernet-Based Compact Disk Server 4–16
4.7.1 CD Server Boot Command 4–16
4.7.2 Selecting an Ethernet Service 4–18
4.8 Ethernet Boot Overview 4–20
4.9 Sample Target-Initiated Ethernet Boot 4–22
4.9.1 Step 1, Gather Information at Target Node 4–22
4.9.2 Step 2, Enter Information into Executor’s NCP Volatile Database 4–24
4.9.3 Step 3, Boot from the Target Node 4–26

Chapter 5 Console

5.1 Description of Console 5–2
5.2 Console Functions 5–4
5.3 Console Mode 5–6
5.4 Console Command Language Control Characters 5–8
5.5 Console Command Language Syntax 5–10
5.6 BOOT 5–12
5.6.1 BOOT Command Examples and Qualifiers 5–12
5.6.2 BOOT Command Description 5–14
5.7 CLEAR EXCEPTION 5–16
5.8 CONTINUE 5–18
5.9 DEPOSIT 5–20
5.9.1 Syntax and Qualifiers 5–20
5.9.2 Examples 5–22
5.10 EXAMINE 5–24
5.10.1 Syntax and Qualifiers 5–24

5.10.2 Examples 5–26
5.11 FIND 5–28
5.12 HALT 5–30
5.13 HELP 5–32
5.14 INITIALIZE 5–34
5.15 REPEAT 5–36
5.16 RESTORE EEPROM 5–38
5.17 SAVE EEPROM 5–40
5.18 SET Commands 5–42
5.18.1 SET BOOT 5–44
5.18.2 SET CPU 5–46
5.18.2.1 Syntax and Qualifiers 5–46
5.18.2.2 Examples 5–48
5.18.3 SET LANGUAGE 5–50
5.18.4 SET MEMORY 5–52
5.18.5 SET TERMINAL 5–54
5.19 SHOW 5–56
5.20 START 5–58
5.21 STOP 5–60
5.22 TEST 5–62
5.23 UNJAM 5–64
5.24 UPDATE 5–66
5.25 Z 5–68
5.26 ! 5–70
5.27 Sample Console Session 5–72

Chapter 6 System Self-Test and Troubleshooting

6.1 Self-Test Overview 6–2
6.2 Sample Self-Test Display 6–4
6.3 Self-Test Progress Trace Line 6–6
6.4 Self-Test Lines NODE #, TYP, and STF 6–8
6.5 Self-Test Lines BPD and ETF 6–10
6.6 Self-Test Lines ILV and Mb 6–12
6.7 Self-Test Identification Line 6–14
6.8 Sample Self-Test Display with VAXBI Adapter 6–16

6.9 Troubleshooting During Booting 6–18
6.10 Forcing a Boot Processor 6–20

Appendix A Compact Disk Drive Instructions

A.1 Controls and Indicators A–1
A.2 Loading a Compact Disk A–2
A.3 Unloading a Compact Disk A–2
A.4 Cleaning Disks A–4

Appendix B TF/TK Tape Drive Instructions

B.1 Controls and Indicators B–2
B.2 Loading a Tape B–3
B.3 Unloading a Tape B–4
B.4 Write-Protecting Your Tape Cartridge B–4
B.5 Labeling a Tape Cartridge B–5
B.6 Tape Handling and Storage Guidelines B–5

Appendix C Device Type Code Assignments

Appendix D VAXBI Options and Adapters

D.1 Supported VAXBI Adapters D–1
D.2 Supported Boot Devices D–2
D.3 VAXBI Expander Cabinet D–3
D.4 Power for the VAXBI Option D–4

Appendix E EVUCA Program

E.1 EVUCA Program Overview E–1
E.2 Updating EEPROM Contents E–2

Appendix F Control Flags for Booting

Appendix G Console Commands

Appendix H Console Error Messages (Model 400 and Higher)

Appendix I Console Error Messages for Model 300

Appendix J Boot Status and Error Messages (Models 500 and 600)

J.1 Ethernet Boot Messages J–1
J.2 Local Disk Boot Messages J–2
J.3 Local Tape Boot Messages J–2
J.4 CI and DSSI Boot Messages J–3

Glossary

Index

Examples

4–1 Sample VAXcluster Boot 4–14
4–2 Sample Ethernet-Based CD Server Boot 4–16
4–3 Selecting an Ethernet Service 4–18
4–4 Step 1, SHOW Ethernet 4–22
4–5 Step 2, Entering Target Node Information 4–24
4–6 Step 3, Booting from the Target Node 4–26
6–1 Forcing a Boot Processor 6–20
E–1 Updating EEPROM Contents E–3

Figures

1–1	Sample System Footprint	1–2
1–2	System Architecture	1–4
1–3	Sample System	1–6
1–4	System Front View	1–8
1–5	System Rear View	1–10
1–6	Adapters	1–12
2–1	Console Load Devices	2–2
2–2	Ethernet-Based Compact Disk Server	2–4
2–3	Accessing the Ethernet-Based CD Server	2–5
2–4	TF85 Tape Drive	2–6
2–5	Power System (Rear View)	2–8
2–6	XMI Card Cage	2–10
2–7	Console and Terminal Connectors	2–12
2–8	Airflow Pattern	2–14
2–9	System Options	2–16
3–1	International and English Control Panels	3–2
3–2	Upper Key Switch (Enable Position)	3–4
3–3	Lower Key Switch (Update Position)	3–6
3–4	Restart Button	3–8
3–5	Control Panel Status Indicator Lights	3–10
3–6	Circuit Breaker and the AC Power Controller	3–12
4–1	Boot Procedure	4–2
4–2	Boot Devices	4–4
4–3	Regular Boot Procedure	4–6
4–4	Determining the Boot Processor	4–10
4–5	Booting from a CI-Based VAXcluster	4–12
4–6	Trigger Booting Using Ethernet	4–20
4–7	Target-Initiated Booting by Ethernet	4–20
5–1	Console	5–2
5–2	Console Switch When in Console Mode	5–6
5–3	BOOT Command	5–14
5–4	Lower Key Switch in Update Position	5–42
6–1	Testing Sequence	6–2
6–2	Self-Test Results	6–4
6–3	Self-Test Results: Progress Trace	6–6

6–4	Self-Test Results: NODE #, TYP, and STF	6–8
6–5	Self-Test Results: BPD and ETF	6–10
6–6	Self-Test Results: ILV and Mb	6–12
6–7	Self-Test Results: Identification Line	6–14
6–8	Self-Test Results: TYP, STF, and XBI Lines	6–16
6–9	Troubleshooting Booting	6–18
A–1	RRD Compact Disk Drive	A–1
A–2	Loading a Compact Disk	A–3
A–3	Disk Caddy Parts	A–4
B–1	TF85 Tape Drive	B–2
B–2	Tape Cartridge	B–4
D–1	VAXBI Expander Cabinet	D–3

Tables

1	VAX 6000 Series Documentation	xii
2	VAX 6000 Model Level Documentation	xiii
3	Associated Documents	xiv
1–1	Electrical Characteristics	1–3
1–2	Environmental Characteristics	1–3
1–3	System Components	1–7
1–4	Adapters	1–13
2–1	AC Power Controller Input Voltage	2–9
2–2	Power Supply Available	2–9
3–1	Control Panel Symbols	3–3
3–2	Upper Key Switch	3–5
3–3	Lower Key Switch	3–7
3–4	Restart Button	3–9
3–5	Control Panel Status Indicator Lights	3–11
4–1	Boot Procedure	4–3
4–2	Boot Devices	4–5
4–3	Sample BOOT Commands	4–8
4–4	Boot Device Mnemonics	4–9
5–1	Console Parts and Functions	5–3
5–2	Console Functions	5–4
5–3	Console Control Characters	5–8
5–4	Console Command Language Syntax	5–10

5–5	BOOT Command Qualifiers	5–13
5–6	DEPOSIT Command Qualifiers	5–20
5–7	EXAMINE Command Qualifiers	5–24
5–8	FIND Command Qualifiers	5–28
5–9	INITIALIZE Command Qualifiers	5–35
5–10	SET CPU Command Qualifiers	5–46
5–11	SET CPU Command Qualifiers’ Effect After a System Reset	5–49
5–12	SET LANGUAGE Command Parameters	5–50
5–13	SET MEMORY Qualifiers	5–52
5–14	SET TERMINAL Command Qualifiers	5–54
5–15	SHOW Commands	5–57
5–16	STOP Command Qualifiers	5–60
5–17	TEST Command Qualifiers	5–62
5–18	Z Command Qualifiers	5–68
6–1	System Configuration for Sample Self-Test	6–5
A–1	RRD Light Summary	A–2
B–1	TF85 Light Summary	B–3
C–1	XMI Device Type Code Assignments	C–1
D–1	VAXBI Adapters	D–1
D–2	VAXBI Boot Devices	D–2
D–3	In-Cabinet VAXBI Power	D–4
E–1	EEPROM Update Data Files	E–1
F–1	R5 Bit Functions for VMS	F–1
F–2	R5 Bit Functions for ULTRIX	F–2
G–1	Console Commands and Qualifiers	G–1
H–1	Console Error Messages Indicating Halt (Model 400 and Higher)	H–2
H–2	Standard Console Error Messages (Model 400 and Higher)	H–3
H–3	Console Error Messages for Models 500 and 600	H–12
H–4	Console Error Messages Unique to Model 600	H–12
I–1	Model 300 Console Error Messages Indicating Halt	I–1
I–2	Model 300 Standard Console Error Messages	I–3

Preface

Intended Audience

This manual is written for the system manager or system operator who has had training in VAX systems and system management tasks and is running a VAX 6000 series 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 are descriptive text and syntax definitions.

This manual has six chapters and ten appendixes, as follows:

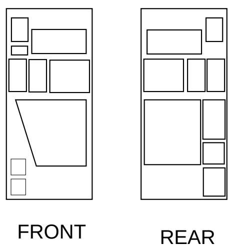
- **Chapter 1, The VAX 6000 Series System**, and **Chapter 2, System Components**, give you a basic introduction to your system and its parts.
- **Chapter 3, Controls and Indicators**, describes how the system presents information and how you use the switches.
- **Chapter 4, Booting**, explains how you turn the system on and get it running.
- **Chapter 5, Console**, explains the console environment, use, and console commands. It includes a sample console session.
- **Chapter 6, System Self-Test and Troubleshooting**, describes self-test in detail and tells you what to do if something goes wrong.
- The **Appendixes** give in-depth information on topics covered in the manual. Appendix A contains RRD compact disk drive instructions, Appendix B contains TF/TK tape drive instructions, Appendix C lists the device type code assignments, Appendix D describes VAXBI options and adapters, Appendix E gives detailed information on the EVUCA program, Appendix F lists control flags for booting, Appendix G summarizes the console commands, Appendix H lists console error

messages for Model 400 and higher systems. Appendix I has console error messages for Model 300 systems, and Appendix J contains boot status and error messages for Model 500 and 600 systems.

- 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 series systems. A shaded area in the icon shows the location of the component or part being discussed.



VAX 6000 Series 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

Title	Order Number
Operation	
VAX 6000 Series Owner's Manual	EK–600EB–OM
VAX 6000 Series Vector Processor Owner's Manual	EK–60VAA–OM
VAX 6000 Vector Processor Programmer's Guide	EK–60VAA–PG

Table 1 (Cont.): VAX 6000 Series Documentation

Title	Order Number
Service and Installation	
VAX 6000 Platform Technical User's Guide	EK–600EA–TM
VAX 6000 Series Installation Guide	EK–600EB–IN
VAX 6000 Installationsanleitung	EK–600GB–IN
VAX 6000 Guide d'installation	EK–600FB–IN
VAX 6000 Guia de instalacion	EK–600SB–IN
VAX 6000 Platform Service Manual	EK–600EA–MG
Options and Upgrades	
VAX 6000: XMI Conversion Manual	EK–650EB–UP
VAX 6000: Installing MS65A Memories	EK–MS65A–UP
VAX 6000: Installing the H7236-A Battery Backup Option	EK–60BBA–IN
VAX 6000: Installing the FV64A Vector Option	EK–60VEA–IN
VAX 6000: Installing the VAXBI Option	EK–60BIA–IN

Manuals specific to models are listed in Table 2.

Table 2: VAX 6000 Model Level Documentation

Title	Order Number
Model 600	
VAX 6000 Model 600 Mini-Reference	EK–660EA–HR
VAX 6000 Model 600 Service Manual	EK–660EA–MG
VAX 6000 Model 600 System Technical User's Guide	EK–660EA–TM
VAX 6000: Installing Model 600 Processors	EK–660EA–UP

Table 2 (Cont.): VAX 6000 Model Level Documentation

Title	Order Number
Model 500	
VAX 6000 Model 500 Mini-Reference	EK–650EA–HR
VAX 6000 Model 500 Service Manual	EK–650EA–MG
VAX 6000 Model 500 System Technical User's Guide	EK–650EA–TM
VAX 6000: Installing Model 500 Processors	EK–KA65A–UP
Models 200/300/400	
VAX 6000 Model 300 and 400 Service Manual	EK–624EA–MG
VAX 6000: Installing Model 200/300/400 Processors	EK–6234A–UP

Associated Documents

Table 3 lists other documents that you may find useful.

Table 3: Associated Documents

Title	Order Number
System Hardware Options	
VAXBI Expander Cabinet Installation Guide	EK–VBIEA–IN
VAXBI Options Handbook	EB–32255–46
System I/O Options	
CIBCA User Guide	EK–CIBCA–UG
CIXCD Interface User Guide	EK–CIXCD–UG
DEC LANcontroller 200 Installation Guide	EK–DEBNI–IN
DEC LANcontroller 400 Installation Guide	EK–DEMNA–IN
DSSI VAXcluster Installation Manual	EK–DVCLU–IN
InfoServer Installation Guide	EK–DIS1K–IN
KDB50 Disk Controller User's Guide	EK–KDB50–UG

Table 3 (Cont.): Associated Documents

Title	Order Number
System I/O Options	
KDM70 Controller User Guide	EK–KDM70–UG
KFMSA Module Installation and User Manual	EK–KFMSA–IM
KFMSA Module Service Guide	EK–KFMSA–SV
RRD40 Disc Drive Owner's Manual	EK–RRD40–OM
RA90/RA92 Disk Drive User Guide	EK–ORA90–UG
RF31/RF72 Integrated Storage Element Installation Manual for BA200–Series Enclosures	EK–RF72D–IM
RF31/RF72 Integrated Storage Element User Guide	EK–RF72D–UF
RF31/RF72 Integrated Storage Element Service Guide	EK–RF72D–SV
SA70 Enclosure User Guide	EK–SA70E–UG
SF200 Storage Array Installation Guide	EK–SF200–IG
SF72 Storage Enclosure and SF200 Storage Array Cabinet Service Guide	EK–SF72S–SG
TF85 Cartridge Tape Subsystem Owner's Manual	EK–TF85–OM
TF857 Magazine Tape Subsystem Service Manual	EK–TF857–OM
VAX 6000 SF2xx Embedded Storage Installation Guide	EK–EMBED–IN
Operating System Manuals	
Guide to Maintaining a VMS System	AA–LA34B–TE
Guide to Setting Up a VMS System	AA–LA25A–TE
Introduction to VMS System Management	AA–LA24A–TE
ULTRIX–32 Guide to System Exercisers	AA–ME96B–TE
VMS Networking Manual	AA–LA48A–TE
VMS System Manager's Manual	AA–LA00B–TE
VMS Upgrade and Installation Supplement: VAX 6000 Series	AA–LB36C–TE

Table 3 (Cont.): Associated Documents

Title	Order Number
VAXclusters and Networking	
DECbridge 500 Installation Guide	EK–DEFEB–IN
DEMFA Installation Guide ??	EK–?????–??
Fiber Distributed Data Interface Description	EK–DFSLD–SD
Guidelines for VAXcluster System Configurations	EK–VAXCS–CG
H4000 Digital Ethernet Transceiver Installation Manual	EK–H4000–IN
HSC Installation Manual	EK–HSCMN–IN
VAXcluster Principles	EK–VAXCP–TM
VAX 6000/VAX 4000—300 SF200 System Manual	TBD
VMS VAXcluster Manual	AA–LA27B–TE
Peripherals	
Installing and Using the VT420 Video Terminal	EK–VT420–UG
RV20 Optical Disk Owner's Manual	EK–ORV20–OM
SC008 Star Coupler User's Guide	EK–SC008–UG
TA78 Magnetic Tape Drive User's Guide	EK–OTA78–UG
TA90 Magnetic Tape Subsystem Owner's Manual	EK–OTA90–OM
TK70 Streaming Tape Drive Owner's Manual	EK–OTK70–OM
TU81/TA81 and TU/81 PLUS Subsystem User's Guide	EK–TUA81–UG
VAX Manuals	
VAX Architecture Reference Manual	EY–3459E–DP
VAX Systems Hardware Handbook — VAXBI Systems	EB–31692–46
VAX Vector Processing Handbook	EC–H0739–46

Chapter 1

The VAX 6000 Series System

The VAX 6000 series computer system is designed for growth and can be configured for many different applications. Like other VAX systems, the VAX 6000 series system can support many users in a time-sharing environment. This system does the following:

- Supports a full range of VAX applications and operating systems
- Functions as a standalone system, a member of a VAXcluster, a boot node of a local area VAXcluster, or as a VAX file server for workstations
- Allows for expansion of processors, memory, and I/O
- Implements symmetric multiprocessing where all processors have equal access to memory
- Supports vector processors on Model 400 and Model 500 systems
- Uses a high-bandwidth system bus designed for multiprocessing
- Performs automatic self-test on power-up, reset, reboot, or system initialization
- Supports I/O devices on the VAXBI bus and provides access to the VMEbus

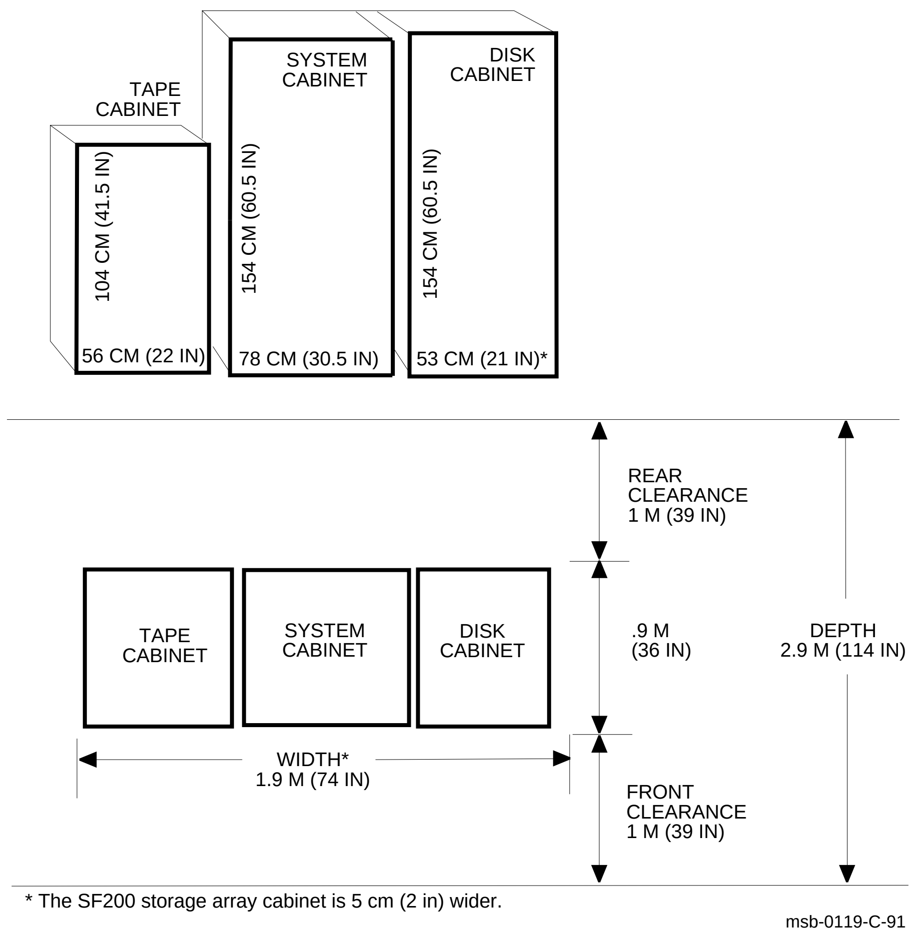
This chapter describes the system packages and introduces the location of components in the cabinet. Sections include:

- System characteristics
- System architecture
- Sample system
- System front view
- System rear view
- Supported adapters

1.1 System Characteristics

All VAX 6000 series systems share the same characteristics as shown in the tables. Figure 1–1 shows a system footprint.

Figure 1–1: Sample System Footprint



The values listed in Table 1–1 relate to the system cabinet only. In-cabinet storage will increase electrical requirements.

Table 1–1: Electrical Characteristics

Electrical[1]	Hz	Without VAXBI	With VAXBI
AC power consumption (max)		2.5 KVA	3.5 KVA
AC current (nom)	60	7.0 A (208 V)	9.7 A (208 V)
	50	3.5 A (416 V) 3.8 A (380 V)	4.8 A (416 V) 5.3 A (380 V)
Voltage input	60	3-phase 208 V RMS	
	50	3-phase 380/416 V RMS	
Frequency tolerance		47–63 Hz	
Surge current		60 A	

[1]These values assume a base system with no optional disks or battery backup unit.

Table 1–2: Environmental Characteristics

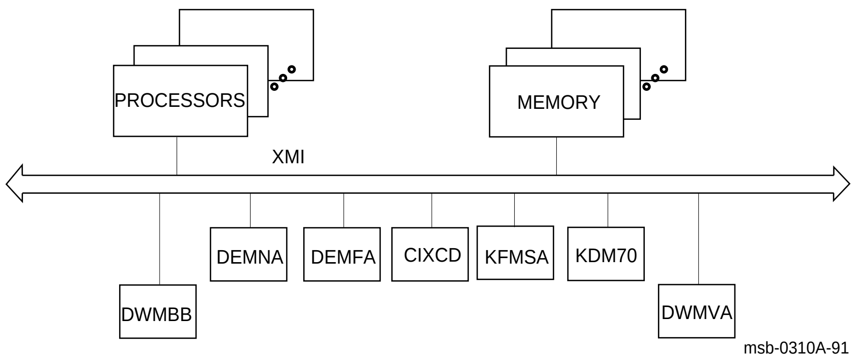
Environmental		
Heat dissipation (max)		5570 Btu/hr
Operating temperature	TF/TK not in use	10º to 40ºC (50º to 104ºF) [1]
	TF/TK in use	15º to 32ºC (59º to 90ºF)
Operating humidity	TF/TK not in use	10 to 90% relative humidity
	TF/TK in use	20 to 80% relative humidity
Altitude	Operating	0 to 2.4 km (0 to 8000 ft)
	Nonoperational	0 to 9.1 km (0 to 30,000 ft)

[1]Model 600 range: 15º to 32ºC (59º to 90ºF)

1.2 System Architecture

The high-speed XMI bus is used to interconnect processors, memory modules, and I/O adapters.

Figure 1–2: System Architecture



The XMI is the 64-bit system bus that interconnects the processors, memory modules, and I/O adapters.

The XMI bus uses the concept of a **node**. A node is a single functional unit that consists of one or more modules. The XMI has three types of nodes: processor nodes, memory nodes, and I/O adapters.

A **processor node** is a single-board scalar processor or a scalar/vector processor pair. Multiprocessing is supported on VAX 6000 systems. Up to six scalar processors can be used in most systems.[1] Models 400 and 500 support vector processing with multiple scalar/vector processor pairs. Symmetric multiprocessing is supported, allowing a program to execute on any processor.

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 (see Section 4.5).

A **memory node** is one memory module. Memory is a global resource equally accessible by all processors on the XMI. A memory module can have 32, 64, or 128 Mbytes of memory and associated ECC and control logic. The memories are automatically interleaved. An optional battery backup unit protects memory in case of power failure. The system supports up to eight memories.

I/O adapters are installed on the XMI bus (see Section 1.6). If your system has a VAXBI, the DWMBB adapter is used to connect VAXBI I/O adapters to the XMI bus. See Appendix D for information on VAXBI I/O adapters. The DWMVA adapter provides an interface to the VMEbus.

[1] Model 200 supports multiprocessing with up to four scalar processors.

1.3 Sample System

Figure 1–3 shows a sample VAX 6000 system. The system cabinet can have an optional console load device and optional in-cabinet disk drives. The system includes a console terminal and printer, an accessories kit, and a documentation set, which includes this manual. The system may have additional storage devices and may be a member of a VAXcluster.

Figure 1–3: Sample System

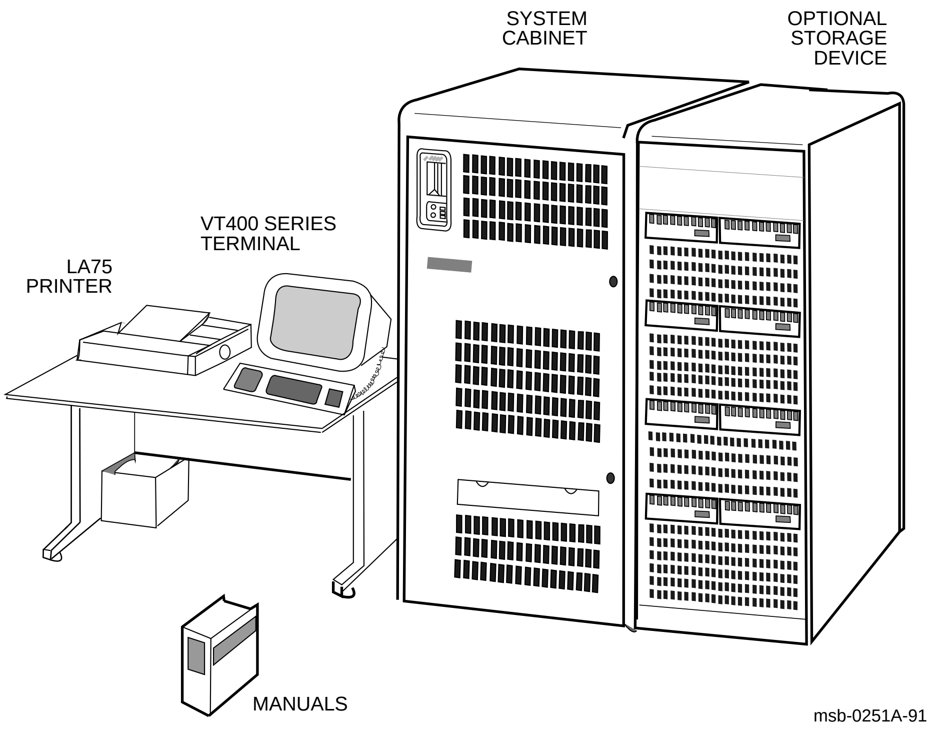


Table 1–3: System Components

Component	Function
System cabinet	Houses system components and optional storage
Console load device	Software distribution; stores and transfers data
Console terminal	Manages system and its resources
Console printer	Provides hardcopy of console transactions
Documentation	See the Preface for a full list of documentation related to VAX 6000 series systems
Storage cabinet	Provides additional storage capacity

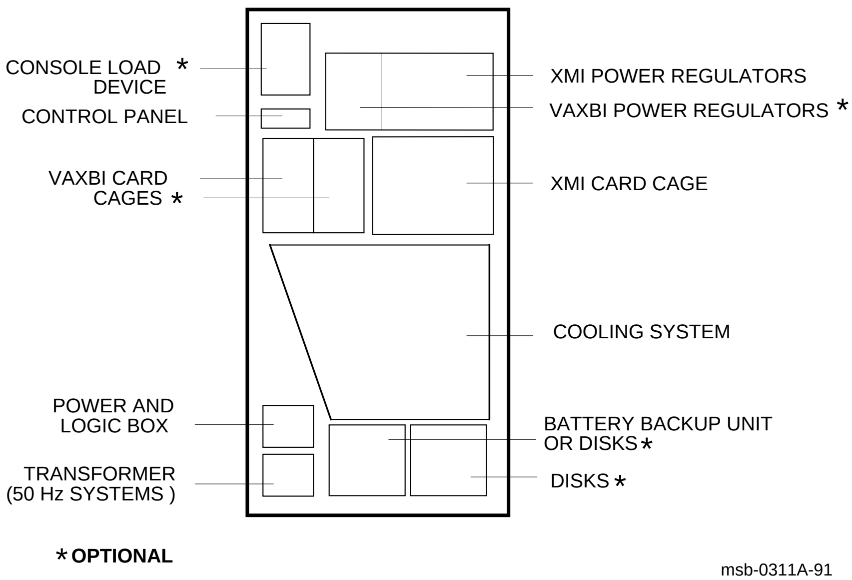
Your Digital customer service engineer has installed your system and verified that it is running properly. Before you turn on the system, familiarize yourself with its components:

- **The system cabinet** houses the XMI card cage (which contains the processors, memories, and I/O adapters) and the control panel with status indicators. Optional hardware in the cabinet includes a console load device, a VAXBI backplane, and disk drives.
- **The console load device** is used for installing operating systems, software, and some diagnostics. The console load device can be a tape drive, either in the cabinet or in the SF200 storage array, or it can be an Ethernet-based compact disk server.
- **A storage cabinet** provides local storage and archiving capability.
- **The console terminal** is used for booting and for system management operations.
- **A system documentation kit**

1.4 System Front View

The control panel and optional console load device and disk control panel are on the front of the system cabinet, accessible with the doors closed. With the front door open, Digital customer service engineers can access the power regulators, the XMI card cage and optional VAXBI card cages, the cooling system, and the optional battery backup unit.

Figure 1–4: System Front View



WARNING: *The inside of the system cabinet is not designed to be accessed by the customer. The information in this chapter is for your information only. The cabinet doors are to be opened only by Digital customer service engineers.*

These components are visible from the inside front of the cabinet (see Figure 1–4 for their location):

- Control panel
- XMI power regulators
- XMI card cage
- Cooling system
  One of the two blowers is visible from the front of the cabinet.
- Power and logic box
- Transformer (on 50 Hz systems only)

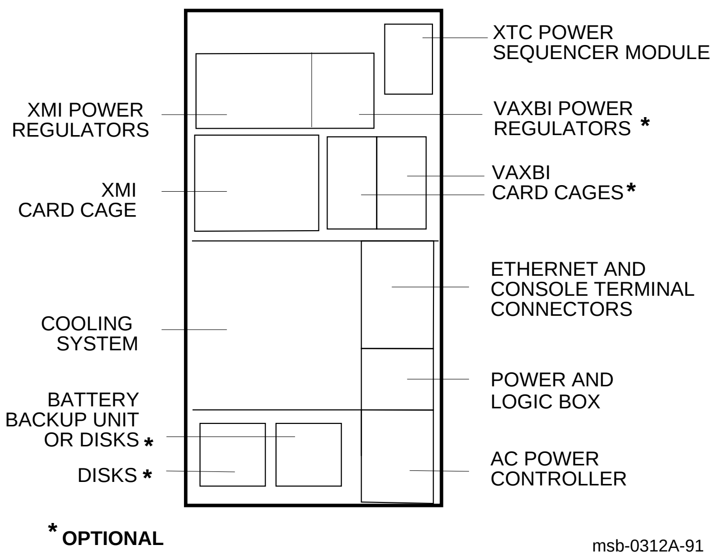
Optional components:

- Console load device
- VAXBI power regulators
- Two VAXBI card cages configured as one 12-slot channel
- Battery backup unit
- Disks

1.5 System Rear View

With the rear door open, Digital customer service engineers can access the power sequencer module (XTC); the power regulators; the I/O bulkhead space behind the card cages; Ethernet and console terminal connectors; cooling system; power and logic box; battery backup unit and disks, if present; and the AC power controller.

Figure 1–5: System Rear View



WARNING: *The inside of the system cabinet is not designed to be accessed by the customer. The information in this chapter is for your information only. The cabinet doors are to be opened only by Digital customer service engineers.*

These components are visible from the rear of the cabinet (see Figure 1–5):

- Power sequencer module (XTC) located on the back of the system control assembly
- XMI power regulators
- I/O bulkhead space
  The panel covering the XMI and VAXBI areas is the I/O bulkhead panel and provides space for additional I/O connections.
- XMI backplane and cables
- Ethernet and console terminal connectors
- Cooling system, with open grid over a blower
- Power and logic box
- AC power controller

Optional components:

VAXBI power regulators
VAXBI backplane and cables
Battery backup unit
Disks

1.6 Supported Adapters

VAX 6000 systems provide interfaces to other buses and to the Ethernet. Systems can be clustered and storage can be added and shared among systems. The system supports the following adapters: CIXCD, DEC LANcontroller 400 (DEMNA), DEMFA, DWMBB, DWMVA, KDM70, and KFMSA.

Figure 1–6: Adapters

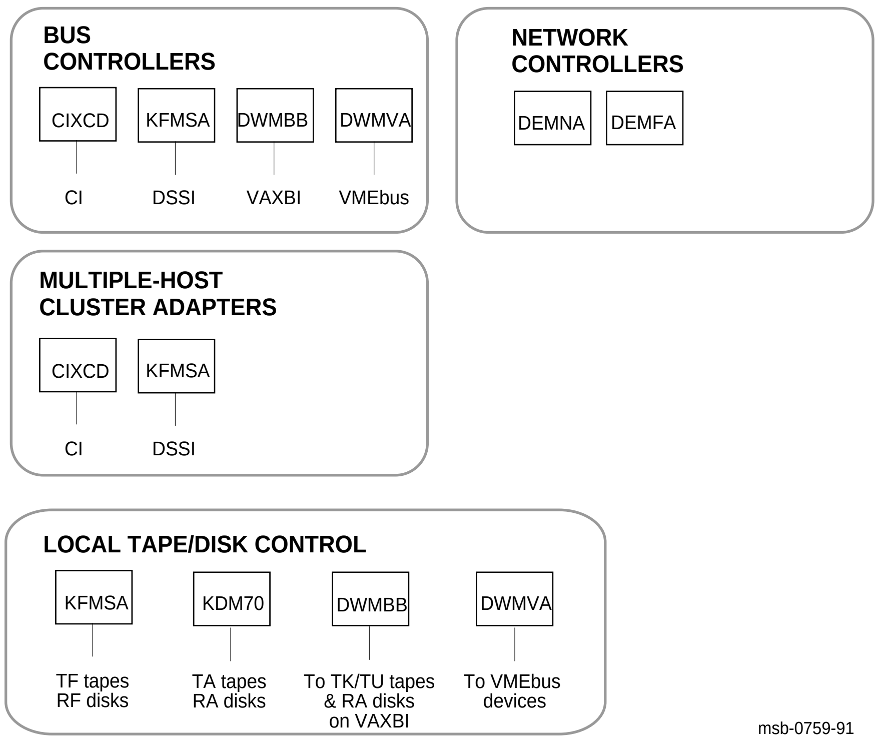


Table 1–4 describes the adapters supported by the system. Note that some adapters require more than one slot on the XMI.

Table 1–4: Adapters

Adapter	XMI Slots	Function
CIXCD	1	CI port interface; connects the system to a Star Coupler.
DEMFA	1	FDDI (fiber optic) port interface; connects a system to a local area network.
DEMNA	1	Ethernet port interface; connects a system to a local area network.
DWMBB	1	XMI-to-VAXBI interface, a two-module set. The DWMBB/A is in the XMI card cage; the DWMBB/B is installed in the VAXBI card cage.[1]
DWMVA	1	XMI-to-VMEbus interface, a two-module set. The DWMVA/A is in the XMI card cage; the DWMVA/B is installed in a VMEbus expansion cabinet.
KDM70	2	Disk adapter; enables connection to RA disk drives.
KFMSA	1	DSSI adapter; enables connection to TF tape drives and to RF disk drives.

[1]See Appendix D for information on VAXBI adapters and options.

For more information on adapters, see Digital's *Systems and Options Catalog* or the *VAX 6000 Platform Service Manual*.

Appendix C lists the device type codes for XMI adapters, and Appendix D lists the device type codes for VAXBI adapters.

Chapter 2
System Components

This chapter describes system components, noting their locations and functions. Sections include:

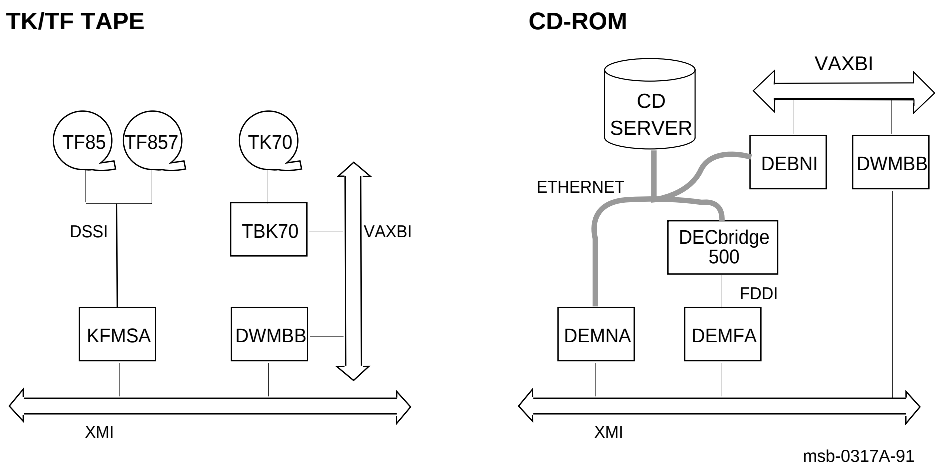
- Console load devices
  Ethernet-based compact disk server
  Tape drive
- Power system
- XMI card cage
- I/O connections
- Cooling system
- Options

WARNING: *The inside of the system cabinet is not designed to be accessed by the customer. The information in this chapter is for your information only. The cabinet doors are to be opened only by Digital customer service engineers.*

2.1 Console Load Devices

Figure 2–1 shows VAX 6000 console load devices and their adapters.

Figure 2–1: Console Load Devices



The console load device is used for:

- Installing or updating software
- Loading diagnostics
- Loading the standalone backup program
- Interchanging user data

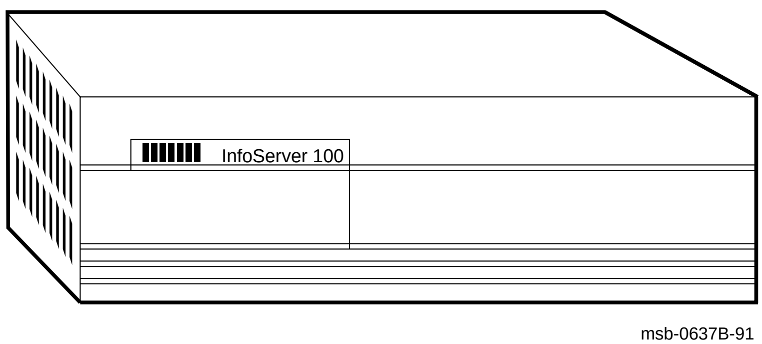
Any of the following can serve as the console load device:

- Ethernet-based compact disk (CD) server
- TF or TK tape drive in the system cabinet
- TF magazine tape subsystem in a storage cabinet

2.1.1 Ethernet-Based Compact Disk Server

The InfoServer is a console load device. During system installation the InfoServer can be used to boot the VAX Diagnostic Supervisor and standalone backup; it is not needed to load or initialize the system following installation.

Figure 2–2: Ethernet-Based Compact Disk Server

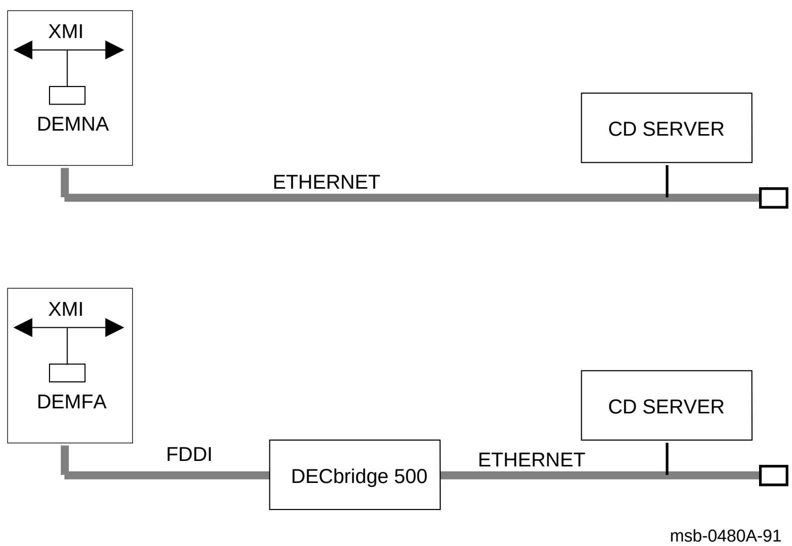


The InfoServer, an Ethernet-based compact disk (CD) server, is part of a local area network. The CD server functions as a read-only storage device for any system on the Ethernet. The CD server is used to access CD-ROMs for software installation, diagnostics, and on-line documentation.

The DEMNA adapter[1] and DEMFA adapter provide an interface to the Ethernet-based CD server. Both adapters are in the XMI card cage. However, as shown in Figure 2–3, the DEMFA provides access to the FDDI (Fiber Distributed Data Interface) network and requires a DECbridge 500 for connection to the Ethernet.

Section 4.7 describes how to boot VMS over the Ethernet using the CD server as the console load device. For more information on how to use the CD server, see the *InfoServer Installation Guide*.

Figure 2–3: Accessing the Ethernet-Based CD Server

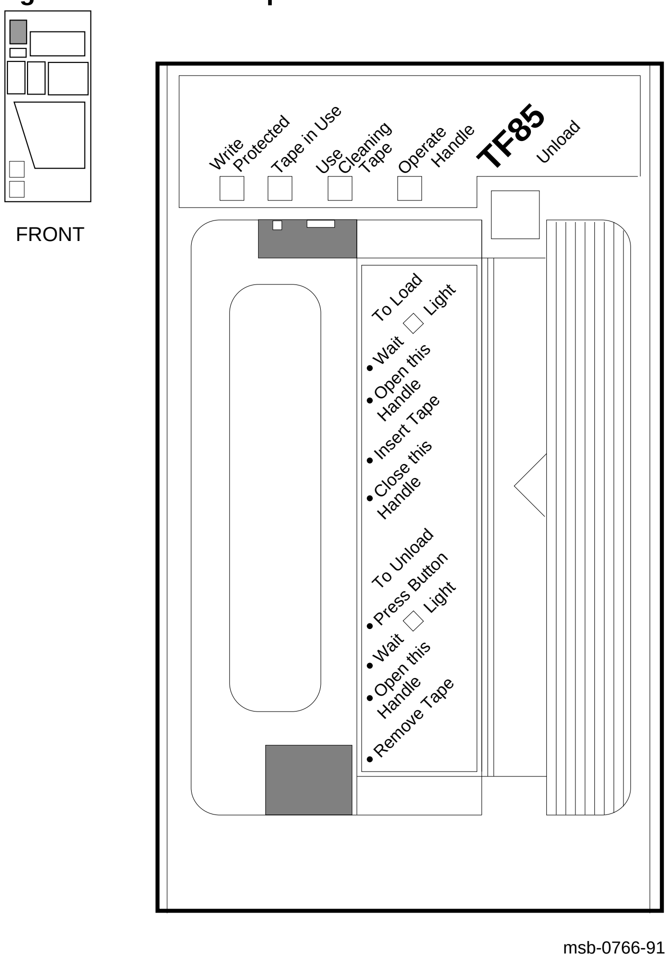


[1] The DEBNI or DEBNA adapter can also be used as an interface to the Ethernet-based compact disk server.

2.1.2 In-Cabinet Tape Drive

The system cabinet can have a TF or TK tape drive at the upper left of the cabinet. Either tape drive serves as a console load device.

Figure 2–4: TF85 Tape Drive



Three tape drives serve as console load devices. The operation of the TK70 tape drive is similar to that of the TF85 tape drive (see Appendix B for information on how to use the tape drive). The TF857 tape drive also serves as a console load device. The TF857 is a magazine tape subsystem in the SF2xx storage array. Both the TF85 and TF857 tape drives are DSSI devices and require the KFMSA adapter.

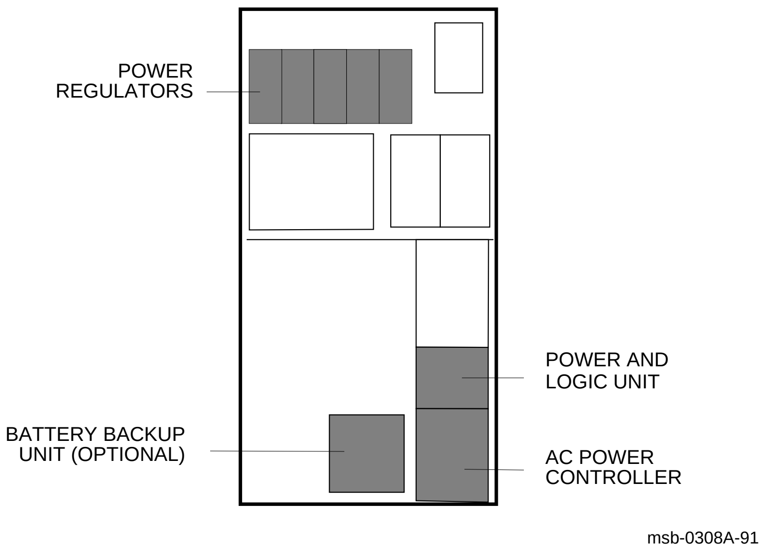
The TK70 tape drive requires the VAXBI option and is controlled by the TBK70 adapter, which resides in the VAXBI card cage.

For detailed information on how to use the tape drives, see the books listed in the Associated Documents section of the Preface.

2.2 Power System

The power system consists of an AC power controller with circuit breaker, the power and logic box, three power regulators, and an optional battery backup unit.

Figure 2–5: Power System (Rear View)



You can see most of the power system from the rear of the cabinet. The AC power controller with circuit breaker (see Section 3.6) is in the lower right corner. The power and logic box is just above the AC power controller. Across the top of the cabinet are the power regulators.

Table 2–1 gives the input voltage for the power controller. On the back of the power controller are two fuse-protected power outlets. Devices attached at these outlets are powered down only when the AC power controller circuit breaker is set to Off.

The XMI power supply is made up of three power regulators.[1] The power supply provides sufficient power for any combination of modules (see Table 2–2.)

Table 2–1: AC Power Controller Input Voltage

Model No.	Hz	Nominal Input Voltages	Phase
H405-E	60	208 V	3
H405-F	50	380 V	3
	50	416 V	3

Table 2–2: Power Supply Available

DC Voltage	Available XMI Current
+5V	124.0 A
+3.3V	80.0 A
+12V	4.0 A
–12V	2.5 A
–5.2V	20.0 A
–2V	7.0 A

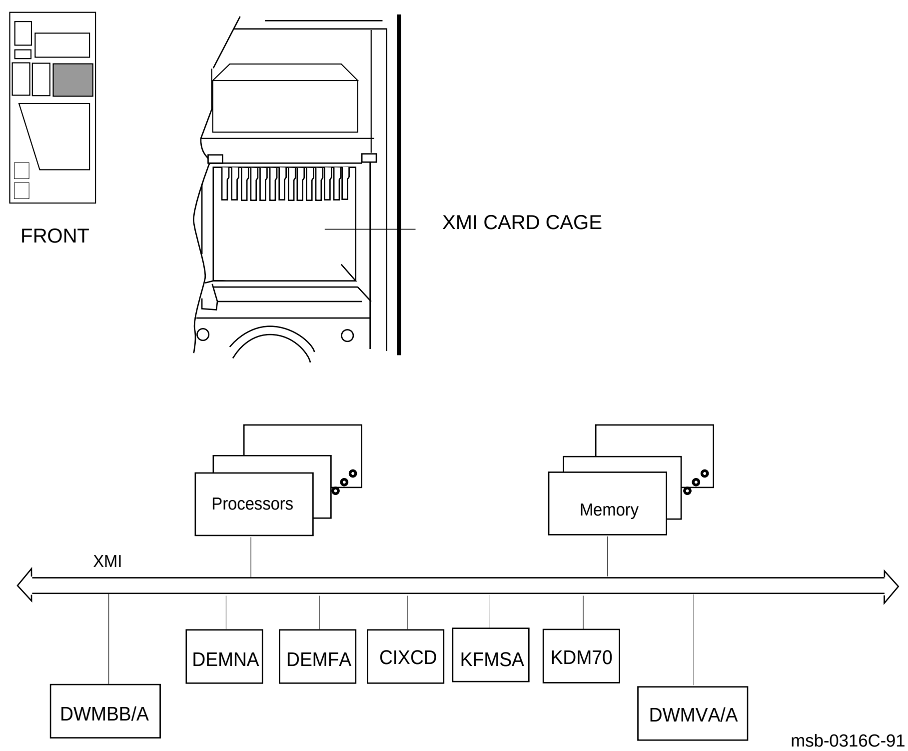
The optional battery backup unit (BBU), if present, is located near the bottom of the cabinet. This unit supplies power to sustain memory for up to 10 minutes following power interruption to system memory. The unit also provides 1 second of ride-through capability. The BBU has its own fan, which operates only when power is being supplied by the BBU. The system control panel indicates the status of the battery backup unit.

[1] Two additional power regulators are required if the system includes the optional VAXBI.

2.3 XMI Card Cage

The 14-slot XMI card cage houses processors, memories, and adapters. The XMI high-speed system bus interconnects the modules; it has a maximum bandwidth of 100 Mbytes per second and supports up to six processors.

Figure 2–6: XMI Card Cage



The system bus, the XMI, allows several transactions to occur simultaneously, making efficient use of the bus bandwidth. The bus includes the XMI backplane, the electrical environment of the bus, the protocol that nodes use on the bus, and the logic to implement this protocol.

The 14-slot XMI card cage is located in the upper third of the cabinet on the right side, as viewed from the front of the cabinet. A clear latched door protects the components housed in the card cage and helps to direct the airflow over the modules. Indicator lights on the XMI modules can be viewed through this clear front door. The protective door must be closed for operation, as it controls the power supply to the card cage.

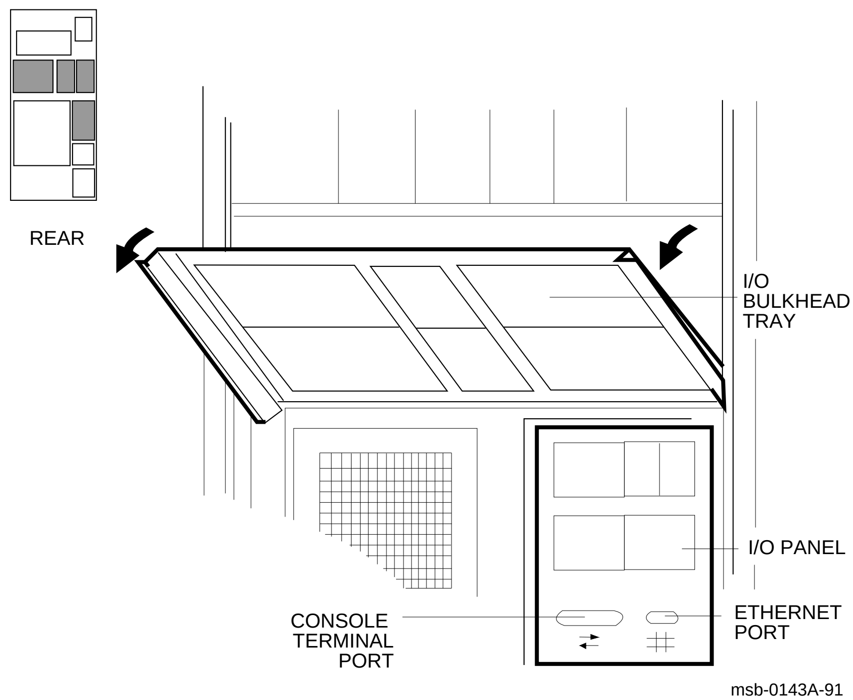
Each slot of the XMI card cage is hardwired to a 4-bit node ID code that corresponds to the physical slot number in the card cage. The node ID number of the module is its slot position. The slots are numbered 1 through E (hexadecimal) from right to left, as you view the card cage from the front of the cabinet.

For information on XMI card cage configuration rules and module indicator lights, see the *VAX 6000 Platform Service Manual*. The *VAX 6000 Platform Technical User's Guide* gives in-depth information on the operation of the system bus.

2.4 I/O Connections

I/O connections are installed on the bulkhead connections tray and the I/O panel. The I/O tray is located in the rear of the cabinet, above the cooling system and below the power regulators, and covers the XMI backplane. The I/O panel is just below the right-hand side of the I/O tray and houses the Ethernet and console terminal ports.

Figure 2–7: Console and Terminal Connectors



The I/O bulkhead connections tray is located in the rear of the cabinet, above the cooling system and I/O panel, and below the power regulators. It is hinged at the bottom, and folds out and down for servicing the card cages and backplanes. The I/O panel is on the right side below the tray.

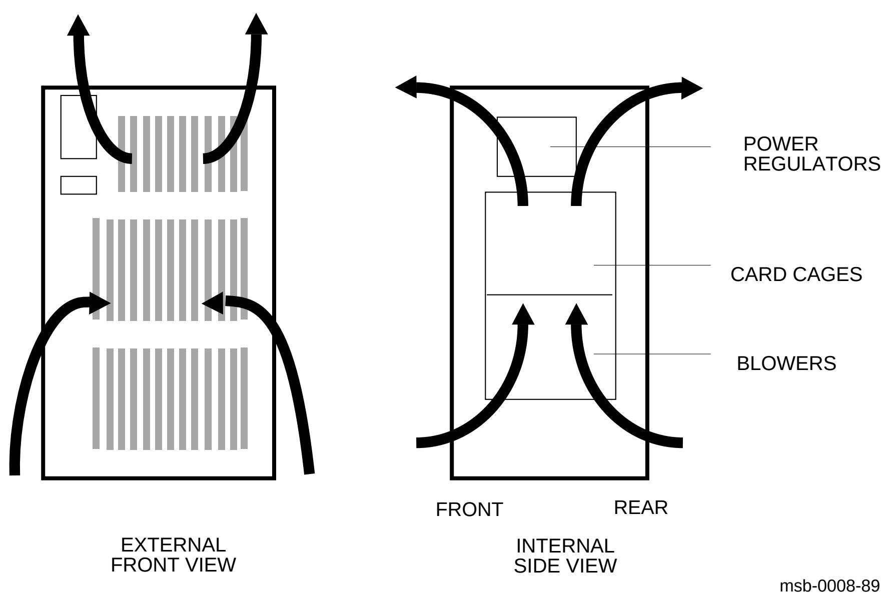
The I/O tray and panel have 30 panel units designed to accommodate a variety of I/O connectors.

The Ethernet and console terminal connectors are at the bottom of the I/O panel. The Ethernet port is a 15-pin receptacle located on the bottom right, and the console terminal port is the 25-pin receptacle on the left. These connectors are labeled with international symbols, as shown in Figure 2–7.

2.5 Cooling System

The cooling system consists of a fan, two blower units, and an airflow path through the XMI card cage (and VAXBI card cages, if present).

Figure 2–8: Airflow Pattern



The cooling system is designed to keep system components at an optimal operating temperature. It is important to keep the front and rear doors free of obstructions, leaving a clear space of 39.4 inches (1 meter) from the cabinet to maximize air intake.

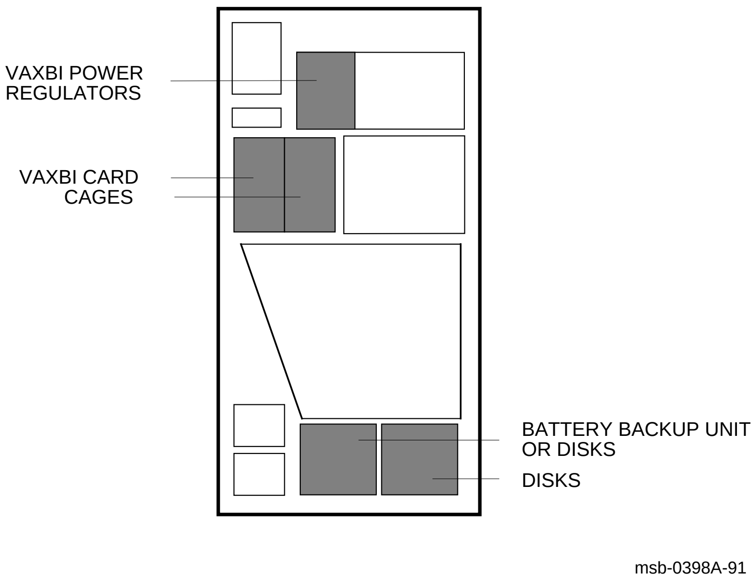
The blowers, located in the lower half of the cabinet, draw air in through the doors and push air up through the card cages. The air is directed through a duct to cool the console load device if there are no VAXBI card cages in the system. The airflow continues through the top of the card cages, through the power regulators, and out the top of the front and rear doors. A fan cools the power and logic box.

The system has safety detectors for the cooling system: an airflow sensor and a thermostat are installed above the power regulators in the top of the cabinet. Extreme conditions activate these detectors. Under extreme temperatures, the thermostat shuts off all output power (including power at the two unswitched outlets) at the AC power controller. In this condition the battery backup unit is disabled and will not provide power. If the airflow to your system is seriously blocked for an extended period of time, the airflow sensor shuts off the power supply. If either condition occurs, call your Digital customer service engineer.

2.6 Options

Other system options in addition to the console load device include the VAXBI card cages and power regulators, battery backup unit, and in-cabinet disks.

Figure 2–9: System Options



Options include the VAXBI I/O interface, battery backup unit, and in-cabinet disks. The VAXBI card cages are located in the upper third of the cabinet on the left side, as viewed from the front of the cabinet. The disks and battery backup unit are located beneath the blowers and are rack-mounted. The first of these two options to be installed is placed adjacent to the AC power controller.

VAXBI Card Cages and Power Regulators

The optional VAXBI I/O interface is a one-channel bus housed in two 6-slot VAXBI card cages. Two power regulators supply power to the VAXBI backplane. A clear latched door protects the components housed in the VAXBI card cages and helps to direct the airflow over the modules. Indicator lights on the modules can be viewed through this clear front door. The protective door must be closed for normal operation, as it controls the power supply to the card cages.

Additional VAXBI card cages can be added to a system by installing a VAXBI expander cabinet (see Appendix D). See the *VAX 6000 Platform Service Manual* for more information on the VAXBI card cages.

Battery Backup Unit

The battery backup unit supplies power to sustain the system for up to 10 minutes following a power interruption. Ride-through capability for up to 1 second is provided. The system control panel indicates the status of the battery backup unit.

Disks

The system cabinet provides space for two SA7x building blocks (8 RA7x drives), two RA92 disk drives, or two SF7x arrays (8 RF7x drives), or some combination. The disk control panel is accessed through an opening in the cabinet front door.

NOTE: *Installation of a battery backup unit allows only one disk enclosure to be installed in the system cabinet.*

Chapter 3

Controls and Indicators

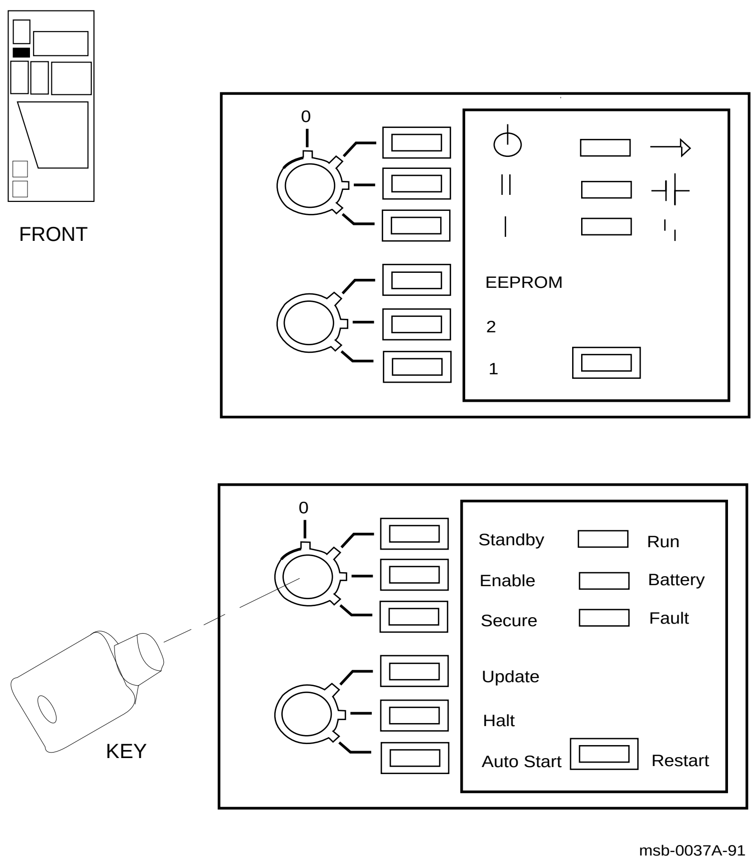
This chapter introduces system controls and indicators. Sections include:

- Control panel
- Upper key switch
- Lower key switch
- Restart button
- Status indicator lights
- Circuit breaker

3.1 Control Panel

The control panel, at the upper left of the cabinet front, contains the upper and lower key switches, status lights, and a Restart button. The upper and lower switches are operated by a key.

Figure 3–1: International and English Control Panels



The control panel is at the upper left of the cabinet. You use the control panel when powering on the machine or changing the operating mode of your system.

The upper and lower switches are operated by a key. Two keys are shipped with each system. The key has a toothed hollow barrel and fits into the slotted circle of each switch. Each key works on both switches.

Labels for the control panel's upper and lower key switches can be in English or in international symbols. Table 3–1 gives the relationship between the international symbols and English equivalents. References to the control panel in the remainder of this manual refer to the English labels.

Table 3–1: Control Panel Symbols

Location	English	International Symbol
Upper key switch	O	O (Off)
	Standby	⏻
	Enable	||
	Secure	|
Lower key switch	Update	EEPROM
	Halt	2
	Auto Start	1
Status indicators	Run	⟶
	Battery	⊣⊢
	Fault	
Restart button	Restart	(None, blank)

msb-0575A-91

3.2 Upper Key Switch

The control panel's upper key switch regulates power going into the system and determines use of the console terminal. The four switch positions are Off, Standby, Enable, and Secure.

Figure 3–2: Upper Key Switch (Enable Position)

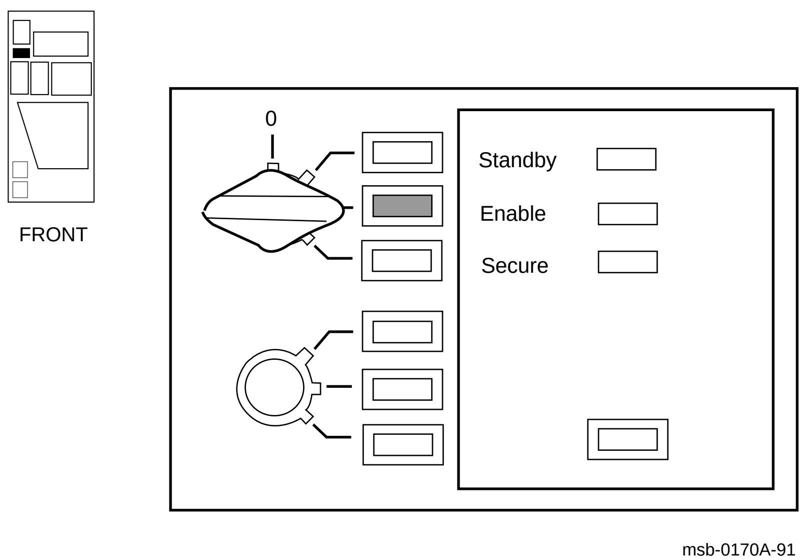


Table 3–2: Upper Key Switch

Position	Effect	Light Color
O (Off)	Removes all power, except to the battery backup charger and optional storage.	No light
Standby	Supplies power to XMI backplane, blowers, and in-cabinet console load device.	Red
Enable	Supplies power to whole system; console terminal is enabled. Used for console mode or restart, and to start self-test.	Yellow
Secure (Normal Position)	Prevents entry to console mode; position used while machine is executing programs. Disables Restart button. When switch at Secure, the system performs an automatic restart, regardless of the setting of the lower key switch.	Green

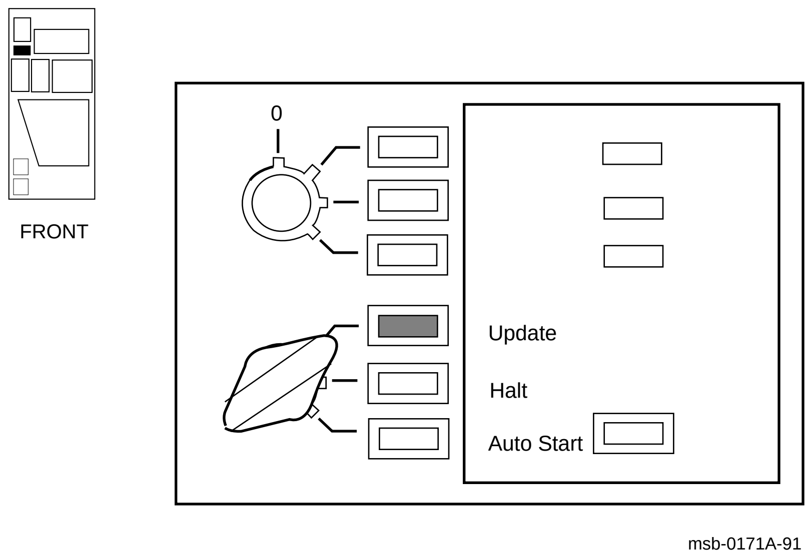
The upper key switch has four positions: Off, Standby, Enable, and Secure. You change the position of the upper key switch by inserting and turning a key. A light to the right of each key position lights to show what mode is in operation. When the switch is set to Off, no lights are lit. Each position modifies power to the system as follows:

- **Off** removes all power from the system, disabling the battery backup unit's output. This position is a total off, except for power to the battery charger and optional storage. To ensure total absence of power in the machine, pull the circuit breaker and unplug the machine. See Section 3.6.
- **Standby** powers the XMI backplane, blowers, and in-cabinet console load device.
- **Enable** supplies power to the entire system. While the upper switch is in the Enable position, you can use the console (see Chapter 5) or the Restart button (see Section 3.4). Also, when you move the upper switch from Standby to Enable, the system runs self-test. If the power goes off with the switch in the Enable position, the operation of the system is controlled by the position of the lower key switch. Figure 3–2 shows the upper key switch with the key in the Enable position and the Enable light lit.
- **Secure** prevents interruptions to program execution. When the switch is in the Secure position, the console terminal can only be used as a user terminal (in program mode). A CTRL/P does not cause a switch to console mode. Secure also disables the Restart button. If the power goes off with the switch in the Secure position, the system may reboot if it has a battery backup unit or if power is restored.

3.3 Lower Key Switch

The control panel's lower key switch controls system operation. The three positions for this switch are Update, Halt, and Auto Start.

Figure 3–3: Lower Key Switch (Update Position)



When the upper key switch is in the Secure position, the lower key switch has the effect of Auto Start, regardless of its setting. (See Section 3.2.)

Table 3–3: Lower Key Switch

Position	Effect	Light Color
Update	Enables writing to CPUs and adapters. Halts boot processor in console mode on power-up or when Restart button is pressed. Used for updating parameters stored in EEPROMs (upper key switch must be set to Enable). Prevents an auto restart.	Red
Halt	Prevents an auto restart if a failure or transient power outage occurs.	Yellow
Auto Start (Normal Position)	Allows restart or reboot. Used for normal operation of the system.	Green

The lower key switch has three positions: Update, Halt, and Auto Start. A light to the right of each key position lights to indicate which mode is engaged.

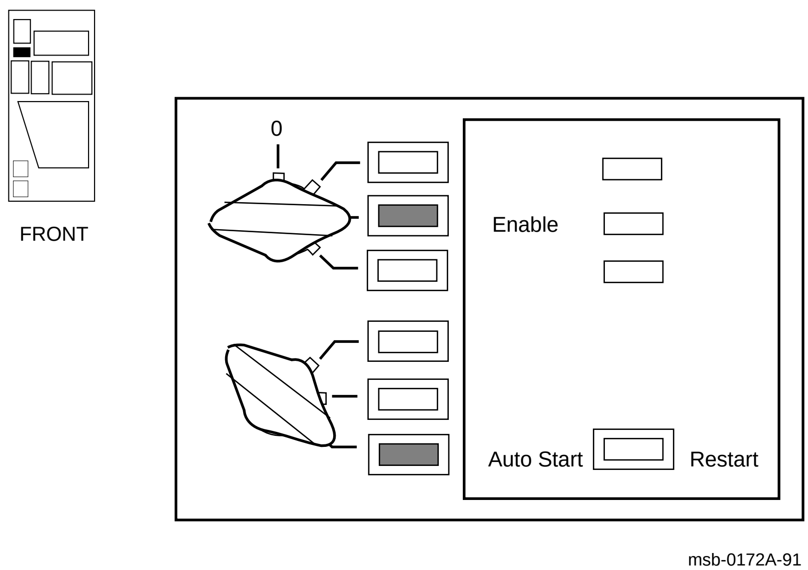
Each position engages the primary processor in a different way.

- **Update** readies the CPU for parameter changes to the EEPROM which are entered from the console terminal. You must have the switch in the Update position to use some console commands: UPDATE and most SET commands (see Section 5.18.1 through Section 5.18.5). (The upper key switch must be set to Enable.) Digital customer service engineers and self-maintenance customers use Update for updates to the console, self-test, and ROM diagnostics programs. See also Section 5.24. When the key is in the Update position, an automatic restart is inhibited. Figure 3–3 shows the lower key switch with the key in the Update position and the Update light lit.
- **Halt** inhibits automatic restart when a failure or transient power outage happens. It is the opposite of Auto Start. On power-up, the system halts in console mode, and you can issue a BOOT command (see Section 5.6).
- **Auto Start** is the key position for normal operation. The key must be in this position for automatic rebooting or restarting the system following a power failure.

3.4 Restart Button

The Restart button begins self-test, reboot, or both, depending on the position of the upper and lower key switches.

Figure 3–4: Restart Button



The upper key switch controls the effect of the Restart button. When the upper key switch is in the Enable position, the Restart button is operative. If the upper key switch is not in the Enable position, the Restart button is ignored.

Table 3–4: Restart Button

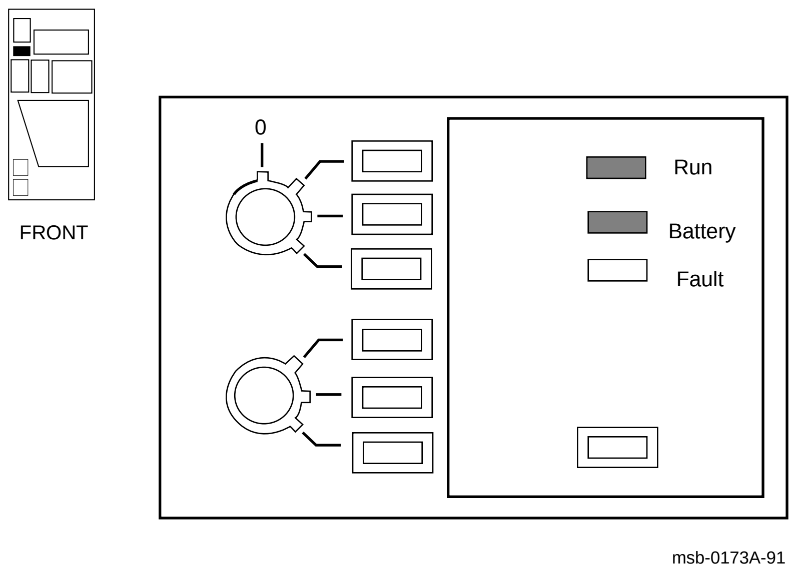
Upper Key Switch	Lower Key Switch	Restart Button Function
Enable	Update or Halt	Runs self-test, then halts.
Enable	Auto Start	Runs self-test and attempts a reboot. If the reboot fails, control returns to the console.
Standby or Secure	Any position	Does not function.

When you press the Restart button, the system runs self-test. For the Restart button to reboot the operating system, the upper key switch must be set to Enable and the lower key switch must be set to Auto Start. Figure 3–4 shows the control panel with upper and lower key switches in position for using the Restart button to reboot. If the system fails self-test, the processor does not reboot the operating system.

3.5 Status Indicator Lights

The control panel has three status indicator lights: Run, Battery, and Fault. These lights indicate the operating status of the system.

Figure 3–5: Control Panel Status Indicator Lights



Three status indicator lights on the control panel show the state of system execution (*Run*), the presence of a battery backup unit (*Battery*), and hardware errors (*Fault*).

Figure 3–5 shows a system that is in operation, with a fully charged battery backup unit installed. Table 3–5 describes the conditions indicated by the status indicator lights.

Table 3–5: Control Panel Status Indicator Lights

Light	Color	State	Meaning
Run	Green	On	System is executing operating system instructions on at least one processor.
		Off	System is in console mode, is set to Standby, or is turned off.
Battery	Green	On	Battery backup unit is charged to 98% of full capacity or BBU is supplying power to the load.
		Flashing 1 x/sec	Battery backup unit is charging.
		Flashing 10 x/sec	Battery backup unit requires service.
		Off	System does not have a battery backup unit.
Fault	Red	On	Self-test is in progress. If light does not turn off, system has a hardware fault. See Chapter 6 for self-test information.
		Off	Self-test has completed, or the system is turned off.

3.6 Circuit Breaker

The circuit breaker is on the AC power controller, which is at the bottom right corner at the back of the cabinet.

Figure 3–6: Circuit Breaker and the AC Power Controller

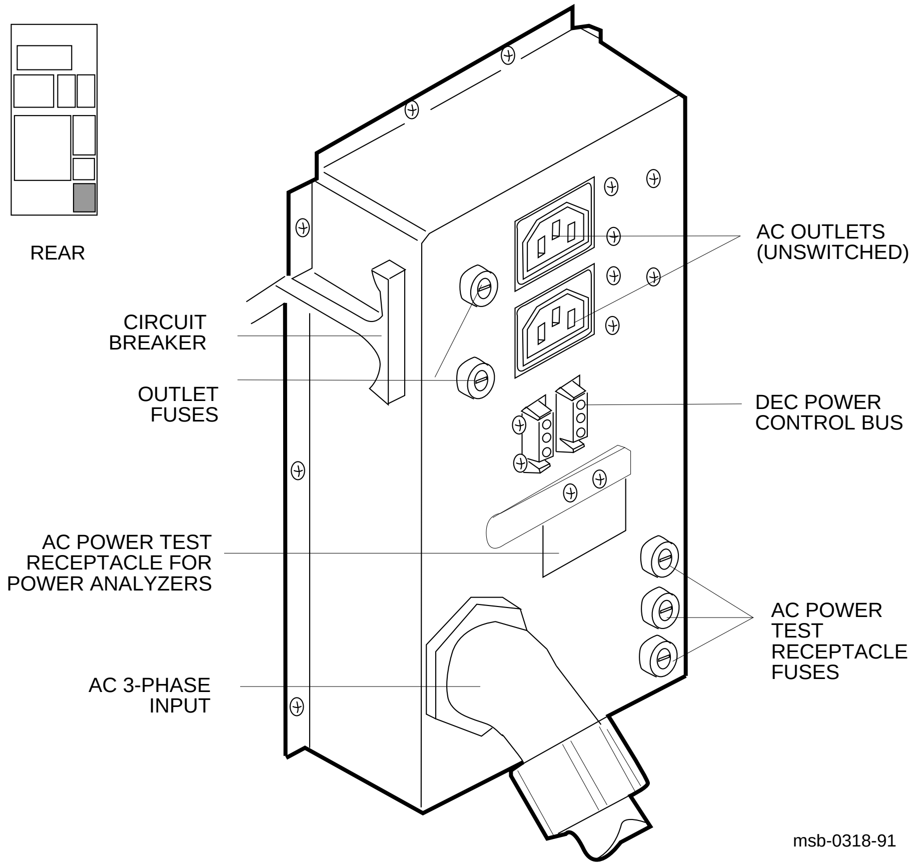


Figure 3–6 shows the AC power controller, which is at the rear of the cabinet.

Circuit Breaker

The circuit breaker controls power to the entire system, including the power regulators, blowers, and in-cabinet options. Current overload causes the circuit breaker to move automatically to the Off position, so that power to the system is turned off.

For normal operation, the circuit breaker must be in the On position, which is fully pressed in. To trip the circuit breaker, pull it out toward you, away from the machine, until the circuit breaker handle is flush with the AC power controller.

If the temperature of the system exceeds 75^{o}C (167^{o}F), the "contactor" in the AC power controller is opened and the system powers down.

AC Power Test Receptacle Fuses

These three fuses are used as a protective measure during power testing at the AC power test receptacle.

AC Outlets (Unswitched)

The two unswitched outlets on the AC power controller are used to power in-cabinet disk drives or a battery backup unit. The outlet fuses are to the left of the outlets.

DEC Power Control Bus

Two DEC power control bus connectors are located on the AC power controller. The power control bus provides central power-up and power-down capability by connecting the power controllers in your system, VAXBI expander, and storage cabinets.

AC Power Test Receptacle

This covered receptacle is used by Digital customer service engineers to connect test equipment that monitors AC power.

Chapter 4

Booting

This chapter describes how to boot the system. Sections include:

- How booting works
- Boot devices
- Regular boot procedure
- Boot device selection
- Boot processor selection
- Booting from an HSC disk
  - VAXcluster boot overview
  - Sample VAXcluster boot
- Booting from an Ethernet-based compact disk server
  - CD server boot command
  - Selecting an Ethernet service
- Ethernet boot overview
- Sample target-initiated Ethernet boot
  - Step 1, Gather information at target node
  - Step 2, Enter information into executor's NCP volatile database
  - Step 3, Boot from the target node

4.1 How Booting Works

The boot program reads the virtual memory boot program (VMB) from the boot device. VMB in turn boots the operating system.

Figure 4–1: Boot Procedure

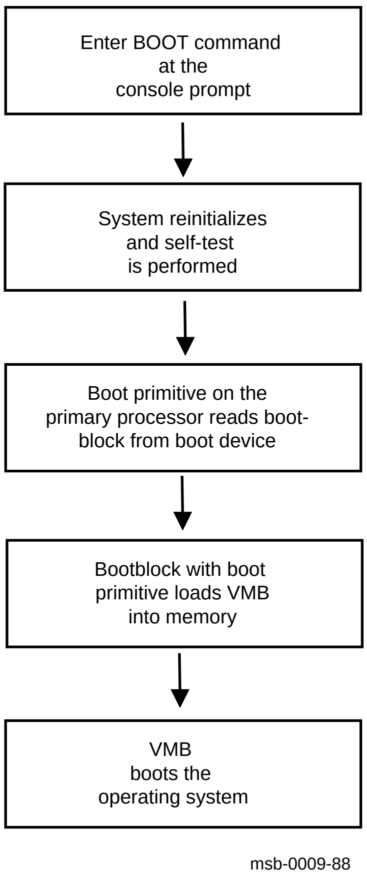


Table 4–1: Boot Procedure

Step	Procedure
1	You enter BOOT command from the console terminal in console mode. The BOOT command specifies the boot device and the path needed to reach it.
2	System reinitializes and performs self-test.
3	Boot primitive is invoked from console ROM on the boot processor. Boot primitive reads the bootblock from the specified boot device and transfers control to the bootblock.
4	The bootblock contains code and a pointer to VMB. The bootblock loads VMB into the first 256-Kbyte block of available memory.
5	Once VMB is loaded into memory, the bootblock transfers control to VMB, which in turn starts the operating system.

Boot primitive

Each boot device has a small program called a boot primitive that is stored in ROM on each processor with the console program. The boot primitive reads the bootblock from its boot device. How to load boot primitives is explained in Appendix E.

Boot device

The boot device contains the bootblock and typically also contains VMB. The system can be booted from one of four boot devices: the system console load device, a local system disk connected through a KDM70 or a KFMSA, a disk connected to the system through a CI adapter (CIXCD), or a disk connected to the system through an Ethernet adapter. See Appendix D for a list of boot devices connected to the system by VAXBI adapters.

Bootblock

The bootblock is logical 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.

VMB

The virtual memory boot program (VMB.EXE) 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.

4.2 Boot Devices

The system can be booted from one of four boot devices: the system console load device, a local system disk, a disk connected to the system through a CIXCD adapter, or by Ethernet from a remote disk on another system.

Figure 4–2: Boot Devices

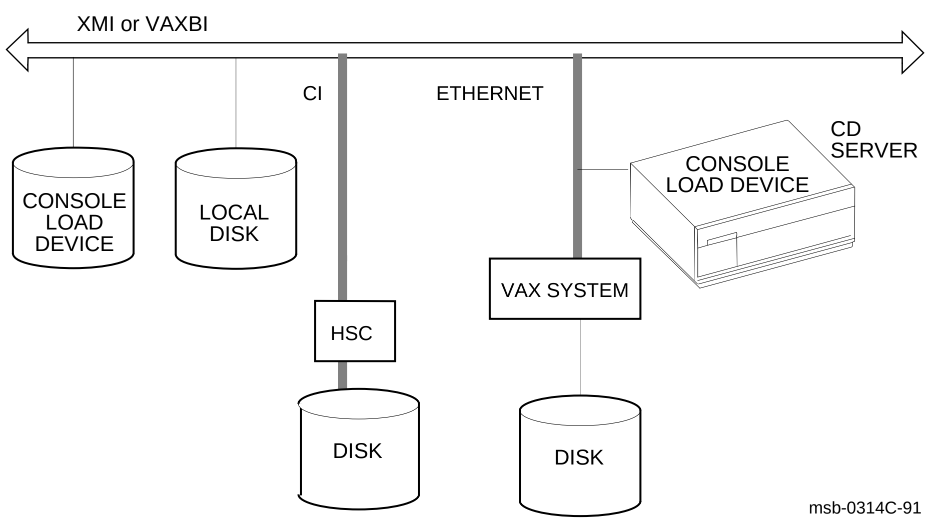


Table 4–2: Boot Devices

Device	Location
Console load device	Tape drive in the upper left corner of the system cabinet, a TF857 tape drive in an SFxxx cabinet, or an Ethernet-based CD server. All are used for booting standalone backup or diagnostics. See Section 2.1.
Local disk	Disk connected to the system through a KDM70 or KFMSA adapter. Regular boot procedure specifies such a disk as default boot device for individual systems that are not VAXclustered or networked.
CI disk	Disk located on system's HSC controller connected to the system by the CIXCD adapter on the XMI.
Ethernet disk	Disk connected to another system on the Ethernet, through the DEMNA Ethernet port interface or the FDDI DEMFA adapter.

You have a choice of five boot device *locations*. They are:

- A local console load device
- A local disk
- A remote console load device
- A remote disk on an HSC controller
- An Ethernet disk on another system

The console load device can be used to boot the standalone backup program and the VAX Diagnostic Supervisor.

Typically, you designate the local boot device as the default boot device after the initial boot of the system in installation. You can also store the BOOT specification parameters by giving these parameters a nickname. See the BOOT command in Chapter 5 for more information on storing boot device parameters.

Boot devices can also be on the optional VAXBI. See Appendix D for boot devices supported by the VAXBI.

4.3 Regular Boot Procedure

With the system in console mode, you issue a BOOT command. You can give a complete specification in which the qualifiers determine the boot device, or you can use a nickname.

Figure 4–3: Regular Boot Procedure

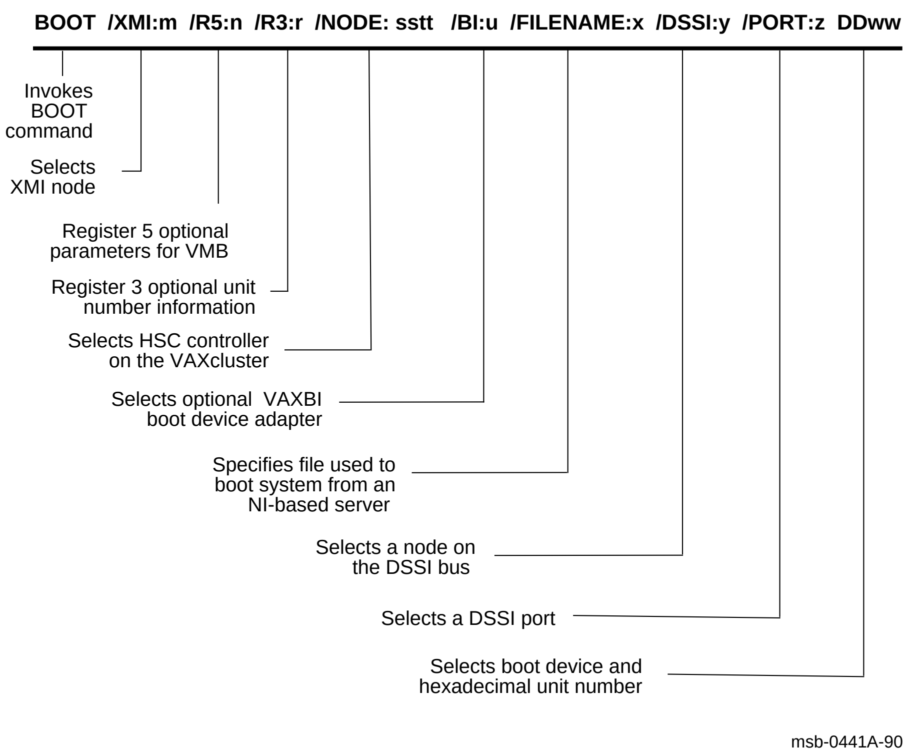


Figure 4–3 shows the components of the BOOT command. The /XMI node number you enter corresponds to an adapter; if you have an optional VAXBI, the /XMI node number will correspond to a DWMBB adapter and you must then use the /BI qualifier to specify the node number of the boot device on that VAXBI.

When using the /R5:n qualifier, see Appendix F for the values of *n*. The /R3:r qualifier is used with VMS when you boot from a shadow set, where *r* is two unit numbers. The first unit number is the functional unit number of the shadow set, and the second unit number is the physical unit number of one of the disks in the shadow set. Refer to your operating system manual for more details.

To designate an HSC disk as the boot device, you use the /XMI and /NODE (and /BI, if the CI adapter is a VAXBI device) qualifiers with their respective node numbers. The /XMI node number must be an adapter, the /NODE node number carries the CI node number of one or two HSC controllers, and DD*ww*, where DD specifies the device type as a disk and *ww* is the hexadecimal unit of the disk boot device.

To designate a DSSI disk as the boot device, you use the /XMI and /DSSI_NODE qualifiers with their respective node numbers. The /XMI node number is a KFMSA adapter, the /DSSI_NODE number is the adapter that provides access to the boot device. The /PORT qualifier specifies the DSSI bus 1 or 2 on the KFSMA adapter and, DD*ww*, where DD specifies the device type and *ww* is the hexadecimal unit of the boot device.

To designate an Ethernet-based CD server as the boot device, use the /XMI qualifier to specify the node number of the DEMNA or DEMFA adapter, and, /FILENAME:x, where *x* specifies the 1- to 16-character file name that the CD server loads during the boot process.

For convenience, you can store a BOOT command under a nickname, using the SET BOOT command. Any four characters can be used for a nickname. However, to avoid confusion, use nicknames that are different from device specifications. Also, note that the system reserves the name DEFAULT to specify a special saved boot specification, which is called when you enter the BOOT command without a nickname, so DEFA should not be used as a nickname. The default boot specification is also used when the control panel is set to Auto Start.

You can store up to 10 saved boot specifications, in addition to the default specification. Table 4–4 gives the specific mnemonics for each device type. See Section 5.18.1 for details on creating nicknames for boot devices. See Section 5.6 for more information on the BOOT command.

4.4 Boot Device Selection

You can boot the operating system in a number of ways. Table 4–3 lists some examples.

Table 4–3: Sample BOOT Commands

Boot Procedure	BOOT Command	Refer to
Boot from in-cabinet console load device	BOOT CSA1	
Boot VAX/DS from an in-cabinet console load device	BOOT /R5:10 CSA1	
Boot from local RA disk	BOOT /XMI:m DUww	Section 5.6
Boot from local RF disk	BOOT /XMI:m /DSSI_NODE:y /PORT:z DIww	Section 5.6
Boot from HSC disk	BOOT /XMI:m /R5:v/NODE:sstt DUww	Section 4.6 Appendix F
Boot from a DSSI TF tape	BOOT /XMI:m /DSSI_NODE:y /PORT:z MIww	Section 5.6
Boot from an Ethernet-based CD server	BOOT /XMI:m /FILENAME:ISL_LVAX_n EX0	Section 4.7
Boot over FDDI from a CD server	BOOT /XMI:m /FILENAME:ISL_LVAX_n FX0	Section 4.7
Boot VAX/DS from an Ethernet-based CD server	BOOT /XMI:m/FILENAME:ISL_LVAX_x[1] /R5:10 EX0 Section 4.7	
Boot over the Ethernet from a VAXBI device	BOOT /XMI:m /BI:x ET0	Section 4.8
Boot VAX/DS from disk	BOOT /XMI:m /R5:10 DUww	Appendix F
Conversational boot	BOOT /XMI:m /R5:1 DUww	Appendix F
Boot from VMS shadow set	BOOT /XMI:m /R3:w /NODE:sstt DUww	Section 5.6

[1]Where *x* is a letter that indicates the version.

You can issue a complete boot specification, or you can use a nickname that has been defined for a complete boot specification. Issuing BOOT alone will boot from whatever has been set as the default boot.

When you use the BOOT CSA1 command, you designate the in-cabinet TF or TK tape drive as your boot device.

Boot device mnemonics are listed in Table 4–4.

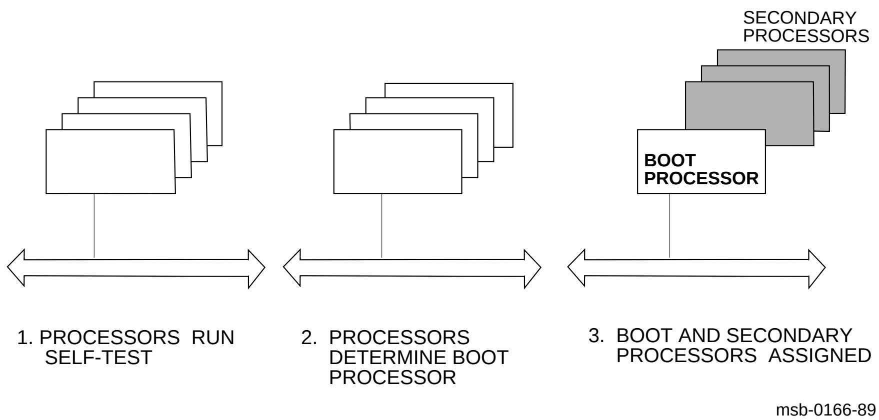
Table 4–4: Boot Device Mnemonics

Mnemonic	Device	Device Type	Bus
DIxx	KFMSA	Disk	XMI
DUxx	KDB50	Disk	VAXBI
	KDM70	Disk	XMI
	CIBCA	Disk	VAXBI
	CIXCD	Disk	XMI
ETxx	DEBNI	Disk	VAXBI
	DEBNA	Disk	VAXBI
EXxx	DEMNA	Disk	XMI
FXxx	DEMFA	Disk	XMI
MIxx	KFMSA	Tape	XMI
MUxx	TBK70	Tape	VAXBI

4.5 Boot Processor Selection

One processor is selected as the boot processor, and all other processors become secondary processors. This determination is made by the system at power-up or initialization, and can be altered by using console commands.

Figure 4–4: Determining the Boot Processor



Each processor has an image of the console program and boot code in ROM, but there is only one console terminal and a single system control panel. One processor is designated as the boot processor (or primary processor) and becomes the primary communicator to the console program. The signals from the console terminal and system control panel are bused on the XMI and are driven by the boot processor.

At power-up or initialization of the system, the console program in each processor begins parallel execution. Each processor performs self-test and then checks with the other processors to determine which processor becomes the boot processor. The boot processor is the processor with the lowest node ID number, passing self-test, that is eligible to become the boot processor (see Section 5.18.2).

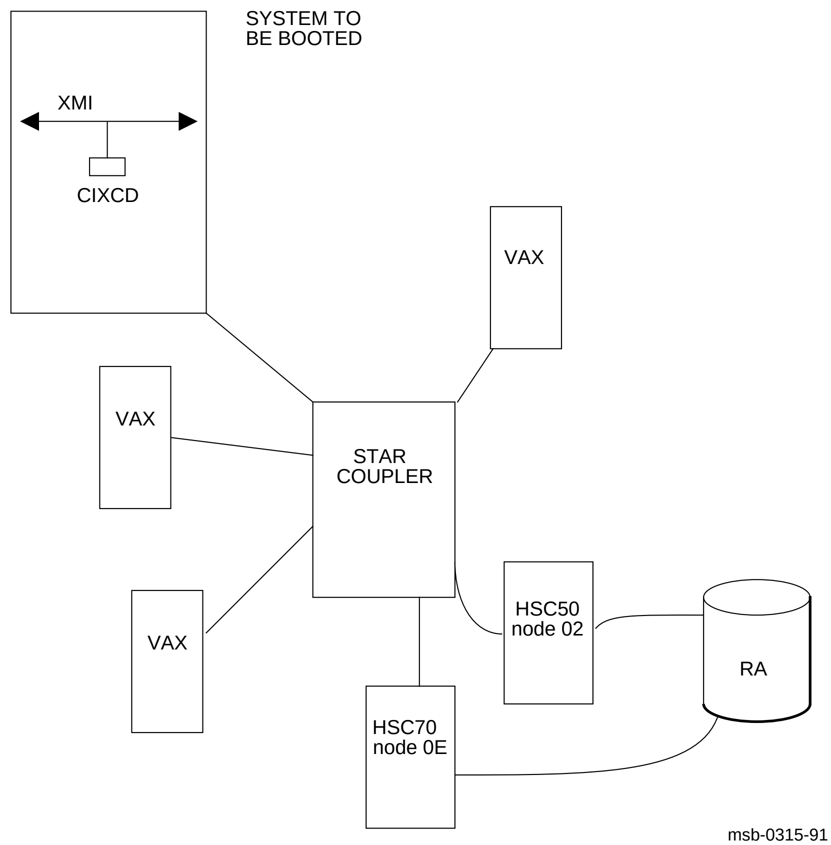
Once the boot processor has been determined, all other processors on the system become secondary processors. The console programs in secondary processors wait for commands from the boot processor.

4.6 Booting from an HSC Disk

4.6.1 VAXcluster Boot Overview

This section describes booting VMS from a VAXcluster. ULTRIX is booted in the same way, except that R5 bits must be specified (see Appendix F).

Figure 4–5: Booting from a CI-Based VAXcluster



When you boot VMS from a VAXcluster, you need to gather the following information:

- Node number of the HSC controller(s)
- Device address of the disk unit that will execute boot
- Location of the system root

The node number of an HSC controller is a 2-digit hexadecimal number. The device type is of the form DU0 (see Section 5.6). The location of the system root is a hexadecimal number that indicates the system to be booted.

Figure 4–5 shows a sample VAXcluster configuration. Note that there are two HSC controllers with node numbers 02 and 0E in this configuration. Section 4.6.2 discusses a CI boot on the system configuration shown in Figure 4–5.

If your system is part of a VAXcluster and you want it to boot from the cluster system disk, you may set its default boot specification to this disk using the SET BOOT console command (see Section 5.18.1).

4.6.2 Sample VAXcluster Boot

This section shows a sample boot from a system to be booted in the VAXcluster configuration shown in Figure 4–5.

Example 4–1: Sample VAXcluster Boot

```
>>> SHOW CONFIGURATION ❶       ! Enter command.
     Type          Rev         !
 1+  KA65A   (8080) 0006       ! Find the XMI address of
 2+  KA65A   (8080) 0006       ! the CIXCD, which is the
 3+  KA65A   (8080) 0006       ! VAXcluster interface.
 6+  MS65A   (4001) 0084       ! The CIXCD is XMI D.
 7+  MS65A   (4001) 0084
 8+  MS65A   (4001) 0084
 9+  MS65A   (4001) 0084
 D+  CIXCD   (0C05) 1652 ❶
 E+  DEMNA   (0C03) 0600

>>> BOOT /XMI:D❶ /R5:70000000❷ /NODE:0E02❸ DUC❹

Initializing system

#123456789 0123456789 0123456789 0123456789 012345#  ❺

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
    +   +   .   .   .   +   +   +   +   .   .   +   +   +       STF
    .   .   .   .   .   .   .   .   .   .   .   E   E   B       BPD
    .   .   .   .   .   .   .   .   .   .   .   +   +   +       ETF
    .   .   .   .   .   .   .   .   .   .   .   E   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  SN = SG01234567

Loading system software

* Initializing adapter ❻
* Adapter initialized successfully
* Connecting to HSC
* Connecting to MSCP server layer
* Connecting to boot disk
* Reading bootblock from disk
* Passing control to transfer address

   [operating system banner appears]
```

❶ SHOW CONFIGURATION displays the positions of modules. The first column identifies each module's node, slot location, and self-test status. The second column entries are the device names; the third, the device type codes. The last column shows the revision level of each module. The CIXCD is located at XMI node D. Enter this value D in the BOOT command as the argument to the /XMI qualifier.

❷ In the BOOT command, the system root is the argument to the /R5 qualifier. In Example 4–1 the system root for VMS SYS7 is specified. (To boot ULTRIX, the appropriate /R5 qualifier from Appendix F would be given.)

❸ The arguments to /NODE are hexadecimal HSC node numbers. In this example they are 0E (decimal 14) and 02 (decimal 2). Listing two arguments tells the system to connect to either the HSC at VAXcluster node 14 or the HSC at VAXcluster node 2. The command takes a maximum of two parameters for this qualifier. If your disk is dual-ported to the HSC controller, be sure to use both nodes; this gives the system an alternate route in case one HSC is disabled.

❹ DUC requests booting from a disk with hexadecimal unit number C.

❺ System runs self-test (see Chapter 6 for a description of the self-test display).

❻ Status messages appear, indicating a successful boot.[1]

For more information on VAXcluster booting, see Appendix F and the VMS installation manual.

---

[1] These boot status messages apply only to Model 500 and 600 systems. See Appendix J.

4.7 Booting from an Ethernet-Based Compact Disk Server

4.7.1 CD Server Boot Command

This section shows a sample boot on a Model 500 system from an Ethernet-based compact disk (CD) server. The first step is issuing the boot command.

Example 4–2: Sample Ethernet-Based CD Server Boot

```
>>> SHOW CONFIGURATION ❶        ! Enter command.
      Type          Rev         !
  1+  KA65A   (8080) 0006       ! Find the XMI address of the
  2+  KA65A   (8080) 0006       ! DEMNA, which is the Ethernet
  3+  KA65A   (8080) 0006       ! interface.  Here, the DEMNA
  6+  MS65A   (4001) 0084       ! connects to the system bus at
  7+  MS65A   (4001) 0084       ! XMI node E. ❶
  8+  MS65A   (4001) 0084
  9+  MS65A   (4001) 0084
  B+  KDM70   (0C22) 0002
  D+  CIXCD   (0C05) 1652
  E+  DEMNA   (0C03) 0600 ❶

>>> BOOT /XMI:E ❶ /FILENAME:ISL_LVAX_A ❷ EX0 ❸

Initializing system

#123456789 0123456789 0123456789 0123456789 012345# ❹

F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0   NODE #

    A   A   .   A   .   M   M   M   M   .   .   P   P   P       TYP
    +   +   .   +   .   +   +   +   +   .   .   +   +   +       STF
    .   .   .   .   .   .   .   .   .   .   .   E   E   B       BPD
    .   .   .   .   .   .   .   .   .   .   .   +   +   +       ETF
    .   .   .   .   .   .   .   .   .   .   .   E   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  SN = SG01234567

Loading system software
```

Example 4–2 Cont'd on next page

Example 4–2 (Cont.): Sample Ethernet-Based CD Server Boot

```
* Initializing adapter ❺
* Specified boot adapter initialized successfully
* "Request Program" MOP message sent - waiting for service from remote node
* Remote service link established
* Reading boot image from remote node
* Passing control to transfer address
```

❶ SHOW CONFIGURATION displays the positions of the modules. The DEMNA is located at XMI node E. Enter this value E in the BOOT command as the argument to the /XMI qualifier.

❷ In the boot command, the /FILENAME qualifier is used to request a file that will be loaded into system memory from the remote CD server. In this example the file ISL_LVAX_A is the argument to the /FILENAME qualifier.

❸ EX0 specifies the DEMNA adapter as the boot device.

❹ System runs self-test (see Chapter 6 for a description of the self-test display).

❺ Boot status messages are displayed (see Appendix J for the boot status and error messages displayed by Model 500 and 600 systems).

4.7.2 Selecting an Ethernet Service

The second step of booting over the Ethernet with a CD server is selecting the service that boots VMS.

Example 4–3: Selecting an Ethernet Service

```
 Ethernet Initial System Load Function ❶

  FUNCTION        FUNCTION
    ID
    1     -       Display Menu
    2     -       Help
    3     -       Choose Service
    4     -       Stop

 Enter a function Id value: 3 ❷

 Service options:
               1 = Find Services
               2 = Enter Known Service Name
               =>1 ❸

 Working

 Servers found: 1

 Service Name Format:
        Service Name
        Server Name
        Ethernet ID

#1
VMS054
ESS_08002B150589
08-00-2B-15-05-89

#2
CD_BIN_83371
ESS_08002B150589
08-00-2B-15-05-89

 Enter a number =>1 ❹

        [operating system banner appears] ❺
```

❶ The Ethernet Initial System Load Function menu is displayed.

❷ The system prompts you for a function ID value. Enter 3 to select a service.

❸ The Service options menu is displayed. Enter 1 to display the available Ethernet servers and services. In this example one server, ESS_08002B150589, is found on the Ethernet. Next, the Service Name Format is displayed, followed by the services. Service #1, VMS054, is used to boot VMS.

❹ Enter 1 to select service #1.

❺ The operating system banner appears.

4.8 Ethernet Boot Overview

To boot VMS over the Ethernet, you use the Network Control Program (NCP). The system supports booting over the Ethernet, both trigger booting and booting initiated by the system as a target node.

Figure 4–6: Trigger Booting Using Ethernet

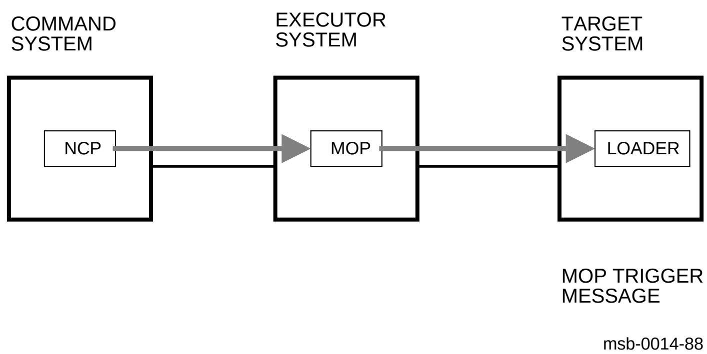
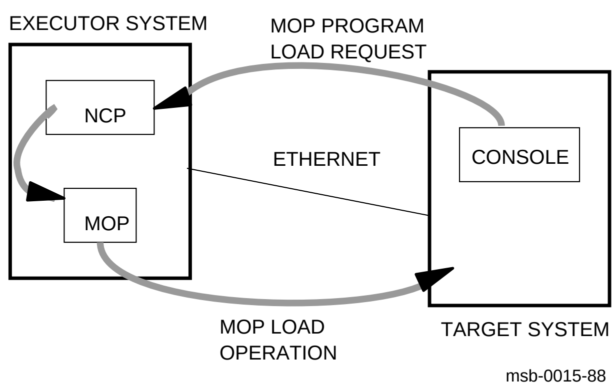


Figure 4–7: Target-Initiated Booting by Ethernet



The Ethernet is used to boot in two ways. Figure 4–6 and Figure 4–7 illustrate these methods.

A **trigger boot** initiates a BOOT command from a command system, which sends the command over the Ethernet to the executor system, which causes a boot in the target system (the VAX 6000 series system). The target system loads its boot program from the boot device that is designated as the default. (The default can be a local disk or the Ethernet.) The target system must have its control panel key switches at Auto Start and Enable. Commands are issued only from the command node, not from the target machine.

Target-initiated booting (Figure 4–7) is initiated by a console BOOT command from the target system. This BOOT command specifies an Ethernet controller as the boot device:

```
BOOT /XMI:m /R5:v EX0
```

(See Section 4.3 and Section 5.6 for more information on the BOOT command.)

Information must be entered in the executor system for the boot to succeed. Although the BOOT command is initiated at the VAX 6000 target node, the executor node Network Control Program's (NCP) volatile database requires an entry for the target node.

Section 4.9 shows an example of a target-initiated Ethernet boot.

4.9 Sample Target-Initiated Ethernet Boot

To perform a target-initiated boot over the Ethernet: (1) gather information at the target node, (2) enter the information into the Network Control Program volatile database on the executor node, and (3) issue a BOOT command from the target node. Example 4–4 through Example 4–6 show this procedure.

4.9.1 Step 1, Gather Information at Target Node

This section presents an example of booting over the Ethernet, in a target-initiated boot from a VAX 6000 series system.

Example 4–4: Step 1, SHOW Ethernet

```
>>> SHOW ETHERNET  ❶            ! Enter command on target machine.
Ethernet Adapters:
  XMI:E  08-00-2B-08-3D-64 ❷    ! The DEMNA is XMI node E. Its hardware
                                ! Ethernet address follows.
FDDI Adapters:
  XMI:D  08-00-26-1C-0D-B7      ! The DEMFA is XMI node D. Its FDDI
                                ! hardware address follows.
```

The first step in an Ethernet boot is to gather the information from your system that you need in the Ethernet boot.

❶ Use the SHOW ETHERNET command to find the address of your system on the Ethernet and write it down. You load this address into the executor system's volatile database in the next step (Section 4.9.2).

❷ The system reports the hardware Ethernet address for the DEMNA. The DEMNA is at XMI node E.

The system also reports the FDDI hardware address for the DEMFA adapter.

The system manager of the target system and the system manager of the executor system assign a node name and a network address for the target machine. In Example 4–4 the assigned node name is TARGET, and the node network address is 9.961.

4.9.2 Step 2, Enter Information into Executor's NCP Volatile Database

The second step of booting over an Ethernet is entering the target node information into the Network Control Program (NCP) volatile database on the executor system.

Example 4–5: Step 2, Entering Target Node Information

```
$ MCR NCP ❶                           ! On the executor system, at
                                      ! the DCL prompt, run NCP.
NCP>                                  ! NCP prompt appears.
                                      ! Enter information from the
                                      ! target node that you
                                      ! gathered in Step 1.

NCP> set node TARGET service circuit MNA-0 ❷
NCP> set node TARGET address 9.961 ❸
NCP> set node TARGET hardware address 08-00-2B-08-3D-64 ❹
NCP> set node TARGET tertiary loader sys$system:tertiary_vmb.exe ❺

NCP> show node TARGET char ❻          ! Check information by showing
                                      ! node TARGET's characteristics.
    Node Volatile Characteristics as of DD-MMM-YYYY 00:00:01

    Remote node  =  9.961  (TARGET)

    Service circuit         = MNA-0 ❼
    Hardware address        = 08-00-2B-08-3D-64
    Tertiary loader         = sys$system:tertiary_vmb.exe

NCP>                                 ! Prompt returns; show
NCP> sho circuit mna-0 char ❽        ! circuit characteristics.

    Circuit Volatile Characteristics as of DD-MMM-YYYY 00:00:01

    Circuit = MNA-0

    State                   = on
    Service                 = enabled ❽
    Designated router       = 9.739 (ABCDEF)
    Cost                    = 4
    Router priority         = 64
    Hello timer             = 15
    Type                    = Ethernet
    Adjacent node           = 9.739 (ABCDEF)
    Listen time             = 45
```

Example 4–5 Cont'd on next page

Example 4–5 (Cont.): Step 2, Entering Target Node Information

```
NCP>                                ! Prompt returns; show
NCP> sho line MNA-0 char ❾          ! line characteristics.

    Line Volatile Characteristics as of DD-MMM-YYYY 00:00:01

    Line = MNA-0

    Receive buffers         = 6
    Controller              = normal
    Protocol                = Ethernet
    Service timer           = 5000 ❾
    Hardware address        = 08-00-2B-06-01-00
    Device buffer size      = 1498
```

On the executor system, under VMS or the executor's operating system, run NCP, the Network Control Program, on an appropriate privileged account. Enter the information into the volatile database. The main piece of information required for booting is the hardware address. However, the database structure requires the rest of the information to qualify as a valid record entry.

❶ Run NCP.

❷ Assign the service circuit.

❸ Assign the node name (in this example, TARGET) and the network address (9.961).

❹ Enter the hardware address found in Step 1, Section 4.9.1.

❺ Enter the tertiary loader pathname. In this example the tertiary loader comes from a directory named SYS$SYSTEM. Check your operating system documentation for details.

❻ Check your work, using the SHOW NODE command.

❼ The service circuit code is dependent on your hardware.

❽ Check the circuit characteristics. Service must be enabled.[1]

❾ Check the line characteristics. The service timer is usually set to 5000 for an Ethernet boot.

---

[1] If you need to change the circuit characteristic to enable, you must turn the circuit off, set service to enable, and turn the circuit on again *quickly*, or all links could be lost to the system.

4.9.3 Step 3, Boot from the Target Node

The third step in Ethernet booting is to issue the BOOT command from the target node.

Example 4–6: Step 3, Booting from the Target Node

```
>>> BOOT /XMI:E EX0                 ! Enter BOOT command

Initializing system

#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
    +   +   .   .   .   +   +   +   +   .   .   +   +   +       STF
    .   .   .   .   .   .   .   .   .   .   .   E   E   B       BPD
    .   .   .   .   .   .   .   .   .   .   .   +   +   +       ETF
    .   .   .   .   .   .   .   .   .   .   .   E   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  SN = SG01234567

Loading system software

* Initializing adapter
* Specified adapter initialized successfully
* "Request Program" MOP message sent - waiting for service from remote node
* Remote service link established
* Reading boot image from remote node
* Passing control to transfer address

   [operating system banner appears]
```

Enter the BOOT command from the console terminal in console mode. Use the XMI node number that you found in Step 1, Section 4.9.1, describing the path of the Ethernet controller.[1]

Because the target VAX 6000 series system has been registered in the NCP volatile database of the executor system, the Ethernet boot completes. The executor system determines the location of the boot program.

[1] If your system has a VAXBI and a DEBNI or DEBNA Ethernet adapter, you must enter an additional /BI:n node qualifier.

Chapter 5
Console

The console program for the VAX 6000 series follows VAX console standards as described in Chapter 11, VAX Console Subsystems, of the *VAX Systems Hardware Handbook — VAXBI Systems*.

This chapter describes the console, its functions, and its language. Examples are given for each console command and its qualifiers.[1] Individual sections include:

- Description of console
- Console functions
- Console mode
- Console command language control characters
- Console command language syntax
- Console commands

BOOT	REPEAT	START
CLEAR EXCEPTION	RESTORE EEPROM	STOP
CONTINUE	SAVE EEPROM	TEST
DEPOSIT	SET BOOT	UNJAM
EXAMINE	SET CPU	UPDATE
FIND	SET LANGUAGE	Z
HALT	SET MEMORY	!
HELP	SET TERMINAL	
INITIALIZE	SHOW	

- Sample console session

[1] The RESTORE EEPROM and SAVE EEPROM commands apply only to systems having a TK tape drive. Qualifiers for a specific model (or models) are identified.

5.1 Description of Console

The console subsystem consists of a console terminal, console program located in ROM on the CPU modules, and dedicated memory. The console program runs on all processors and is automatically entered when the boot processor encounters a restart condition or when an operator invokes console mode on the console terminal.

Figure 5–1: Console

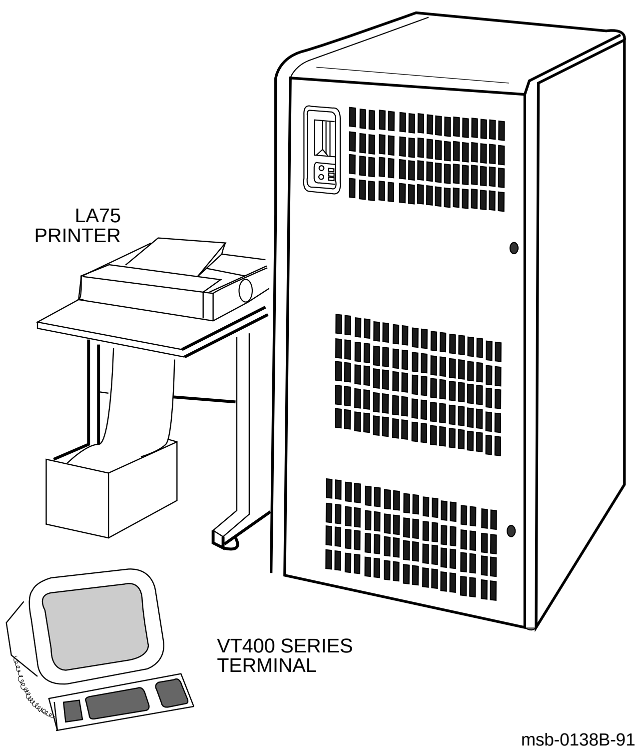


Table 5–1: Console Parts and Functions

Part	Function
Console terminal	Used for input, entering console commands.
Console printer	Provides a hardcopy record of console sessions.
Console terminal port	Connects the console terminal to the system.
Console program	Software interface; translates console commands to the processors; resides in ROM on each processor.
Dedicated memory	The console communications area (CCA) in main memory that allows the console programs on each processor to communicate with each other.

In multiprocessor systems, the console program runs on all processors, and the console program on the primary processor communicates with the console terminal (see Section 4.5). Each processor communicates with the others through a segment of shared main memory called the console communications area (CCA).

To use the console terminal in console mode, set the upper key switch on the control panel to the Enable position. The lower key switch can be at Update or Halt. The control panel is described in Chapter 3.

You designate a terminal as the console terminal by connecting it to the console terminal port. The console terminal port connection is on the I/O distribution panel above the AC power controller (see Section 2.4). The default console baud rate is 1200 when a system is installed. See the SET TERMINAL command for instructions on changing the defaults. The break key can also be used to change the baud rate (see Section 5.4).

When the system is in console mode, the terminal has exclusive use of the system.

The console prompt is:

```
>>>
```

5.2 Console Functions

Using the console program, you can examine and modify the system memory and registers, boot or restart an operating system, designate a primary processor, disable a vector processor, and return to program mode.

Table 5–2: Console Functions

Console Use	Commands Used
Bootstrap operating system	BOOT
Change console terminal parameters	SET TERMINAL, SAVE EEPROM, RESTORE EEPROM
Continue program mode	CONTINUE, START
Clear error state in some registers	CLEAR EXCEPTION
Deposit to or change memory interleave	FIND, DEPOSIT, SET MEMORY
Deposit or change registers	DEPOSIT
Designate primary processor or disable a vector processor	SET CPU
Display system parameters	SHOW
Examine memory	FIND, EXAMINE
Examine registers	EXAMINE
Execute ROM diagnostics	TEST
Exit from program mode	CTRL/P
Remove English from error messages	SET LANGUAGE
Run diagnostics	TEST, BOOT, INITIALIZE
Receive information on console commands	HELP
Set system parameters	SET
Start system	BOOT, INITIALIZE, START, CONTINUE
Stop system or device	STOP
Store boot specifications	SET BOOT

You use the console terminal to control the system manually, correct errors, determine the status of machine registers and counters, determine the contents of storage, and revise the contents of storage.

Following self-test or a SET CPU console command, one processor in a multiprocessor system is designated as the primary or boot processor. The location of the primary processor is determined at startup or when the system is reset. The primary processor performs a bootstrap or warm restart of the system. The operating system controls the other processors from the primary processor.

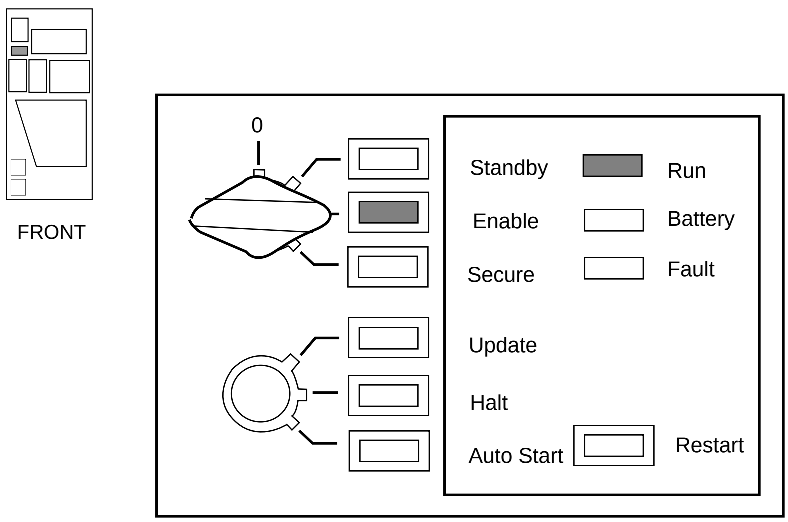
Nonprimary processors are called secondary processors. Secondary processors communicate with the console terminal through the primary processor when performing I/O during a console session. For information on how the primary processor is determined, see Section 4.5.

Appendix G gives a summary list of each console command and its functions.

5.3 Console Mode

To enter console mode from program mode, turn the upper key switch on the front control panel to the Enable position and type CTRL/P at the console terminal.

Figure 5–2: Console Switch When in Console Mode



The console terminal can operate in two modes: program mode and console mode. In program mode, the console terminal operates like a user terminal on the system and is under control of the operating system. In console mode, the system and the console terminal are operating under the console program.

When the console terminal operates in program mode, any input to the terminal is passed on to the operating system, as if the console were another terminal. When the console terminal operates in console mode, input is passed to the console program running on the primary processor.

To enter console mode, set the upper key switch to the Enable position, and type CTRL/P on the console terminal; or you can power up with

Halt selected. If you type CTRL/P when the upper key switch is in the Secure position, the CTRL/P is passed on to the operating system, and the operating mode does not change.

CTRL/P interrupts program mode on the boot processor. The secondary processors continue operating in program mode until they must wait for resources locked by the primary processor. Some I/O devices also require the attention of the primary processor. If a system remains in console mode for more than 30 seconds, various system timeouts could cause the system to hang when an attempt is made to return to program mode.

If you want to halt a secondary processor, you can issue a STOP command (see Section 5.21).

To resume program mode, use one of these commands:

CONTINUE	Resumes the program that was interrupted by the CTRL/P
START	Restarts the primary processor at a specified address

The console mode prompt is >>>. After entering a command, you may receive a system error message with number codes in the form:

```
?nnnn <message>
```

where *nnnn* is a number in hexadecimal format.[1] These codes indicate an error or a halted processor. See Appendix H for a listing of error codes for Model 400 and higher systems. Appendix I lists Model 300 error codes; Appendix J lists status and error codes for Model 500 and 600 systems.

When a secondary processor issues an error message, the primary processor is responsible for displaying the error on the console terminal. The primary processor displays these messages with a prefix indicating the node of the originating processor. For example, if a secondary processor at node 5 halted, the primary processor would display the error message:

```
Node 5:  ?0006 Halt instruction executed in kernel mode.
Node 5:  PC  = E00D26B4
Node 5:  PSL = 041F0600
Node 5:  ISP = 000002F0
```

[1] On Models 300 and 400 a system error message and number code will appear as:

```
?nn <message>
```

5.4 Console Command Language Control Characters

Eleven ASCII control characters have special meaning when you type them on the console terminal running in console mode. See Table 5–3.

Table 5–3: Console Control Characters

Character	Function
BREAK	Increments the console baud rate, if enabled.
CTRL/C	Causes the console to abort processing of a command.
CTRL/O	Causes the console to discard output to the console terminal until the next CTRL/O is entered.
CTRL/P	In console mode, acts like CTRL/C. In program mode, causes the boot processor to halt and begin running the console program.
CTRL/Q	Resumes console output that was suspended with CTRL/S.
CTRL/R	Redisplays the current line.
CTRL/S	Suspends console output on the console terminal until CTRL/Q is typed.
CTRL/U	Discards all characters on the current line.
DELETE	Deletes the previously typed character.
ESC	Suppresses any special meaning associated with a given character.
RETURN	Carriage return; ends a command line.

BREAK increments the console terminal baud rate to the next higher rate and displays a new console prompt. If you use BREAK at the highest baud rate, the program "wraps around" to the lowest rate. You can quickly synchronize the console baud rate to the console terminal if the default speeds do not match. To do this, hit BREAK repeatedly until the console prompt ">>> " appears. The baud rates are 300, 600, 1200, 2400, 4800, 9600, 19200, and 38400. (Model 600 does not support 38400.) It is not recommended to run faster than 1200 baud.

CTRL/C aborts processing of a command. Echoed as ^C, CTRL/C also resumes output which you suspended using CTRL/O. When you type CTRL/C as part of a command line, the line is deleted as if you entered CTRL/U.

CTRL/O stops output to the console terminal until you enter the next CTRL/O. CTRL/O is echoed as ^O followed by a carriage return and is not echoed when you reenable output. Output is also reenabled when the console prompts for a command, issues an error message, enters program mode, or when you type CTRL/P or CTRL/C.

CTRL/P works like CTRL/C and is echoed as ^P, if the console terminal is in console mode. If the console terminal is in program mode and is secured, CTRL/P is not echoed, but is passed to the operating system for processing. If the console is in program mode and is not secured, CTRL/P halts the processor and begins the console program; it also can terminate the Z command.

CTRL/Q resumes console output on the console terminal that you suspended with CTRL/S. The CTRL/Q key is not echoed.

CTRL/R is echoed as ^R, followed by a carriage return, line feed, and printing of the current command line, omitting deleted characters. This command is useful for hardcopy terminals.

CTRL/S suspends output to the console terminal until you type CTRL/Q. Any characters you enter after CTRL/S are buffered but not echoed until output is resumed. The CTRL/S input is not echoed.

CTRL/U discards all characters that you entered on the current line. It is echoed as ^U, followed by a carriage return, line feed, and a new console prompt.

DELETE deletes the previously typed character. If you define your console terminal as a hardcopy terminal (SET TERMINAL /HARD), a Delete is echoed with a backslash [\ ] followed by the character being deleted. If you delete several characters consecutively, the system echoes only the deleted characters, followed by another backslash at the end of the series. This displays the deleted characters surrounded by backslashes.

With a video console terminal, each Delete backs up the cursor and erases the previously displayed character.

ESC (escape) suppresses any special meaning associated with the character that immediately follows it. Control characters that would terminate a Z command are passed through to the target node. The character is echoed as "$". (On VT200 terminals and up, use CTRL/3 in place of the ESCAPE key.)

RETURN ends a command line. Any command entered before Return is received by the program.

5.5 Console Command Language Syntax

The console command language has syntax rules for forming commands. Commands contain up to 80 characters, can be abbreviated, and accept qualifiers. Tabs and spaces are compressed. Numbers are in hexadecimal notation.

Table 5–4: Console Command Language Syntax

Command Parameter	Attribute or Action
Length	80 characters maximum.
Abbreviation	Varies with the command; usually the shortest unique combination of letters.
Multiple adjacent spaces	Treated as a single space.
Multiple adjacent tabs	Treated as a single space.
Qualifier(s)	Can appear after the command keyword or after any symbol or number in the command; are preceded by a slash (/).
Numbers	Most appear in hexadecimal format.
No characters	Treated as a null command; no action taken.

NOTE: *Model 600 provides command recall; that is, pressing the Uparrow key recalls the previous command typed.*

The console program accepts commands up to 80 characters long. This does not include the terminating carriage return or any characters you delete as you enter the command. A command longer than 80 characters causes an error message of the form:

```
?0036 Command too long.
```

You can abbreviate commands and some qualifiers by dropping characters from the end of the word. You must enter the minimum number of characters to identify the keyword unambiguously. In the command reference sections that follow, characters that you can omit appear within square brackets ([ ]).

Multiple adjacent spaces and tabs are compressed and treated as a single space. The program ignores leading and trailing spaces.

You can use command qualifiers after the command keyword or after any symbol or number in the command. See individual keyword descriptions for examples.

Most numbers in console commands are in hexadecimal notation. However, the console program does accept decimal notation for console baud rate, register names (R0, R1, and so on), and vector registers.

You can use uppercase or lowercase characters for input. The console program converts all lowercase characters to uppercase.

A command line with no characters is a null command. The console program takes no action and does not issue an error message. The console prompt returns.

5.6 BOOT

The BOOT command initializes the system and begins the boot program. See Section 4.1 for information on how booting works on a VAX 6000 series system. The examples are explained in Section 5.6.2. For details on the SET BOOT command, see Section 5.18.1.

5.6.1 BOOT Command Examples and Qualifiers

Examples

1.
```
>>> BOOT                ! Boots from the special boot
                        ! specification named DEFAULT.
```

2.
```
>>> BOOT/XMI:C DU0      ! Boots from a disk with hex unit no. 0 con-
                        ! nected via a KDM70 controller at XMI node C.
```

3.
```
>>> BOOT DIAG           ! Boots from the saved boot specification
                        ! that was created and given the name DIAG.
```

4.
```
>>> BOOT /XMI:E/BI:4/R5:70000000/NODE:0E02 DU0
                        ! Boots on a VAXcluster from an HSC con-
                        ! troller dual-ported at unit numbers 0E and 02
                        ! with a system root of SYS7.  A VAXBI device
                        ! is used to boot this system.
                        ! See Section 4.6.2.
```

5.
```
>>> BOOT /XMI:E /FILENAME:ISL_LVAX_B EX0
                        ! Boots VAX/DS from an Ethernet-based CD server.
                        ! This example shows access from  a DEMNA adapter.
                        ! CD server can also be accessed over the FDDI
                        ! using the DEMFA adapter; device then would be FX0.
                        ! The server boots the system with the file
                        ! ISL_LVAX_B.
```

6.
```
>>> BOOT /XMI:E /DSSI_NODE:4 /PORT:1 DI5
             ! Boots from a disk with unit number 5 connected by the
             ! KFMSA adapter and the controller at DSSI node 4, port 1.
```

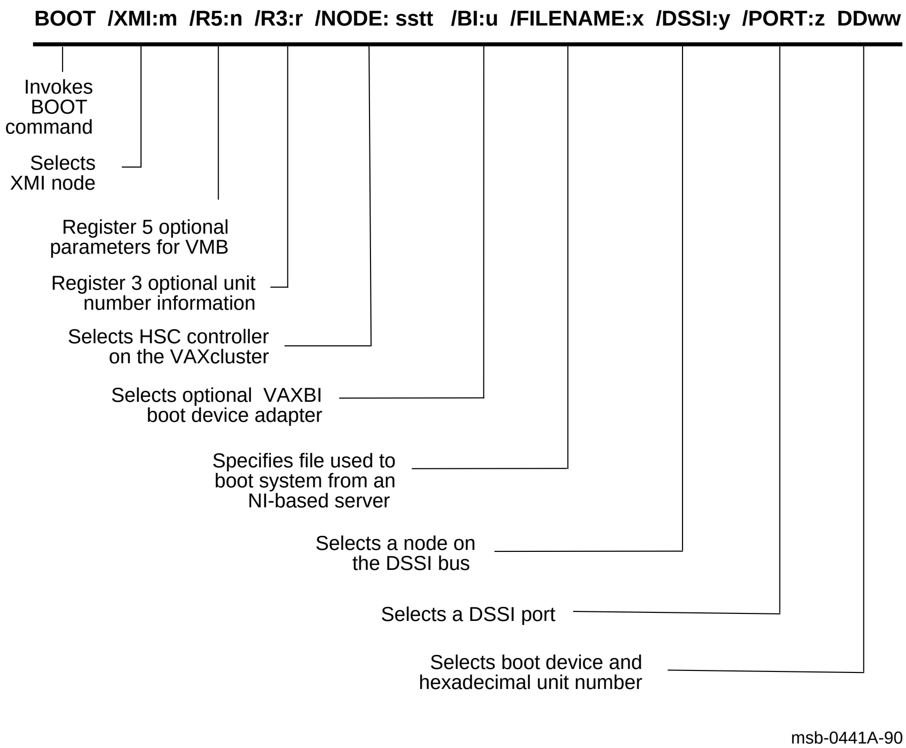
7.
```
>>> BOOT /XMI:E /DSSI_NODE:5 /PORT:2 MI3
             ! Boots from a TF tape on a DSSI.
```

Table 5–5: BOOT Command Qualifiers

Qualifier	Function
/X[MI]:number	Specifies the XMI node number of the node that connects the boot device.
/R5:number	Specifies the hexadecimal value to be loaded into register R5 immediately before the virtual memory boot (VMB) program receives control. Use as a bit mask to select VMB options and to set the system root directory. See Appendix F.
/R3:number	Specifies the hexadecimal value to be loaded into register R3 immediately before the virtual memory boot (VMB) program receives control. This qualifier is used when multiple unit numbers must be specified: for example, when booting from VMS shadow sets. If /R3 is specified, the unit number portion of the device name is ignored.
/N[ODE]:number	Specifies the remote node(s) that provide access to the boot device. The /XMI (and optionally /BI) qualifiers must have identified a controller that supports "nodes" such as a VAXcluster adapter. The /NODE qualifier would then specify the VAXcluster node number(s) of the HSC controlling the boot device.
/B[I]:number	Specifies a VAXBI node that connects the boot device. The /XMI qualifier must have selected a node containing a DWMBB/A.
/FILE[NAME]:file	Specifies the file name used to boot from an Ethernet-based server. The file name can be 1 to 16 characters in length.
/D[SSI_NODE]:number	Specifies the DSSI node that provides access to the boot device. The /XMI qualifier must have selected a node containing a KFMSA adapter.
/PO[RT]:number	Specifies DSSI port 1 or 2 on the KFMSA adapter.

5.6.2 BOOT Command Description

Figure 5–3: BOOT Command



The BOOT command syntax is:

```
B[OOT][/qualifier] [<parameter>]
```

BOOT command qualifiers are summarized in Table 5–5. The qualifier includes a variable which is a node number, a value to be loaded into a register, or the name of a file when using the /FILENAME qualifier. A variable is a required argument to the qualifier. If you do not specify a variable, you receive an error message in the form:

```
?0021 Illegal command
```

In the syntax <parameter> can be a string of the form *ddnn*. The variable *dd* is a 2-character mnemonic for the device type (MI or MU for tape, DU or DI for disk, EX, FX, or ET for Ethernet, or CSA1 for the in-cabinet console load device), and *nn* is a 1- or 2-digit hexadecimal number for the boot device. The *nn* portion of the boot device is ignored if the /R3 qualifier is used.

You can also use <parameter> as a 1- to 4-character name of a saved boot specification that you have created. Your saved specification needs to supply values for the boot device and other qualifiers, if required. You can override any saved qualifier value by specifying the qualifier with a new value. For information on creating a saved boot specification, see Section 5.18.1.

If you omit <parameter>, the program uses the default saved boot specification. You define a default saved boot specification by using the reserved name DEFAULT and the SET BOOT command. Use unique names when you name your saved boot specifications. To avoid confusion, choose names for saved boot specifications that are distinct from the actual device names.

When you have successfully specified the command, your console terminal waits while the system initializes itself and performs self-test. When the operating system comes up, your console terminal displays the login banners of the operating system, and your console terminal is then operating in program mode.

5.7 CLEAR EXCEPTION

The CLEAR EXCEPTION command is used to clear error states in registers.

Examples

1.

```
>>> EXAMINE 21800000                    ! Attempt to examine a non-
?0029 Machine check accessing memory.  ! existent address produces
                                        ! a machine check and sets
>>> CLEAR EXCEPTION                     ! some error bits.
XBE0   = 000000C0                       ! Register values before
XFADR0 = 61900008                       ! CLEAR EXCEPTION clears
XBEER0 = 01240001                       ! the error bits.
PCSTS  = 000008C0

>>>
```

2.

```
>>> ^P

>>> SHOW CONFIGURATION
      Type                Rev
  1+  KA65A      (8080)   0006
  2+  KA65A      (8080)   0006
  3+  KA65A      (8080)   0006
  4+  KA65A      (8080)   0006
  5+  FV64A      (0000)   0001
  6+  MS65A      (4001)   0084
  8+  MS65A      (4001)   0084
  9+  MS65A      (4001)   0084
  A+  MS65A      (4001)   0084
  D+  CIXCD      (0C05)   1652
  E+  DEMNA      (0C03)   0600

>>> CLEAR EXCEPTION                     ! Issued to clear error
XBE0   = 00000081                       ! state that exists.
XFADR0 = 61900008
XBEER0 = 01240001
PCSTS  = 000008C0

>>> CONTINUE
```

The CLEAR EXCEPTION command syntax is:

```
CL[EAR] EX[CEPTION]
```

This command is used to clear error states that remain from a previous command, a sequence of commands, or a console action. The console displays what is currently in the registers shown in the examples and then clears write-one-to-clear bits in the XBER, XBEER, and CPU-specific registers.

In most cases the console program cleans up the error state. However, if an error state is generated after a CTRL/P, the CONTINUE command will not clean up error state. For example, if you type CTRL/P, then issue a SHOW CONFIGURATION command, and then CONTINUE, some error state exists and may cause spurious errors. CLEAR EXCEPTION should be used to clear these errors before giving a CONTINUE command.

5.8 CONTINUE

The CONTINUE command begins processing at the point where it was interrupted by a CTRL/P console command. Programs continue processing at the address currently in the program counter of the processors.

Example

```
$ ^P                          ! Stops processing on boot procesor;
                              ! processor enters console mode.
                              !
?0002 External halt (CTRL/P, break, or external halt)
    PC  = 801DBAA6            ! System responds with error message;
    PSL = 04C38201            ! system has halted with address
    ISP = 80B15200            ! 801DBAA6 in the program counter (PC).
                              !
>>>                           !
>>> [console session begins] !
    .                         !
    .                         !
    .                         !
                              !
>>> CONTINUE                  ! Processing resumes at the address
                              ! where processing was stopped by
                              ! CTRL/P.  Here, processing continues
                              ! at address 801DBAA6.
```

The CONTINUE command takes no arguments. Its syntax is:

```
C[ONTINUE]
```

CONTINUE causes the processor to resume program mode, executing at the address currently in the program counter (PC). This address is the address that was in the PC when the primary processor received the CTRL/P input to interrupt processing and change to console mode. The system displays the hexadecimal PC and the hex values for PSL and –SP (see Appendix H).

When the boot processor receives a CONTINUE command, it does not perform processor initialization as it would for a boot procedure. The boot processor just returns to the program it was processing.

Following execution of the CONTINUE command, the console terminal enters program mode, and any ASCII characters entered on the console terminal are passed on to the operating system. In program mode, the console terminal acts like any other terminal on the system, until a CTRL/P is issued to toggle it back to console mode.

NOTE: *The CONTINUE command should be used selectively. Do not use the CONTINUE command to return to program mode if you have modified memory in console mode.*

5.9 DEPOSIT

The DEPOSIT command stores data in a specified address.

5.9.1 Syntax and Qualifiers

Table 5–6: DEPOSIT Command Qualifiers

Qualifier	Meaning
/B	Defines data size as a byte.
/G	Defines the address space as the general register set, R0 through R15.
/I	Defines the address space as the internal processor registers, accessed through MTPR and MFPR instructions.
/L	Defines data size as a longword; initial default.
/M[1]	Defines the address space as a vector indirect register; accesses addresses 400 and higher.
/N:<count>	Defines the address space as the first of a range.[2] <count> is a required value with /N.
/P	Defines the address space as physical memory; initial default.
/Q	Defines data size as a quadword; initial default for vector registers (except for VCR and VLR).
/V	Defines the address space as virtual memory. All access and protection checking occur. Use when your operating system has been running prior to system halt.[3]
/VE[1]	Defines the address space as the vector register set.
/W	Defines data size as a word.

[1]Used only with an attached vector processor (Models 400 and 500).

[2]The console deposits to the first address, then to the specified number of succeeding addresses. Even if the address is '–', the succeeding addresses are at higher addresses (that is, the symbol specifies only the starting address, not the direction).

[3]If memory management has not been enabled, virtual addresses are equal to physical addresses. If access is not allowed to a program running with the current processor status longword (PSL), the console issues an error message. Virtual space deposits cause the PTE<M> bit to be set in the mapping PTE and force the processor write buffer to be flushed.

The DEPOSIT command syntax is:

```
D[EPOSIT] [/qualifier] <address>  <data>
```

where /qualifier is a value from Table 5–6, and the variable <data> is a hexadecimal value to be stored. The value must fit in the data size to be deposited. The variable <address> is a 1- to 8-digit hexadecimal value or one of the following:

- PSL, the processor status longword. You cannot use any address space qualifier with PSL.
- PC, the program counter. The address space is set to /G.
- SP, the stack pointer. The address space is set to /G.
- Rn, the general purpose register *n*. The register number is in decimal. The address space is set to /G.
- For use with an attached vector processor:
  VCR, 7-bit Vector Count Register. No address qualifier is permitted.
  VLR, 7-bit Vector Length Register. No address qualifier is permitted.
  VMR, 64-bit Vector Mask Register. No address qualifier is permitted.
  V0–V15, vector registers. Elements of a vector register are specified V*n:mm*, where *n* is a decimal number 0–15 specifying the vector register, and *mm* is a hex number 0–3F specifying the element within the vector register. The address qualifier must be set to /VE.
- +, the location immediately following the last location you referenced in an EXAMINE or DEPOSIT command. For physical and virtual memory, the referenced location is the last location plus the size of the reference (1 for byte, 2 for word, 4 for longword). For other address spaces, the address is the last referenced address plus one.
- –, the location immediately preceding the last location you referenced in an EXAMINE or DEPOSIT command. For physical and virtual memory, the referenced location is the last location minus the size of the reference (1 for byte, 2 for word, 4 for longword). For other address spaces, the address is the last referenced address minus one.
- *, the last location you referenced in an EXAMINE or DEPOSIT command.
- @, the location addressed by the last location you referenced in an EXAMINE or DEPOSIT command.

If no qualifiers are given with subsequent commands, the system uses the qualifiers from the preceding command as the defaults. With the /M qualifier, the address is a 3-digit hex number (400 or above).

5.9.2 Examples

Examples

1.
```
>>> D/P 27 0                 !  Deposits the value of 0 to physical
                             !  address 27.
```

2.
```
>>> D/N:8 R0 FFFF            !  Loads registers R0-R8 with FFFF.
```

3.
```
>>> DEPOSIT/P/B/N:1FF 0 0    !  Deposits zeros to the first
                             !  512 bytes of physical memory
                             !  beginning with address 0.
```

4.
```
>>> DEPOSIT/VE V12 0         !  Deposits zero into all 64 elements
                             !  of vector register V12.
```

5.
```
>>> DEPOSIT VLR 1            !  Deposits one in the Vector Length
                             !  Register.
```

6.
```
>>> DEPOSIT/M  440 0         !  Deposits zeros to vector indirect
                             !  register with address 440 (hex).
```

The DEPOSIT command directs data into the specified address. If you do not specify any address space or data size qualifiers, the defaults are the last address space or data size specified in a DEPOSIT or EXAMINE command. After processor initialization, the default address space is physical memory, the default data size is longword, and the default address is zero.

If the specified value is too large to fit in the data size, the console program ignores the command and issues an error message. If the specified value is smaller than the data size to be deposited, the console program fills the high order data positions with zeros. If you specify conflicting data sizes or address spaces, the console program ignores the command and issues an error message.

5.10 EXAMINE

The EXAMINE command displays the contents of a specified address. The qualifiers are identical to the DEPOSIT command's qualifiers.

5.10.1 Syntax and Qualifiers

Table 5–7: EXAMINE Command Qualifiers

Qualifier	Meaning
/B	Defines data size as a byte.
/G	Defines the address space as the general register set, R0 through R15.
/I	Defines the address space as the internal processor registers, accessed through MTPR and MFPR instructions.
/L	Defines data size as a longword; initial default.
/M[1]	Defines the address space as a vector indirect register; accesses addresses 400 and higher.
/N:<count>	Defines the address space as the first of a range.[2] <count> is a required value with /N.
/P	Defines the address space as physical memory; initial default.
/Q	Defines data size as a quadword; initial default for vector registers (except for VCR and VLR).
/V	Defines the address space as virtual memory. All access and protection checking occur.[3]
/VE[1]	Defines the address space as the vector register set.
/W	Defines data size as a word.

[1]Used only with an attached vector processor (Models 400 and 500).

[2]The console examines the first address, then the specified number of succeeding addresses. Even if the address is '–', the succeeding addresses are at higher addresses; that is, the symbol specifies only the starting address, not the direction.

[3]If memory management has not been enabled, virtual addresses are equal to physical addresses. If access is not allowed to a program running with the current processor status longword (PSL), the console issues an error message. Virtual space deposits cause the PTE<M> bit to be set in the mapping PTE and force the processor write buffer to be flushed.

The EXAMINE command syntax is:

```
E[XAMINE] [/qualifier] [<address>]
```

where /qualifier is a value from Table 5–7, and <address> is a 1- to 8-digit hexadecimal value or one of the following:

- PSL, the processor status longword. You cannot use any address space qualifier with PSL.
- PC, the program counter. The address space is set to /G.
- SP, the stack pointer. The address space is set to /G.
- Rn, the general purpose register *n*. The register number is in decimal. The address space is set to /G.
- For use with an attached vector processor:
  VCR, 7-bit Vector Count Register. No address qualifier is permitted.
  VLR, 7-bit Vector Length Register. No address qualifier is permitted.
  VMR, 64-bit Vector Mask Register. No address qualifier is permitted.
  V0–V15, vector registers. Elements of a vector register are specified V*n:mm*, where *n* is a decimal number 0–15 specifying the vector register, and *mm* is a hex number 0–3F specifying the element within the vector register. The address qualifier must be set to /VE.
- +, the location immediately following the last location you referenced in an EXAMINE or DEPOSIT command. For physical and virtual memory, the referenced location is the last location plus the size of the reference (1 for byte, 2 for word, 4 for longword). For other address spaces, the address is the last referenced address plus one.
- –, the location immediately preceding the last location you referenced in an EXAMINE or DEPOSIT command. For physical and virtual memory, the referenced location is the last location minus the size of the reference (1 for byte, 2 for word, 4 for longword). For other address spaces, the address is the last referenced address minus one.
- *, the last location you referenced in an EXAMINE or DEPOSIT command.
- @, the location addressed by the last location you referenced in an EXAMINE or DEPOSIT command.

If no qualifiers are given with subsequent commands, the system uses the qualifiers from the preceding command as the defaults. With the /M qualifier, the address is a 3-digit hex number (400 or above).

5.10.2 Examples

Examples

1.
```
>>> E/N:8 R0                  !  Examines registers R0-R8.
```

2.
```
>>> EXAMINE/P/B/N:1FF         !  Examines the first 512 bytes.
```

3.
```
>>> EXAMINE/N:5/W/P -         !  Examines the previous word
                              !  in the physical address space
                              !  and the next five words.
```

4.
```
>>> E/I 3E                    !  Examines the system ID register.
                              !  System responds with output.
   I   0000003E   0B000001
```

5.
```
>>> EXAMINE VLR               !  Examines the Vector Length
                              !  Register.
  M 00000001 0E
```

6.
```
>>> EXAMINE/M  440            !  Examines the vector indirect
                              !  register at hex address 440.
  M 440 FFFFFFFF 00000000     !  /M is used to access vector
                              !  indirect registers.
```

7.
```
>>> 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
```

The system response to the EXAMINE command is in hexadecimal notation:

```
<address space identifier>  <address>  <data>
```

where <address space identifier> can be one of these values:

- P — Physical memory. When virtual memory is examined, the <address space identifier> is P and <address> is the translated physical address.
- G — General register.
- I — Internal processor register.
- For use with an attached vector processor:
  M — Vector indirect register. This identifier is also returned when the PSL is examined.
  VE — Vector data register.

5.11 FIND

The FIND command causes the console program to search main memory starting at address zero for a page-aligned 256-Kbyte block of good memory (that has no errors) or for a restart parameter block (RPB). If the block is found, its address plus 512 is left in the stack pointer. If the block is not found, an error message is issued.

Examples

1.
```
>>> FIND/MEMORY                ! Searches for a 256-Kbyte memory
>>>                            ! block; returns prompt when found.
```

2.
```
>>> F                          ! Searches for restart parameter
>>>                            ! block; returns prompt when found.
```

3.
```
>>> F/RPB                      ! Searches for a restart parameter
>>>                            ! block; returns prompt when found.
```

Table 5–8: FIND Command Qualifiers

Qualifier	Meaning
/ME[MORY]	Searches for a 256-Kbyte memory block.
/RP[B]	Searches for a restart parameter block. This is the default qualifier. Usually used when the system has been running prior to a system halt. If you use this qualifier before the system has run the operating system, the command searches all memory, which causes a long delay.

The FIND command syntax is:

```
F[IND] [/qualifier]
```

where /qualifier is either /MEMORY or /RPB. The FIND command searches main memory to find a page-aligned 256-Kbyte block of good memory or a restart parameter block. If you do not use a qualifier, the FIND command searches for a restart parameter block, as if you used a /RPB qualifier. There is a wait, while the system searches all memory. This may take up to 2 minutes for each 32 Mbytes of memory.

On some VAX systems, the FIND command is a necessary step in the system boot procedure. However, on a VAX 6000 series system, the boot program includes the process of finding the appropriate memory block to boot. You do not use this command during normal boot procedures.

When the memory block is found, its address plus 512 is left in the stack pointer (SP). This convention is established because you load the virtual memory block (VMB) program into the memory block you just found. VMB uses the first page of memory to build the restart parameter block (RPB).

5.12 HALT

The HALT command is a null command for the system operating in console mode. The command is accepted, but no action is taken since the processor has already halted in order to enter console mode.

Example

```
>>> HALT                ! You enter the HALT command.
                        !
    ?0026 Halted        ! System responds with error message
                        ! that indicates the system already
                        ! is halted.
```

The HALT command syntax is:

```
HALT
```

where the command takes no arguments.

On other VAX systems, the HALT command stops the processors. However, on VAX 6000 series systems, HALT has no effect, because the boot processor is already halted as a requisite condition for console mode.

See the STOP command, Section 5.21.

5.13 HELP

The HELP command provides basic information on the console commands, when the console terminal is in console mode.

Examples

1.
```
>>> HELP
  BOOT              FIND          REPEAT            SHOW     UNJAM
  CLEAR_EXCEPTION   HALT          RESTORE_EEPROM    START    UPDATE
  CONTINUE          HELP          SAVE_EEPROM       STOP     Z
  DEPOSIT           INITIALIZE    SET               TEST
  EXAMINE
    SelfTest_Output     CTRL_Characters     !

  For more information, type HELP <topic>.
```

2.
```
>>> HELP FIND
   FIND

Searches memory for the specified item.

Qualifiers

    /MEMORY     - Searches for first 256 Kbytes of good memory.
    /RPB        - Searches for a Restart Parameter Block.
```

3.
```
>>> HELP INIT
   INITIALIZE [qualifiers] [node]

Resets the specified XMI node.  If node  number is omitted,
resets the entire system.

Qualifiers

   /BI:bi-node  - Resets the specified BI node. The node
                  parameter must specify an XMI-to-BI adapter.
```

4.
```
>>> HELP !
! comment

Treats the remainder of the command line as a comment.
```

The syntax for the HELP command is:

```
HELP [<command>]
```

where <command> is one of the entries listed in the main HELP printout.

The HELP command operates when the console program error messages are set in English mode (see Section 5.18.3). To see a list of all HELP files available, enter HELP or HELP HELP at the console prompt, followed by a carriage return. The system responds with a list of available HELP files.

When you issue a SET LANGUAGE INTERNATIONAL command and then enter HELP [<command>], you receive the error message:

```
?005C
```

From the error messages in Table H–2, you can see that ?005C indicates that no help is available.

5.14 INITIALIZE

The INITIALIZE command performs a reset. You can initialize the entire system, a specified XMI node (except memory), or a specified VAXBI node.

Examples

1.
```
>>> INITIALIZE 1          ! Initializes node 1 on the XMI.
                          ! No self-test results are displayed.
```

2.
```
>>> I/B:2 E               ! Initializes node 2 on a VAXBI
                          ! where E is the node on the XMI
                          ! that goes to node 2 on the VAXBI.
```

3.
```
>>> I                     ! Resets the entire system.

#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
    o   +   .   .   .   +   +   +   +   .   +   +   +   +       STF
    .   .   .   .   .   .   .   .   .   .   E   D   E   B       BPD
    .   .   .   .   .   .   .   .   .   .   +   +   +   +       ETF
    .   .   .   .   .   .   .   .   .   .   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

>>>                          ! Prompt returns following self-test.
```

Table 5–9: INITIALIZE Command Qualifiers

Qualifier	Meaning
/B[I]:<VAXBI-node>	Can be used only if the specified XMI node is a DWMBB/A. This qualifier resets the single adapter at node <VAXBI-node> on the specified VAXBI.
None	If no XMI node number is given and a /BI qualifier is omitted, the system resets all nodes on the XMI (and optional VAXBI) and prints out self-test results.

The INITIALIZE command syntax is:

```
I[NITIALIZE] [/qualifiers] [<XMI-node>]
```

where <XMI-node> indicates the XMI node to be initialized.

The INITIALIZE command can be used to reset the system or a specified node — except memory nodes. If <XMI-node> is omitted, the entire system is reset. If the /BI qualifier is used, the XMI node is a DWMBB that connects to the VAXBI bus containing the VAXBI node to be initialized. If the <XMI-node> is a DWMBB and the /BI qualifier is not used, then all devices on the VAXBI bus are reset.

The INITIALIZE command and the Restart button on the system control panel perform the same function: they both reset the machine.

If the node number you designate in an INITIALIZE command does not have a module in it, you receive an error message of the form:

```
?0043 Unable to initialize node.
```

If the node number specifies a memory module, you receive this message:

```
?0049 Memory cannot be initialized.
```

Self-test results are displayed after a system reset but not after a node reset. See Section 6.2 through Section 6.7 for how to interpret self-test results.

5.15 REPEAT

The REPEAT command reexecutes the command that you pass as its argument. You can use the REPEAT command with any command except itself. The key combination CTRL/C stops the REPEAT command.

Example

```
>>> REPEAT E/P 10            ! Continuously displays the
                             ! contents of physical memory at
                             ! address location 10.  To stop
                             ! the display, enter a CTRL/C.
```

The REPEAT command syntax is:

```
R[EPEAT] <command>
```

where <command> is any command other than REPEAT.

REPEAT works as a continuous repeat. The command you pass as an argument to the REPEAT command continues to be executed until you stop the process with CTRL/C.

5.16 RESTORE EEPROM

The RESTORE EEPROM command can be used if your system has a TK tape drive[1] as a console load device. This command copies the TK tape's EEPROM image to the EEPROM of the boot processor.

Example

```
                  ! Lower key switch must be in Update position.
                  ! Load the TK tape with EEPROM contents.
                  ! When the yellow light on the TK70 drive
                  ! stays on, enter RESTORE command.
>>> RES E

?006E EEPROM Revision = x.xx/y.yy
?0070 Tape image Revision = x.xx/y.yy

                  ! System displays the revision level of the
                  ! EEPROM and the TK tape.

                  ! Console program asks if you want to restore;
                  ! the default is no.  Enter Y to continue.

Proceed with EEPROM update? (Y or N) >>> Y
?006A EEPROM changed successfully.

>>>               ! Restore complete; prompt returns.
```

The RESTORE EEPROM command syntax is:

```
RES[TORE] E[EPROM]
```

The RESTORE EEPROM command copies information from the TK tape that you previously saved by using the SAVE EEPROM command (see Section 5.17). Before the information is copied to the EEPROM of a processor, you are shown the revision level of the information that resides on the tape as well as information that presently resides on the EEPROM. Then you are asked if you wish to continue the restore operation.

The steps for using RESTORE EEPROM include:

1. Load the TK tape cartridge containing saved EEPROM data into the TK tape drive. This rewinds the tape to the beginning, so that restoration proceeds from this point. See Appendix B for information on the tape drive operation.

[1] RESTORE EEPROM not implemented on TF devices.

2. Put the control panel's lower key switch in the Update position (see Section 3.3).

3. Put the control panel's upper key switch in the Enable position, and type CTRL/P at the console terminal to put the terminal in console mode (see Section 5.3).

4. Move to the processor whose EEPROM contents you wish to restore. Normally, all EEPROM contents will be the same. If you are restoring the contents of the boot processor, proceed to the next step. If you wish to restore the contents of a secondary processor, change the boot processor using the SET CPU command (see Section 5.18.2).

5. At the prompt, enter RESTORE EEPROM. The console program queries you, requiring your confirmation to proceed with the RESTORE EEPROM operation.

6. Enter *Y* to indicate your intention to proceed. The restore process takes less than 2 minutes to complete.

7. When the console prompt returns, the restore operation is complete. Restored information includes:

    Systemwide console parameters
      (baud rate, interleave, terminal characteristics)
    Saved boot specifications
    Diagnostic patches
    Console patches
    Boot primitives

8. Rewind the TK tape.

9. Reset the system using the INITIALIZE command or the control panel Restart button. All restored changes are visible following a system reset.

10. Use the SHOW command to verify the contents of the EEPROM.

NOTE: *With the UPDATE command you can restore EEPROM contents on all secondary processors (see Section 5.24), but you must be sure all ROMs are the same revision. The EVUCA program can also be used to restore the EEPROM contents (see Appendix E).*

Because each system has its own identifying information stored in the EEPROM, only the TK tape for that system should be used for a RESTORE EEPROM operation.

5.17 SAVE EEPROM

The SAVE EEPROM command can be used if your system has a TK tape drive[1] as a console load device. This command copies the EEPROM contents of the boot processor to the TK tape.

Examples

1.

```
                             ! Load a TK tape. When the yellow
                             ! light on the TK drive stays on,
>>> SAVE EEPROM              ! the tape is ready.  Enter SAVE command.
                             ! System prompts user to proceed.  Enter
                             ! a Y to continue.

Proceed with save to tape? (Y or N) >>> Y
?006C EEPROM saved to tape successfully.
>>>                          ! System confirms SAVE is complete.
```

2.

```
>>> SA E                      ! SAVE EEPROM to the TK tape.
                              !
?003D  Error initializing I/O device.
                              ! TK tape not initialized.
                              ! Reload and reenter the command.
```

The SAVE EEPROM command syntax is:

```
SA[VE] E[EPROM]
```

SAVE EEPROM copies information from the EEPROM of the boot processor to the beginning of the tape in the TK tape drive. As the information is copied, the TK controller writes a block and then checks it against the contents of the EEPROM to verify. You should save the contents of the EEPROM every time customer service installs a new EEPROM patch level.

The SAVE EEPROM command overwrites whatever is on the TK tape. To be safe, use a blank tape cartridge.

There are several steps to the SAVE EEPROM procedure:

1. Load a TK tape cartridge (write-enabled) into the TK tape drive and press the control panel button. This rewinds the tape so it records from

[1] SAVE EEPROM not implemented on TF devices.

the beginning of the tape. See Appendix B for information on the tape drive operation.

2. Put the control panel's upper key switch in the Enable position, and type CTRL/P at the console terminal to put the terminal in console mode (see Section 5.3).
3. If you wish to save the contents of a secondary processor's EEPROM, first make it the boot processor using the SET CPU command. (See Section 4.5 and Section 5.18.2.)
4. At the prompt, enter SAVE EEPROM. This operation overwrites any existing information on the TK cartridge, so be sure you have inserted an appropriate tape.
5. The console program queries you, requiring your confirmation to proceed with the SAVE EEPROM operation.
6. Enter Y to indicate your intention to proceed. The save process takes less than a minute to complete.
7. The console program confirms that the save operation has completed successfully. When the console prompt returns, the save operation is complete. Saved information includes:

   Systemwide console parameters
      (baud rate, interleave, terminal characteristics)
   Saved boot specifications
   Diagnostic patches
   Console patches
   Boot primitives

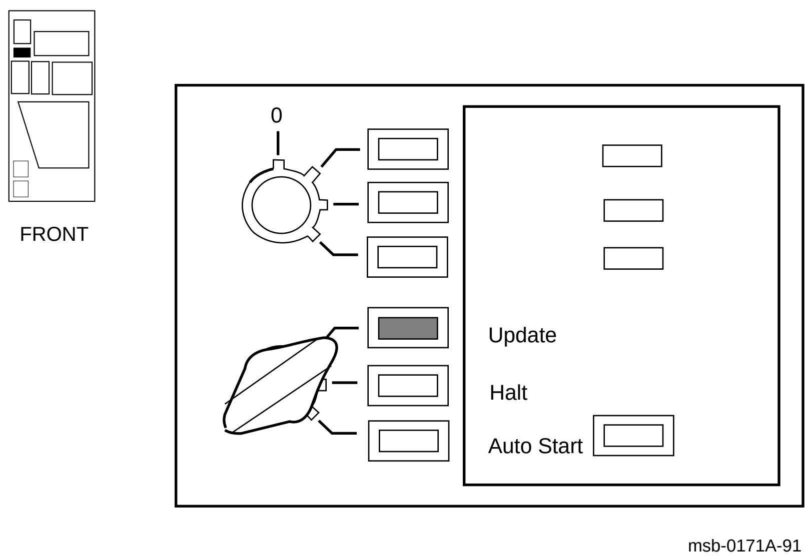
8. Rewind the tape. When the green light turns on and the beep sounds, you can remove the tape. Label and write-protect the tape (see Section B.4).

Because each system has its own identifying information stored in the EEPROM, only the TK tape for that system should be used for a SAVE EEPROM operation.

5.18 SET Commands

SET commands allow you to change the configuration parameters on the boot device, primary processor, memory, and terminal, and to modify the output of the error messages. To store the new parameters in the processor's EEPROM, the control panel's lower key switch must be in the Update position. Some SET commands take effect immediately, but the changes will be lost at the next node or system reset if the EEPROM is not updated.

Figure 5–4: Lower Key Switch in Update Position



This section describes the following SET commands:

- SET BOOT
- SET CPU
- SET LANGUAGE
- SET MEMORY
- SET TERMINAL

If you issue a set command and the control panel's lower key switch is not in the Update position, you may receive the following error message:

```
?0040  Key switch must be at "Update" to update EEPROM.
```

Despite the error message, the change completes and remains until the primary processor or system is reset.

Whenever you issue a SET command, the console program tries to pass the current values of all parameters that can be set to all system processors. This requires that all system processors be in console mode. You receive an error message if any processor is still running. Processors not in console mode are not updated. Use the STOP command to stop each processor, or issue an INITIALIZE command to stop all processors. (See Section 5.21 and Section 5.14.)

When you issue a system reset, the system checks the validity of the systemwide parameters on each node and sends an error message to you if the settings on any node do not match those on the current boot processor or are corrupted.

5.18.1 SET BOOT

The SET BOOT command allows you to store a BOOT command by a nickname for easy reference. Then you can reference the full BOOT command by the nickname. The lower key switch on the control panel must be set to Update.

Examples

1.
```
                              ! Turn key switch to Update.
>>> SET BOOT DIAG /XMI:D DU0
>>>        ! This creates a saved boot specification called DIAG
           ! that boots the disk unit 0 from XMI node D.
           ! SHOW BOOT command displays all saved BOOT specifications.
>>> SHO BOOT
  DEFAULT  /XMI:D DU1
  DIAG     /XMI:D DU0
```

2.
```
>>> SET BOOT DIAG             ! Removes the boot specification saved
                              ! under the name DIAG.
>>> SHO BOOT                  ! SHOW BOOT command confirms DIAG is re-
  DEFAULT  /XMI:D DU1         ! moved from saved BOOT specifications,
                              ! from example #1 above.
```

3.
```
>>> SET BOOT DEFAULT /XMI:E /NODE:0405 /R5:40000000 DU0
                              ! Sets the default boot device for the
                              ! system to be disk unit 0 (DU0) on the
                              ! VAXcluster; booting from system root 4.
                              ! 04 and 05 are the specified VAXcluster
                              ! node numbers of the HSC that controls
                              ! the boot device.
                              ! SHOW BOOT command confirms DEFAULT is
>>> SHO BOOT                  ! changed in saved BOOT specifications.
  DEFAULT  /R5:40000000 /XMI:E /NODE:0405 DU0
```

The SET BOOT command syntax is:

```
SE[T] B[OOT] <nickname> [<boot-parameters>]
```

where <nickname> is a 1- to 4-character name for the boot specification you are saving.

The string <boot-parameters> is any legal set of BOOT command parameters and qualifiers that do not reference another saved boot specification. If you omit <boot-parameters>, you delete the saved boot specification (if any) associated with <nickname>. The lower key switch on the control panel must be set to the Update position.

You can store up to 10 saved boot specifications plus the default specification. Avoid using saved boot specification nicknames that are identical to device specifications.

A Digital customer service engineer sets the system default boot device at installation. The default is chosen to point to the system disk or VAXcluster disk and to allow the system to reboot automatically after a power interruption.

Before you name a boot specification, you may want to enter:

```
SHOW BOOT
```

This command displays all boot specification names that have been saved to date. See Section 5.19 for additional information on the SHOW BOOT command.

5.18.2 SET CPU

The SET CPU command allows you to specify a particular processor as the primary processor or designate its eligibility to become the primary processor. You can also disable a vector processor module.

5.18.2.1 Syntax and Qualifiers

Table 5–10: SET CPU Command Qualifiers

Qualifier	Meaning
/E[NABLED] /ALL	Processor is included in the system configuration and is enabled to leave console mode. With the /ALL qualifier all processors are enabled to leave console mode.
/NO_E[NABLED]	Processor is disabled from leaving console mode; START, BOOT, and CONTINUE commands are ignored.
/NEX[T_PRIMARY]	Processor will be the primary (boot) processor at the next system reset.
/P[RIMARY] /ALL	Processor will be eligible to be selected as the primary (boot) processor at the next system reset. With the /ALL qualifier all processors are eligible to become the boot processor; initial default.
/NOP[RIMARY]	Processor will not be eligible to be selected as the primary (boot) processor at the next system reset.
/V[ECTOR_ENABLED][1]	Vector processor attached to the specified scalar processor is included in the system configuration and can be sent vector instructions; initial default.
/NOV[ECTOR_ENABLED][1]	Vector processor attached to the specified scalar processor is excluded from the system configuration.
None	Processor immediately becomes the new primary processor; the next system prompt comes from the new primary processor.

[1]Used only with an attached vector processor (Models 400 and 500).

The SET CPU command syntax is:

```
SE[T] C[PU] [/qualifier] [<XMI-node>]
```

where <XMI-node> is the XMI node number of the processor to be affected. If you omit <XMI-node>, the system uses the current processor.

If you omit all qualifiers, the SET CPU command immediately causes the specified processor to become the primary processor. The console terminal is then connected to the new primary processor, and the next console prompt is generated by the designated processor.

If you use qualifiers, the SET CPU command changes the processor parameters that take effect at the next system reset. These qualifiers modify the EEPROM (if the lower key switch is set to Update) and take effect immediately:

- /ENABLE
- /NOENABLED
- /VECTOR_ENABLED
- /NOVECTOR_ENABLED

The /NEXT_PRIMARY qualifier acts the same as if you had issued a SET CPU/NOPRIMARY command for all other nodes. To undo /NEXT_PRIMARY, you can issue the SET CPU/PRIMARY/ALL command. The /PRIMARY qualifier means that the specified processor is eligible to be the primary processor.

The /NOVECTOR_ENABLED qualifier removes the vector processor from the system configuration. The scalar processor is not affected. The /VECTOR_ENABLED qualifier restores the vector processor to the configuration.

The effect of the SET CPU command qualifiers is shown on the BPD lines of the system self-test display (see Section 6.5).

NOTE: *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.*

5.18.2.2 Examples

Examples

1.
```
>>> SET CPU/PRIMARY 1   ! The processor at XMI node 1 may become the
                        ! primary processor at the next system reset.
```

2.
```
>>> SE CPU 1            ! Processor at XMI node 1 immediately
                        ! becomes the new primary processor.
                        ! The next system prompt is generated
                        ! from the processor at node 1.
```

3.
```
>>> SET CPU/NOVECTOR_ENABLED 4  ! The vector processor attached
                                ! to the scalar processor at node 4
                                ! is disabled.
```

Table 5–11: SET CPU Command Qualifiers' Effect After a System Reset

Qualifier	BPD Value at Next Reset[1]
/NEX[T_PRIMARY]	B for boot processor; must be chosen the boot processor at the next system reset. All other CPUs show as D.
/NOE[NABLED]	D for disable; processor is not included in the configuration.
/NOP[RIMARY]	D for disable; can be only a secondary processor.
/P[RIMARY]	B if selected as the boot processor; E if it is a secondary processor.
/NOV[ECTOR_ENABLED]	D for disable; vector processor is not included in the configuration.
None	B for boot processor.

[1]The key switch must be at Update when the SET CPU command is issued.

5.18.3 SET LANGUAGE

The SET LANGUAGE console command changes the output format of the console error messages. The default is English error messages, as shown in Appendix H.

Examples

1.
```
                                 ! Lower key switch must be in Update
                                 ! position to store change in EEPROM.
>>> SET LANGUAGE INTERNATIONAL ! All error messages now appear as
                                 ! numeric code only, with no
                                 ! English explanation.
>>> CONTINUE                     ! Continue in program mode.
$                                !
$ ^P                             ! A CTRL/P changes to console mode.
 ?0002                           ! System error message indicates
                                 ! external halt; message is in
                                 ! INTERNATIONAL format.
```

2.
```
>>> SET LANGUAGE ENGLISH         ! All error messages now appear with
                                 ! English comments and numeric code.
>>> CONTINUE                     ! Continue in program mode.
$                                !
$ ^P                             ! A CTRL/P changes to console mode.
 ?0002  External halt (CTRL/P, break, or external halt)
      PC  = 801DBAA6             ! System error message indicates
      PSL = 04C38201             ! external halt; message is ENGLISH
      ISP = 80B15200             ! format.
```

3.
```
>>> SET LANGUAGE
?0021 Illegal command            ! Illegal command; requires parameter.
```

Table 5–12: SET LANGUAGE Command Parameters

Parameter	Meaning
ENGLISH	Error messages have both a numeric code and an English explanation.
INTERNATIONAL	English explanations for error codes are suppressed.

The SET LANGUAGE command syntax is:

```
SE[T] LANG[UAGE] <parameter>
```

where <parameter> is a required value from Table 5–12.

The SET LANGUAGE command suppresses English explanations of a command. The default setting is to provide complete information with the error message.

If you use the HELP command while the console program is in International mode, you receive the error message:

```
?005C
```

This indicates that No HELP is available, and the HELP messages have not been translated from English and do not appear in International mode.

Issuing a CTRL/P command to the console program would generate the following error message:

```
?0002 External halt (CTRL/P, break, or external halt)
```

If you then issue a SET LANGUAGE INTERNATIONAL command and again type CTRL/P, the error message reads:

```
?0002
```

While the console program is in International mode, you might receive an error message of:

```
?0021
```

If you then enter SET LANGUAGE ENGLISH and repeat the sequence that generated the error command, the system response is:

```
?0021 Illegal command
```

5.18.4 SET MEMORY

The SET MEMORY command is used to override the system default for interleaving memory. The command takes effect after the next system reset.

Examples

1.
```
>>> SET MEMORY/I:(9+7+6+5, 8, A)
                      ! Explicitly specifies what is created
                      ! as the system memory configuration.
                      ! Four memories are in one set, and 8 and A
                      ! are each in an interleave set of their own.
                      ! Used to exclude a memory from the configuration.
```

2.
```
>>> SET MEMORY  /INTERLEAVE:DEFAULT
                      ! For a system with 6 memory modules:
                      !    4-way interleave of first 4 memories,
                      !    2-way interleave of remaining 2 modules.
                      !
                      ! Assume memory module locations at XMI
                      ! nodes 5 through A.
```

Table 5–13: SET MEMORY Qualifiers

Qualifier	Meaning
/C[ONSOLE_LIMIT]:n	Prevents the console and operating system from using memory above the specified address.
/I[NTERLEAVE]:(interleave-list)	Explicitly specifies how to interleave memory modules.
/I[NTERLEAVE]:D[EFAULT]	Uses the default interleave algorithm.
/I[NTERLEAVE]:N[ONE]	Does not interleave memory.

The SET MEMORY command syntax is:

```
SE[T] M[EMORY] </qualifier>
```

The console program automatically interleaves the memory modules to give the largest possible set. The SET MEMORY command allows you to override the default. This command modifies the configuration stored in the EEPROM. The new configuration takes effect the next time the system is reset or powered up.

An interleave set consists of memory modules in one-, two-, four-, or eight-way configurations. Up to eight interleave sets can be configured. The command can be used to set an upper bound on the memory used by the console.

The default action interleaves memory so that the largest interleave factor is obtained for each group of memory modules. If you have more modules than you can interleave evenly, the console program repeats the criteria with the remaining memory modules until only single arrays remain. An interleave set will be built from like-sized memory modules or from memory modules whose cumulative value is equal to the largest memory module in the set. The console program configures the memory modules starting with the lowest XMI node number, which is placed at the lowest physical address.

Interleave set A always has a starting address of 0. Subsequent interleave sets have starting addresses that are the sum of the memory sizes of preceding interleave sets. As can be seen in the console display, an interleave set number of "–" indicates that the memory module was not included in the configuration.

```
 F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0   NODE #

     .   .   .   .   .   A1  -   .   .   .   .   .   .   .       ILV
     .   .   .   .   .   64  .   .   .   .   .   .   .   .       64 Mb
```

Additional information about the qualifiers includes:

- **/Console_limit:n** allows you to reserve the highest addressed physical memory for special hardware or applications, where *n* is that hexadecimal address. The value is rounded up to the next even page boundary. The console program begins building its in-memory data structures, such as the CCA and bitmap, in memory locations below hexadecimal address *n*.
- For the **/Interleave** qualifier, the interleave-list can have the format of:

  ```
  (node + node ..., node,...)
  ```

  where *node* is the XMI node number of a memory module. Commas separate each set of modules to be interleaved. Each set contains memory modules, separated by plus signs. The console program configures the modules in the order you specify, placing the first module at the lowest physical address.

5.18.5 SET TERMINAL

The SET TERMINAL command sets the characteristics that are stored for the console terminal.

Example

```
>>> SHOW TERMINAL                 ! Enter SHOW TERMINAL.
  /SCOPE    /SPEED: 1200  /BREAK
                                  ! System responds with
                                  ! the parameters stored for
                                  ! /SCOPE, /SPEED, and  /BREAK.
>>> SET TERM/HARDCOPY /SPEED:9600 ! SET TERM changes system
                                  ! parameters.
                                  !
>>> SHOW TERM                     ! Enter the command.
  /HARDCOPY /SPEED: 9600  /BREAK
                                  ! System displays the
                                  ! parameters you set.
```

Table 5–14: SET TERMINAL Command Qualifiers

Qualifier	Meaning
/BREAK	Enables you to adjust the baud rate using the BREAK key.
/NOBREAK	Disables the BREAK key from adjusting the baud rate.
/H[ARDCOPY]	Specifies the console terminal as a hardcopy device.
/NOH[ARDCOPY]	Specifies the console terminal as a video device.
/SC[OPE]	Specifies the console terminal as a video device.
/NOSC[OPE]	Specifies the console terminal as a hardcopy device.
/SP[EED]:n	Sets the baud rate for communication at 300, 600, 1200, 2400, 4800, 9600, 19200, or 38400.*

*38400 is not implemented on Model 600.

The SET TERMINAL command syntax is:

```
SE[T] T[ERMINAL] [/qualifiers]
```

The character format for the SET TERMINAL command is always eight bits—no parity, one stop bit.

This command immediately changes the specified parameter. The new value is stored in the EEPROM if you have the lower key switch set to Update. The EEPROM defaults are /SCOPE, /SPEED:1200, and /BREAK.

The /HARDCOPY qualifier controls the sequence that the console program echoes at the console terminal when you use the Delete key to erase input characters. With the /HARDCOPY parameter set, the console terminal echoes each deleted character within backslashes. The /NOHARDCOPY qualifier causes deleted characters to disappear from the video screen of your terminal.

You can direct the console terminal output to a printer. The VT420 terminal has a Print Screen key (the second key at the top left of the keyboard), or you can select from four printing modes. See *Installing and Using the VT420 Video Terminal*.

5.19 SHOW

The SHOW command displays the current value of parameters specified in a SET command and other configuration information about the system.

Examples

1.
```
>>> SHOW ALL
        Type              Rev      ! Lists all system parameters,
    1+  KA65A      (8080)  0006     ! beginning with the system
    9+  MS65A      (4001)  0084     ! configuration.
    D+  CIXCD      (0C05)  1652
    E+  DEMNA      (0C03)  0601

    Current Primary: 1              ! Shows the status of CPUs
    /NOENABLED-
    /NOVECTOR_ENABLED-              ! Models 400 and 500 only
    /NOPRIMARY-
                                    ! Shows the memory interleave
  F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0   NODE #

      .   .   .   .   .   A1  .   .   .   .   .   .   .   .       ILV
      .   .   .   .   .   64  .   .   .   .   .   .   .   .       64 Mb
    /INTERLEAVE:DEFAULT
    /SCOPE  /SPEED: 1200 /BREAK     ! Shows the terminal characteristics
    English                         ! Shows the language mode
    XMI:E   08-00-2B-08-3D-64       ! Shows the Ethernet address
    DEFAULT /R5:00000010 /XMI:E  DU0 ! Shows Boot specs saved
```

2.
```
>>> SHOW FIELD
  Saved boot specifications:
    DEFAULT /R5:00000010 /XMI:E  DU0

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

  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 #

      .   .   .   .   .   A1  .   .   .   .   .   .   .   .       ILV
      .   .   .   .   .   64  .   .   .   .   .   .   .   .       64 Mb
    /CONSOLE_LIMIT:00100000
    /INTERLEAVE:DEFAULT

  Power system: C

  System serial number: SG01234567
```

The SHOW command syntax is:

```
SH[OW] <object>
```

where <object> is one of these commands:

A[LL]	B[OOT]	CO[NFIGURATION]	CP[U]	D[SSI]
E[THERNET]	F[IELD]	L[ANGUAGE]	M[EMORY]	T[ERMINAL]

Table 5–15: SHOW Commands

Command	Information Displayed
SHOW ALL	Information provided for the SHOW commands. Some information provided for the SHOW FIELD command will not be included when a SHOW ALL is issued.
SHOW BOOT	All BOOT commands and their nicknames that have been created by using the SET BOOT command.
SHOW CONFIGURATION	Hardware device type and revision level for each XMI (and optional VAXBI) node; also indicates whether the node passes or fails self-test. See Appendix C and Appendix D for device type code assignments.
SHOW CPU	The primary processor and the status of other processors. The SET CPU command assigns these values.
SHOW DSSI	DSSI bus numbers, node numbers, and unit numbers for all DSSI devices.
SHOW ETHERNET	Ethernet hardware addresses for all Ethernet adapters on the system and FDDI hardware addresses for all FDDI adapters.
SHOW FIELD[1]	Saved boot commands, console terminal parameters, console language mode, memory configuration, type of power system, and system serial number.
SHOW LANGUAGE	The mode currently set for console error messages, international or English.
SHOW MEMORY	The memory lines from the system self-test. The ILV line indicates the interleave active on the memory arrays, while the second line indicates the size of each memory, its node position, and the total amount of memory on the system.
SHOW TERMINAL	Current parameters set for the console terminal: baud rate and terminal characteristics.

[1]The SHOW FIELD command applies only to Model 500 and 600 systems.

5.20 START

The START command begins execution of an instruction at the address specified in the command string. The START command does not initialize the system.

Examples

1.
```
$ ^P                          ! CTRL/P stops processing;
                              ! system enters console mode.
                              !
?0002 External halt (CTRL/P, break, or external halt)
    PC  = 80159035            ! System responds with error
    PSL = 04C38201            ! message that the system has
    ISP = 80B15200            ! halted with address 80159035
                              ! in the program counter (PC).
>>> [console session begins] !
    .                         !
    .                         !
    .                         !
>>> START                     ! Starts the system at
                              ! address 80159035.
```

2.
```
$ ^P                          ! Stops processing; system enters
                              ! console mode.
                              !
?0002 External halt (CTRL/P, break, or external halt)
    PC  = 80044957            ! System responds with error
    PSL = 04C38201            ! message that the system has
    ISP = 80B15200            ! halted with address 80044957
                              ! in the program counter (PC).
>>> [console session begins] !
    .                         !
    .                         !
    .                         !
>>> START 10000               ! Starts processing at address
                              ! 10000, which is different from
                              ! the address held in the PC at halt.
```

The START command syntax is:

```
STA[RT] [<address>]
```

where <address> is the starting address. If <address> is omitted, the current PC content is used. In this case, the START command has the same effect as the CONTINUE command.

When you specify an address, the START command is the same as executing a Deposit to the program counter (PC) followed by a CONTINUE command. That is, with the START command and an address as an argument, you store an address in the program counter and then call for the system to begin processing at that address.

5.21 STOP

The STOP command halts a specified XMI (or optional VAXBI) node. If the target node is a processor, the processor enters console mode.

Examples

1.
```
>>> STOP 2          ! Stops the processor at node 2 on the XMI.
```

2.
```
>>> STOP E          ! Stops the adapter at node E on the
                    ! XMI. If the adapter at node E is a
                    ! DWMBB/A, then all nodes on the
                    ! VAXBI connected through node E are
                    ! also stopped.
```

3.
```
>>> STOP/BI:6 E     ! Stops the adapter at node 6 on the
                    ! VAXBI accessible through the
                    ! DWMBB/A at node E.
```

4.
```
>>> STO/B:6  E      ! Same as above command.
```

Table 5–16: STOP Command Qualifiers

Qualifier	Meaning
/B[I]:<node>	Specifies a VAXBI node. Causes a VAXBI stop of the node at <node> on the specified VAXBI.
<XMI-node>	Specifies an XMI node. Stops the processor or DWMBB/A at the given node number.

The STOP command syntax is:

```
STO[P] [/qualifier] <XMI-node>
```

where <XMI-node> specifies the XMI node to be halted. If <XMI-node> is a processor module, then only that node is halted. If you stop a processor that is currently running, you receive this message:

```
>>>
Node n:  ?0002 External halt (CTRL/P, break, or external halt)
>>>
Node n:      PC = xxxxxxxx
>>>
```

where *n* is the node you stopped, and *xxxxxxxx* is the address where the processor was halted. If you issue a STOP command to a processor that is in console mode, you do not receive a message. Processors in console mode are already halted.

When the /BI qualifier is used, the XMI node is a DWMBB/A that connects to the VAXBI bus containing the node to be halted.

When the /BI qualifier is not used, as in:

```
STO[P] <XMI-node>
```

if <XMI-node> is a DWMBB/A, all nodes on the VAXBI are halted.

The STOP command has no effect after a system reset. The STOP command does not control the running of diagnostics.

The STOP command does not apply to memories.

5.22 TEST

The TEST command passes control to the system self-test diagnostics.

Example

```
>>> TEST/RBD                    ! Requests ROM-based diagnostics.
                                ! New prompt indicates you are in RBD
RBD1>                           ! monitor program working from the
                                ! device with node number 1.
RBD1> ^Z                        ! You enter CTRL/Z to return
                                ! to console mode.
                                !
?0006 Halt instruction executed in kernel mode.
    PC  = 200601D8              ! Console error message indicates
    PSL = 041F0604              ! RBD program has been halted.
    ISP = 201405B4              !
>>>                             ! Console prompt returns.
```

Table 5–17: TEST Command Qualifiers

Qualifier	Meaning
/R[BD]	Transfers control to the command parser for running ROM-based diagnostics.

The TEST command syntax is:

```
T[EST] </qualifier>
```

The qualifier /RBD transfers control to the command parser for running ROM-based diagnostics. This parser runs various tests and displays the results on the console terminal. Type CTRL/Z or QUIT to return to the console prompt.

If no qualifier is specified, self-test runs on the node at which you type the TEST command. The console prompt then returns. See your system's *Service Manual* for more information on diagnostics.

5.23 UNJAM

The console program accepts the UNJAM command. However, UNJAM has no effect on the system, since this system does not have an independent I/O bus reset option.

Example

```
>>> UNJAM                              ! Enter UNJAM command.
>>>                                    ! Console prompt returns.
                                       ! UNJAM has no effect on the
                                       ! system.
```

The UNJAM command syntax is:

```
UN[JAM]
```

This command is retained for compatibility with other consoles, but has no effect on this system, since the only bus reset is accomplished with a full system reset.

5.24 UPDATE

The UPDATE command copies the contents of the boot processor's EEPROM to the EEPROM of the specified secondary processor. The control panel's lower key switch must be in the Update position to use UPDATE.

Examples

1.
```
                           ! Lower key switch must be set to
                           ! Update; upper switch to Enable.
>>> UPDATE 2               ! Update the processor at node 2.
>>>                        ! There is a pause while this
                           ! command executes.  When the
                           ! console prompt returns, update
                           ! is complete.
```

2.
```
>>> UPDATE ALL             ! Update all the secondary
>>>                        ! processors. There is a pause
                           ! while this command executes.
                           ! When the console prompt returns,
                           ! update is complete.
```

3.
```
>>> UPDATE 1
?0064 Operation only applies to secondary processors.
                           ! The module at node 1 is not a
                           ! secondary processor.
```

4.
```
>>> UPD E                  ! Update the module at node E.
?004E Specified node is not a processor.
                           ! System error message indicates
                           ! that node E houses a memory or
                           ! adapter module.
```

The UPDATE command syntax is:

```
UPD[ATE] <node number>   -or-
UPD[ATE] ALL
```

where <node number> is the node number of the secondary processor that is to receive the contents of the primary processor's EEPROM. When UPDATE is issued, the console program checks the ROM revision levels of the processors. If the ROM revision level of a secondary processor does not match that of the primary, no update is done and a message is displayed.

The secondary processor's EEPROM is updated even if the processor is set to NOENABLED. The UPDATE ALL command updates the EEPROMs on all the secondary processors; the operation takes approximately 5 minutes for each secondary processor.

The UPDATE command copies the parameters that can be set plus some additional information stored in the EEPROM of the boot processor. Whenever the EEPROM on the primary processor has a patch level installed, the change should also be made in any secondary processors. Updated information includes:

- Systemwide console parameters
  Baud rate, interleave, and terminal characteristics
- Saved boot specifications
- Diagnostic patches
- Console patches
- Boot primitives

You must set the control panel lower key switch to the Update position to use this command. If the key switch is not in the Update position when you issue an UPDATE command, you receive the following error message:

```
?0040 Key switch must be at "Update" to update EEPROM.
```

NOTE: *An alternative to the UPDATE command is the EVUCA utility, which also updates a processor's EEPROM contents. See Appendix E.*

5.25 Z

The Z command logically connects the console terminal to another node on the XMI. Characters typed at the console terminal following a Z command are passed to the target node. All output from the target node is displayed on the console terminal.

Examples

1.
```
                         ! Connect from the boot processor at
>>> Z 6                  ! node 1 to CPU at node 6.
?0033 Z connection successfully started.
6>>                      ! Prompt indicates the new processor node no.
6>> D/P 0 12345678       ! Deposits data 12345678 to
                         ! physical address 0.
                         !
6>> ^P                   ! End connection to CPU at node 6.
?0031 Z connection terminated by ^P.
>>>                      ! Prompt indicates you are in console
                         ! mode on the boot processor.
>>> E/P 0
  P 00000000  12345678 ! Examines physical address 0; shows data
                         ! 12345678 was successfully deposited.
>>>
```

2.
```
>>> Z /BI:6 E            ! Connect to VAXBI node 6 through DWMBB in
                         ! XMI slot E.
?0033 Z connection successfully started.
6>>
```

Table 5–18: Z Command Qualifiers

Qualifier	Meaning
/B[I]:<VAXBI-node>	Specifies connection to the VAXBI at node number indicated by <VAXBI-node>.
<XMI-node>	Specifies connection to the XMI at node number indicated by <XMI-node>.

The Z command syntax is:

```
Z [/qualifier] <XMI-node>
```

where <XMI-node> is required and is the number of the target node. When used with the /BI qualifier, <VAXBI-node> specifies the target node on the VAXBI. When no qualifier is present, <XMI-node> specifies the target node on the XMI.

The Z command allows you to access the console program on a secondary processor directly. It also allows you to communicate with adapters that have ROM-based diagnostics.

Use a CTRL/P to terminate the Z command and return control to the primary processor. Use ESC to ignore any special functions of the character following the escape key. For example, ESC CTRL/P passes CTRL/P to the target node.

Only one Z command is in effect at a time. You cannot issue a Z command from one secondary processor to another secondary processor. Z commands can be issued only from the boot processor. A processor node that is already the target of a Z command rejects any Z commands that it receives and issues an error message. This way your characters are not being forwarded to more than one target node, but only to the node you have specified.

Once you have issued a Z command to access a secondary processor and you wish to access another secondary processor, you must issue a CTRL/P to return to the primary processor, and from there issue a Z command to the new secondary processor you wish to access.

When you access a secondary processor, the system modifies the console prompt to include the secondary processor node number (n>>). This modified prompt remains until you use a CTRL/P command to return to the primary processor.

5.26 !

The ! command introduces a comment. The console program ignores anything you enter on the command line following the !. The ! command is useful for documenting your console session on a hardcopy terminal for later reference.

Examples

1.
```
>>> ! THIS IS A COMMENT
>>>
```

2.
```
>>> SET TERM/SCOPE   ! THIS IS A COMMENT ON A COMMAND LINE
>>>
```

The ! command syntax is:

```
! [<comment entered here>]
```

You terminate the comment with a carriage return. If you want to enter several lines of comment, begin each new line with a ! command.

If your comment line exceeds 80 characters, you receive the error message:

```
?0036 Command too long.
```

5.27 Sample Console Session

```
#123456789 0123456789 0123456789 0123456789 012345#    ❶

F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0   NODE #
                                         ❷
    A   A   .   .   M   M   M   .   M   V-  -P  P   P   P       TYP
    +   +   .   .   +   +   +   .   +   +   +   +   +   +       STF
    .   .   .   .   .   .   .   .   .   E   E   E   E   B       BPD
    .   .   .   .   .   .   .   .   .   +   +   +   +   +       ETF
    .   .   .   .   .   .   .   .   .   E   E   E   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  SN = SG01234567

>>> EX/N:3 R0  ❸
    G 00000000  FFFFFFFF
    G 00000001  E0140648
    G 00000002  00000000
    G 00000003  00000010

>>> SHOW CONFIGURATION  ❹

     Type           Rev
 1+  KA65A   (8080) 0006
 2+  KA65A   (8080) 0006
 3+  KA65A   (8080) 0006
 4+  KA65A   (8080) 0006
 5+  FV64A   (0000) 0001
 6+  MS65A   (4001) 0084
 8+  MS65A   (4001) 0084
 9+  MS65A   (4001) 0084
 A+  MS65A   (4001) 0084
 D+  CIXCD   (0C05) 1652
 E+  DEMNA   (0C03) 0601

>>> BOOT /XMI:D/R5:70000000/NODE:0E02 DU0  ❺
      [self-test results appear]

* Initializing adapter  ❻
* Specified adapter initialized successfully
* Connecting to boot disk
?0018 Specified unit offline - Unit unknown, online to
another controller or port disabled via A, B switches
?0006 Halt instruction executed in kernel mode.
PC  = E00C2329
PSL = 041F0600
ISP = 000002F0

Bootstrap failed due to previous error.

>>> BOOT /XMI:D/R5:70000000/NODE:0E02 DU0  ❼

      [self-test results appear, then the operating system banner]
```

Sections of the sample console session flagged by the numbered callouts are explained below.

❶ At power-up, the system performs self-test and displays the results. See Section 6.2 for an explanation of self-test.

❷ The TYP line in the sample self-test display indicates that XMI slot 5 is a vector processor attached to the scalar processor at node 4.[1] The dashed lines indicate that the vector processor (V-) and the scalar processor (-P) are paired.

❸ The console prompt indicates that the terminal is in console mode. Enter an EXAMINE command to examine the contents of register 0 and three additional registers. Output displays the contents for R0, R1, R2, and R3, respectively.

❹ Enter a SHOW CONFIGURATION command to show the hardware configuration. Operator looks at configuration to find the disk controller to know the correct qualifiers to enter with the BOOT command.

System response indicates devices' XMI node numbers, self-test status, device types, and contents of the revision register of the device.

❺ Enter BOOT command, using the CIXCD adapter at XMI node D to locate the disk (DU0), the boot device, in a VAXcluster. Boots from an HSC controller that is dual-ported at unit numbers 0E and 02 with a system root of SYS7.

❻ System issues status and error messages.[2] An error occurred during connection to the boot disk, so a halt instruction is executed. You may assume an error in specifying the device or in the device itself. In this case, there was no disk pack in the disk drive. Error is corrected.

❼ BOOT command is reissued. The operating system begins to boot and presents its banner to the console terminal.

---

[1] Vector processors are supported on Model 400 and 500 systems only.

[2] See Appendix J for a list of status and error messages for Model 500 and 600 systems. Appendix H lists messages common to Model 400 and higher systems; Appendix I lists Model 300 messages.

Chapter 6

System Self-Test and Troubleshooting

This chapter discusses the testing that the system performs and the record displayed at power-up and at a system reset. The chapter includes the following sections:

- Self-test overview
- Sample self-test display
- Self-test progress trace line
- Self-test lines NODE #, TYP, and STF
- Self-test lines BPD and ETF
- Self-test lines ILV and Mb
- Self-test identification line
- Sample self-test display with VAXBI adapter
- Troubleshooting during booting
- Forcing a boot processor

6.1 Self-Test Overview

The system provides a record of its testing in the console self-test display. The control panel Fault light and the module self-test LEDs also indicate success or failure.

Figure 6–1: Testing Sequence

```
 #123456789 0123456789 0123456789 0123456789 012345# ❶
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
    +   +   .   .   +   +   +   +   .   .   +   +   -   +       STF
    .   .   .   .   .   .   .   .   .   .   E   B   E   D       BPD    ❷
    .   .   .   .   .   .   .   .   .   .   +   -   -   +       ETF    ❸
    .   .   .   .   .   .   .   .   .   .   B   E   E   D       BPD    ❹

    .   .   .   .   B2  B1  A2  A1  .   .   .   .   .   .       ILV
    .   .   .   .   64  64  64  64  .   .   .   .   .   .       256 Mb

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

  >>>
```

msb-0701B-91

Following power-up and system reset, the system performs testing and displays the results on the console terminal. As testing begins, the red Fault light on the control panel lights (see Section 3.5).

Most of the modules in the XMI card cage have on-board ROM used for testing. The first indication that testing has begun on (Models 400 and higher only) is the printing of a line of numbers, which is the first line of the self-test display (see ❶ in Figure 6–1). This line, called the progress trace, is generated by a processor module in slot 1 of the XMI card cage.[1] Each number displayed reflects the successful completion of tests run by this processor. For Models 300 and 200 the start of self-test is indicated by the NODE # line in the self-test display.

All modules have yellow LEDs on the outer edge of the module that light when the module passes self-test. The LEDs on the modules can be viewed from the front of the cabinet when the cabinet door is open. If a module fails self-test, its yellow LED does not light.

This first round of testing completes, and the results are shown on the STF line. The boot processor is then determined. This processor then generates the results of testing to this point ❷. The testing takes 10 to 15 seconds.

Next, the processors run additional tests using the memory modules. In this extended testing the processor that had been designated as boot processor could fail, so the boot processor is again determined. Results are displayed on the ETF line ❸.

The status of the boot processor and secondary processors is then displayed on the second BPD line ❹.

The boot processor next configures memory and displays the configuration. At this point if any memory had a noncorrectable error, all memory addresses for that module are tested, so there could be a delay before the console prompt appears.

Note that it is the boot processor determined at ❹ that displays the lines after the first BPD line. The final line before the prompt is from the boot processor's EEPROM.

Each line of the self-test display is described in detail in the following sections of this chapter.

[1] If your system has a T2019 power regulator in the XMI card cage, the progress trace line is not displayed.

6.2 Sample Self-Test Display

The processor modules display the results of self-test. Results are printed on the console terminal, as shown in Figure 6–2.

Figure 6–2: Self-Test Results

```
 #123456789 0123456789 0123456789 0123456789 012345# ❶
F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0   NODE #  ❷

    A   A   .   .   M   M   M   M   .   .   P   P   P   P       TYP     ❸
    +   +   .   .   +   +   +   +   .   .   +   +   -   +       STF     ❹
    .   .   .   .   .   .   .   .   .   .   E   B   E   D       BPD     ❺
    .   .   .   .   .   .   .   .   .   .   +   -   -   +       ETF     ❻
    .   .   .   .   .   .   .   .   .   .   B   E   E   D       BPD

    .   .   .   .   B2  B1  A2  A1  .   .   .   .   .   .       ILV     ❼
    .   .   .   .   64  64  64  64  .   .   .   .   .   .       256 Mb  ❽

 Console = V1.00  RBDs = V1.00  EEPROM = 1.00/1.00  SN = SG01234567
❾                              ❿                   ⓫

 >>>

      ! Items with callout numbers are explained in this chapter.
```

msb-0701-91

The self-test printout in Figure 6–2 reflects the system configuration listed in Table 6–1. Each numbered item in the example is explained in Section 6.3 through Section 6.7. These sections assume the same system configuration, when discussing the printout information.

See Section 6.8 for a description of self-test results when a VAXBI adapter is part of the system configuration. A sample self-test display with a vector processor is shown in Section 5.27.

Table 6–1: System Configuration for Sample Self-Test

Module	XMI Node Number	Module Type
KA66A	1	Processor; disabled from being boot processor.
KA66A	2	Processor; fails first level of self-test.
KA66A	3	Processor; boot processor after first level of self-test, fails extended test.
KA66A	4	Processor; operating as boot processor.
MS65A	7	Memory (64 Mbytes); interleaved with memory at node 8 by a SET MEMORY console command.
MS65A	8	Memory (64 Mbytes); interleaved with memory at node 7 by a SET MEMORY console command.
MS65A	9	Memory (64 Mbytes); interleaved with memory at node A by a SET MEMORY console command.
MS65A	A	Memory (64 Mbytes); interleaved with memory at node 9 by a SET MEMORY console command.
CIXCD	D	I/O adapter; passes self-test.
DEMNA	E	I/O adapter; passes self-test.

6.3 Self-Test Progress Trace Line

A line of decimal numbers indicates the progress of self-test execution (Model 400 and higher systems).

Figure 6–3: Self-Test Results: Progress Trace

```
 #123456789 0123456789 0123456789 0123456789 012345# ❶
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
    +   +   .   .   +   +   +   +   .   .   +   +   -   +       STF
    .   .   .   .   .   .   .   .   .   .   E   B   E   D       BPD
    .   .   .   .   .   .   .   .   .   .   +   -   -   +       ETF
    .   .   .   .   .   .   .   .   .   .   B   E   E   D       BPD

    .   .   .   .   B2  B1  A2  A1  .   .   .   .   .   .       ILV
    .   .   .   .   64  64  64  64  .   .   .   .   .   .       256 Mb

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

msb-0701A-91

The self-test printout in Figure 6–3 reflects the system configuration listed in Table 6–1. The first line shown in Figure 6–3, if complete, shows that the CPU in slot 1 passed all testing. If the final # sign is missing, the last number shown is the number of the failing test. (The number of tests varies by model number of the system.) This line of numbers is displayed only by the processor in slot 1 — and only when this processor undergoes power-up or a system reset. This processor is not always the boot processor.

For example, when the system is reset or the INITIALIZE command is issued, the progress trace line might show:

```
#123456789 01234567           ! Test #17 (decimal) failed.
```

6.4 Self-Test Lines NODE #, TYP, and STF

The next three lines of the self-test printout provide the node number identification (NODE #), type of module (TYP), and self-test status (STF) for modules in the XMI card cage. For Models 300 and 200 the start of self-test is indicated by the NODE # line.

Figure 6–4: Self-Test Results: NODE #, TYP, and STF

```
 #123456789 0123456789 0123456789 0123456789 012345#
  F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0    NODE #  ❷

      A   A   .   .   M   M   M   M   .   .   P   P   P   P        TYP     ❸
      +   +   .   .   +   +   +   +   .   .   +   +   -   +        STF     ❹
      .   .   .   .   .   .   .   .   .   .   E   B   E   D        BPD
      .   .   .   .   .   .   .   .   .   .   +   -   -   +        ETF
      .   .   .   .   .   .   .   .   .   .   B   E   E   D        BPD

      .   .   .   .   B2  B1  A2  A1  .   .   .   .   .   .        ILV
      .   .   .   .   64  64  64  64  .   .   .   .   .   .        256 Mb

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

  >>>
```

msb-0701C-90

The system configuration being tested is discussed in Section 6.2. See Table 6–1.

❷ The NODE # line lists the node numbers on the XMI bus. The nodes on this line are numbered in hexadecimal and reflect the position of the XMI slots as you view the XMI from the front of the cabinet through the clear card cage door (see Figure 6–4).

XMI entries use slots 1 through E, while an optional VAXBI could have entries in slots 0 through F (see Section 6.8). The XMI has 14 slots, and the slot and node numbers are identical. The self-test printout reflects the physical position of the modules in the XMI card cage.

❸ The TYP line in the printout indicates the type of module at each XMI node:

- An I/O adapter (A)
- A scalar processor (P)
- A vector processor (V)
- A memory module (M)
- A period indicating that the slot is not populated or the module is not reporting and may be dead

❹ The STF line shows the results of self-test. This information is taken from the self-test fail bit in the XBER register of each module. The entries are:

+ (pass)
– (fail)
o (does not apply)

6.5 Self-Test Lines BPD and ETF

The fifth, sixth, and seventh lines of the self-test printout provide information on the processors and their boot processor designation (BPD) and the results of the extended test (ETF).

Figure 6–5: Self-Test Results: BPD and ETF

```
 #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
    +   +   .   .   +   +   +   +   .   .   +   +   -   +       STF
    .   .   .   .   .   .   .   .   .   .   E   B   E   D       BPD    5
    .   .   .   .   .   .   .   .   .   .   +   -   -   +       ETF    6
    .   .   .   .   .   .   .   .   .   .   B   E   E   D       BPD    5

    .   .   .   .   B2  B1  A2  A1  .   .   .   .   .   .       ILV
    .   .   .   .   64  64  64  64  .   .   .   .   .   .       256 Mb

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

  >>>
```

msb-0701D-91

The system configuration being tested is discussed in Section 6.2. See Table 6–1.

❺ The BPD line [1] indicates boot processor designation. When the system goes through self-test, the processor with the lowest ID number that passes self-test (STF line is +) becomes the boot processor, unless you intervene. Using the SET CPU command and its qualifiers, you can change the eligibility of the processors to become the boot processor (see Section 5.18.2).

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)

This BPD line is printed twice. After the first determination of the boot processor, the processors go through an extended test. Since it is possible for a processor to pass self-test (at line STF) and fail the extended test (at ETF), the processors again determine the boot processor following the extended test.

In Figure 6–5 the processor at node 3 was chosen boot processor. Then this processor failed the extended test, so the processor at node 4 was chosen boot processor.

❻ During the extended test (ETF) all processors run additional CPU tests involving memory. In Figure 6–5, results printed at this ETF line indicate:

- Two processors passed the extended test (+)
- Two processors failed the extended test (–)

[1] For vector processors, either an "E" or a "D" appears on both BPD lines. An "E" indicates that the vector processor is enabled for use with system software; a "D" indicates that the vector processor is disabled.

6.6 Self-Test Lines ILV and Mb

The ILV line details the interleaving of the memories, and the Mb line gives the Mbytes of each memory module and the total size of system memory.

Figure 6–6: Self-Test Results: ILV and Mb

```
 #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
    +   +   .   .   +   +   +   +   .   .   +   +   -   +        STF
    .   .   .   .   .   .   .   .   .   .   E   B   E   D        BPD
    .   .   .   .   .   .   .   .   .   .   +   -   -   +        ETF
    .   .   .   .   .   .   .   .   .   .   B   E   E   D        BPD

    .   .   .   .   B2  B1  A2  A1  .   .   .   .   .   .        ILV      7
    .   .   .   .   64  64  64  64  .   .   .   .   .   .        256 Mb   8
  Console = V1.00  RBDs = V1.00  EEPROM = 1.00/1.00  SN = SGO1234567

  >>>
```

msb-0701E-91

Passing Memory

The system configuration being tested is discussed in Section 6.2.

7 This ILV line contains a memory interleave value (ILV) for each memory. Each interleave set is indicated by a different letter.

In Figure 6–6, a SET MEMORY command was used to create two interleaved sets of two 64-Mbyte memories each (see Section 5.18.4). This is indicated by the memory modules at nodes 7 and 8 being in the first interleave set A. Memories at nodes 9 and A are in memory interleave set B. The SET MEMORY command was:

```
SET MEMORY /INTERLEAVE:(7+8, 9+A)
```

If the default interleave were set on this configuration, it would be one 4-way interleave (modules at nodes 7, 8, 9, and A):

```
>>> SET MEMORY /INTERLEAVE:DEFAULT
>>> INITIALIZE
>>> SHOW MEMORY
F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0   NODE #

    .   .   .   .   A4  A3  A2  A1  .   .   .   .   .   .        ILV
    .   .   .   .   64  64  64  64  .   .   .   .   .   .       256 Mb
    /INTERLEAVE:DEFAULT
```

❽ The line after the ILV line displays the size of each configured memory module and gives the total Mbytes of system memory. In Figure 6–6, the total is 256 Mbytes.

Failing Memory

When a memory module does not pass its self-test, the boot processor tests the memory and failing memory pages are noted. The console program then puts the failing module in an interleave set by itself and maintains the largest possible interleave set with the remaining modules. The failing module is included in the configuration, but the addresses that failed self-test are not used. If the memory at node A failed self-test, it would be included in the configuration, but would not be interleaved with node 9. A SHOW MEMORY command shows the interleave with a failing module at node A:

```
>>> SHOW MEMORY
F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0   NODE #

    .   .   .   .   C1  B1  A2  A1  .   .   .   .   .   .        ILV
    .   .   .   .   64  64  64  64  .   .   .   .   .   .       256 Mb
    /INTERLEAVE:(7+8, 9+A)
```

Note that the /INTERLEAVE line above displays the interleave set as it is stored in the EEPROM. The ILV line shows the configuration actually in effect, including any changes due to self-test failures or incorrect interleave lists.

To exclude a memory that is failing self-test, you use the SET MEMORY command, without designating the node you want to exclude. In this example, to exclude the memory at node A:

```
>>> SET MEMORY /INTERLEAVE:(7+8, 9)
>>> INITIALIZE
>>> 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  .   .   .   .   .   .       192 Mb
    /INTERLEAVE:(7+8, 9)
```

6.7 Self-Test Identification Line

The last line of the self-test printout gives the console ROM and RBD ROM version numbers, the EEPROM's version number and console patch level number, and the serial number of the machine.

Figure 6–7: Self-Test Results: Identification Line

```
 #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
    +   +   .   .   +   +   +   +   .   .   +   +   -   +       STF
    .   .   .   .   .   .   .   .   .   .   E   B   E   D       BPD
    .   .   .   .   .   .   .   .   .   .   +   -   -   +       ETF
    .   .   .   .   .   .   .   .   .   .   B   E   E   D       BPD

    .   .   .   .   B2  B1  A2  A1  .   .   .   .   .   .       ILV
    .   .   .   .   64  64  64  64  .   .   .   .   .   .       256 Mb
 Console = V1.00  RBDs = V1.00  EEPROM = 1.00/1.00  SN = SGO1234567
❾                               ❿                  ⓫
 >>>
```

msb-0701F-91

The information in the self-test identification line reflects what is in the boot processor's EEPROM.

The system configuration being tested is discussed in Section 6.2. See Table 6–1.

❾ The console and RBD information indicates the version of read-only memory that is installed on the processors. For Model 400 the console version is called ROM0; the RBD version is called ROM1. For Model 300 only one ROM version is displayed.

Each processor has a console ROM and an RBD ROM; each ROM has its own version. In Figure 6–7, all processors have version V1.00 ROM resident. All processors should run with the same level of ROM. If your processors have mixed levels of ROM, the ROM level of the primary processor is displayed here, and you receive an error message that your processors have different ROM levels. Contact your customer service engineer to fix the ROM levels.

❿ The EEPROM information gives the boot processor's version of EEPROM and the patch level. In Figure 6–7, the first number, 1.00, gives the version of the contents of the EEPROM, and the second number, 1.00, is the console patch level. If you run processors whose EEPROMs do not match, you will receive an error message. Contact your customer service engineer.

⓫ SN gives the system serial number. The system serial number is also on the cabinet.

6.8 Sample Self-Test Display with VAXBI Adapter

The self-test printout contains an additional line when an optional VAXBI adapter is part of the system configuration. The XBI line provides information on the node numbers and self-test status for modules in the VAXBI card cages, which are connected to the XMI through a DWMBB adapter.

Figure 6–8: Self-Test Results: TYP, STF, and XBI Lines

```
 #123456789 0123456789 0123456789 0123456789 012345#
F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0   NODE #

❶   A   A   .   .   M   M   M   M   .   .   P   P   P   P       TYP
❷   o   +   .   .   +   +   +   +   .   .   +   +   -   +       STF
    .   .   .   .   .   .   .   .   .   .   E   B   E   D       BPD
    .   .   .   .   .   .   .   .   .   .   +   -   -   +       ETF
    .   .   .   .   .   .   .   .   .   .   B   E   E   D       BPD
.   .   .   .   .   .   .   .   .   +   .   +   -   +   +   .   XBI E + ❸
    .   .   .   .   B2  B1  A2  A1  .   .   .   .   .   .       ILV
    .   .   .   .   64  64  64  64  .   .   .   .   .   .       256 Mb

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

  >>>
```

msb-0701G-91

The system configuration shown in Figure 6–8 contains a DWMBB/A in XMI slot E.

❶ The TYP line in this printout indicates that adapters in this configuration are in XMI slots D and E.

❷ Because the DWMBB adapter does not have a module-resident self-test, its entry for the STF line will always be "o".

❸ The test results for the DWMBB/A and DWMBB/B modules are indicated on the XBI line, at the far right. In this example, the DWMBB modules have passed self-test (**XBI E +**). The results of the VAXBI I/O adapter self-tests are shown in columns 0 through F, which stand for the VAXBI node numbers; in this configuration, node numbers 1, 2, 3, 4, and 6 are used. The adapter at node 3 failed its self-test.

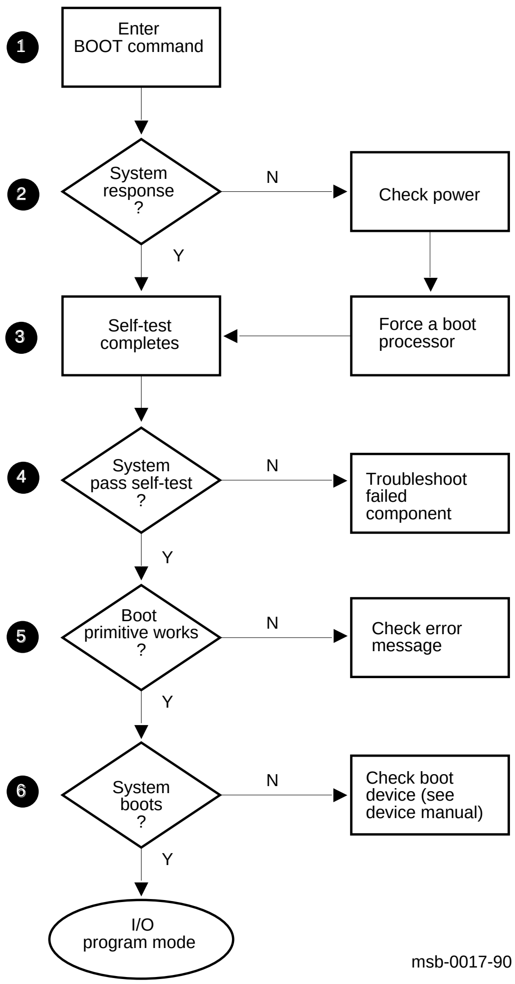
When a DWMBB passes self-test, each node on that VAXBI is indicated by symbols + and –, indicating the self-test status for that node number on the VAXBI. A period (.) indicates that that node number is not used. When a DWMBB fails self-test, the failure is reported, and the VAXBI device self-tests are not displayed.

Note that XMI entries use slots 1 through E, while the VAXBI can have entries in slots 0 through F. An XMI slot and node number are the same; VAXBI slot and node numbers are not identical. Node plugs (labeled 1 to 12) in the VAXBI backplane are used to identify the number of a node.

6.9 Troubleshooting During Booting

When booting fails, you can check several parameters.

Figure 6–9: Troubleshooting Booting



If you need to load a boot primitive before booting, see Appendix E. If the boot procedure fails, check through the steps shown in Figure 6–9.

1. Enter the BOOT command.
   Was the console terminal in console mode? If you are using a nickname (a stored BOOT command), did you use a valid nickname? You can check the nickname by using the SHOW command (see Section 5.19).

2. System response?
   If the system did not respond, check the power to the system. Turn the system off and on again. Check the power indicator lights (see Section 3.5) and console terminal connections. Check that the console terminal is in console mode.

   If the system still does not respond, try forcing a boot processor (see Section 6.10).

3. Self-test completes.
   You receive self-test printout. See Section 6.2 through Section 6.7 for a full explanation of the self-test results.

4. System pass self-test?
   If the system did not pass self-test, identify the modules that failed. If the failed module is your designated boot processor and your terminal is in console mode, use the SET CPU /NOPRIMARY command to reassign the boot processor, and reboot.

   If all system modules passed self-test, check your boot primitive.

5. Boot primitive works?
   If the boot primitive is working, then the boot primitive program reads the bootblock into memory and the system should boot. If the boot primitive fails, the boot primitive program is not able to reach the boot device. The console program displays several error messages. (Error messages are explained in Appendix H and Appendix I.) Check to see if the boot device passed self-test. Is the boot device connected to the system? Is it powered up?

6. System boots?
   If the system does not boot, check the boot device for malfunctioning. See the disk or tape drive manual for the specific boot device. There may also be a software failure. Check the boot device to be certain that the bootblock is on the boot device.

If these steps fail, call your customer service engineer.

6.10 Forcing a Boot Processor

The system may hang either because it cannot designate a processor to be the boot processor, or because none of the processors can find enough memory. When the system is hung, the console does not respond. After you check electrical and control panel connections, force a boot processor.

Example 6–1: Forcing a Boot Processor

```
#123456789 0123456789 0123456   ! Processor in slot 1 failed self-test
                                ! at test 26. System hangs.

[ >>3 ]   ! User enters ">>3" (not echoed), which forces the processor
          ! at node 3 to become the boot processor.  System runs
          ! self-test and prints results.

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
    +   +   .   .   +   +   +   +   .   .   .   +   +   -        STF
    .   .   .   .   .   .   .   .   .   .   .   B   D   E        BPD
    .   .   .   .   .   .   .   .   .   .   .   +   +   -        ETF
    .   .   .   .   .   .   .   .   .   .   .   B   D   E        BPD

    .   .   .   .   A4  A3  A2  A1  .   .   .   .   .   .        ILV
    .   .   .   .   64  64  64  64  .   .   .   .   .   .       256 Mb

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

>>>     ! Console prompt appears. You are now in console mode.
        ! Not all console commands are available to you following
        ! the forcing of a boot processor.
```

If self-test fails after power-up or a system reset, the system may be hung so that you cannot get a console prompt. The system hangs during the boot process for one of two reasons:

- No boot processor can be found. Processors either are disabled from becoming the boot processor or they fail self-test.
- No memory can be located.

Example 6–1 shows a case where no boot processor could be found. Although there are three processors, two of the processors (at nodes 2 and 3) were made ineligible to become the boot processor (with the SET CPU command and qualifiers). The third processor at node 1 is eligible to be a boot processor, but it failed self-test. The system cannot find a boot processor and hangs.

If the system hangs, after 60 seconds you can use the console program to intervene. You can force a processor to become the boot processor and override any previous commands by typing:

```
>>n
```

where *n* is the XMI node number of a processor. The console program then makes this the boot processor. Self-test begins and processor *n* displays the results of self-test. In Example 6–1 the processor at node 3 was forced to become the boot processor. The >>*n* sequence is not echoed; you may have to type this more than once to get to the console prompt.

Now you can examine the status of the processors by using the SHOW CPU command. Self-test results do not give a true picture of the processor status (BPD line), because you forced one of the processors to become the boot processor. Only the SHOW CPU command gives the setting in the EEPROM.

In Example 6–1, a next console action might be to SET CPU /PRIMARY for the processors at nodes 2 and 3 to avoid repeating a processor lock. Not all console commands are available to the system when you force a boot processor, so it is useful to correct the cause of the system hang.

When a system hang is caused by inability to locate sufficient memory, the processors cannot find memory as they attempt to run the extended test. Partial self-test results are displayed and the following message:

```
?0047 Insufficient working memory for normal operation.
```

The system hangs here until the >>*n* command is entered. When a system hang is caused by a memory problem and you force a boot processor, self-test results show that either all memory failed self-test or that no good memory exists. Check the installation of your memories and restart the system.

Appendix A
Compact Disk Drive Instructions

The Ethernet-based InfoServer can house one or two RRD compact disk drives. This appendix describes the RRD compact disk drive.

A.1 Controls and Indicators

The RRD compact disk drive has a green power light and a green activity light. Table A–1 lists the functions of the lights shown in Figure A–1.

Figure A–1: RRD Compact Disk Drive

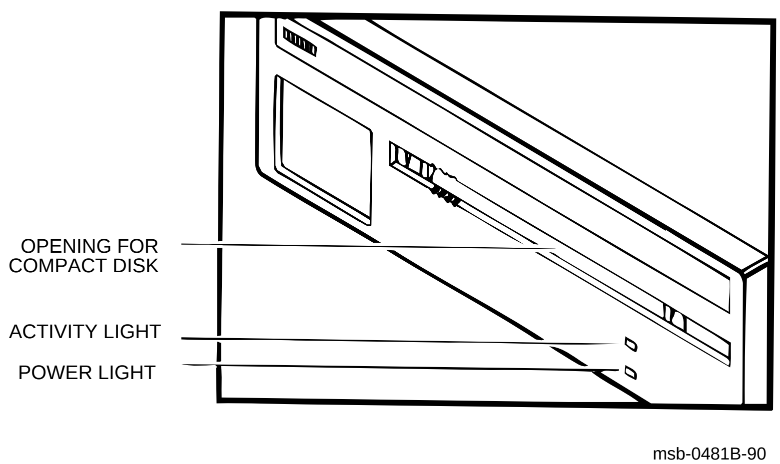


Table A–1: RRD Light Summary

Light	State	Condition
Green (Power)	Off	No power to drive
	On	Power to drive
Green (Activity)	Off	No disk in drive
	On	Disk is properly inserted into drive
	Blinking	Data is being transferred

A.2 Loading a Compact Disk

To load a disk, follow these steps:

1. Make sure the power light is on.
2. Insert the disk caddy into the drive, as shown in Figure A–2. Make sure that the disk label is facing up and that the notches on the left side of the caddy line up with the notches on the drive door.
3. Slide the caddy in as far as it will go and then remove it. The disk and its housing remain in the drive.
4. The activity light should go on within 5 seconds.

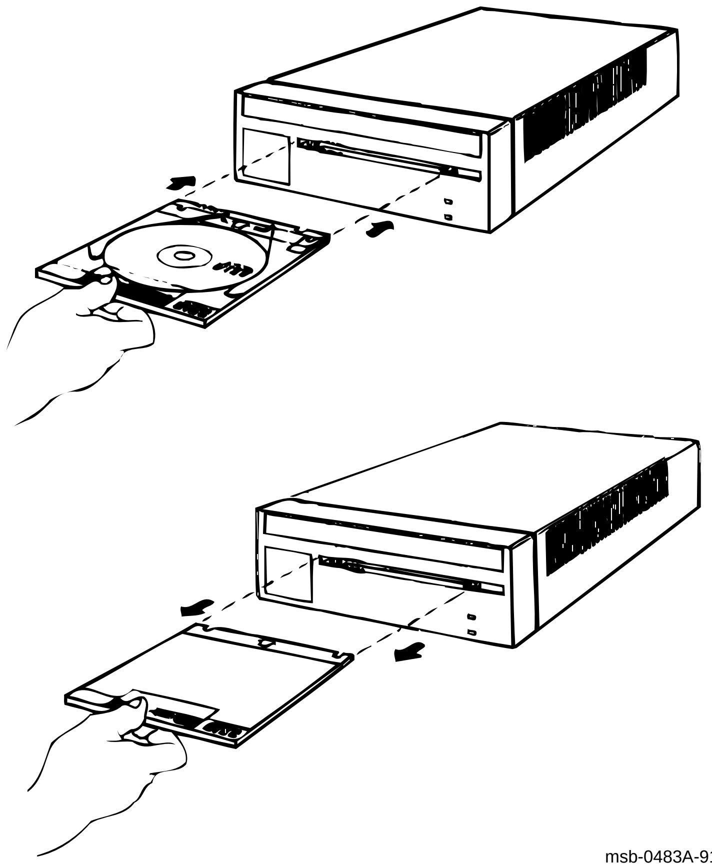
   If the activity light does not go on, the disk may be inserted incorrectly. Unload the disk and then reload. If you think the disk is loaded correctly, but the green light does not go on, try loading another disk into the drive. If the new disk loads successfully, then the original disk may be defective. Call your Digital customer service engineer.

A.3 Unloading a Compact Disk

Before unloading a disk, check to see that the disk is not transferring data (activity light is not blinking). Do not unload if the disk is transferring data. To unload a disk:

1. Position the transparent sleeve by making sure that the arrow on the sleeve is pointing toward the drive.
2. Insert the sleeve all the way into the drive.
3. Remove the caddy. The disk and caddy will be in the sleeve. The activity light goes off.

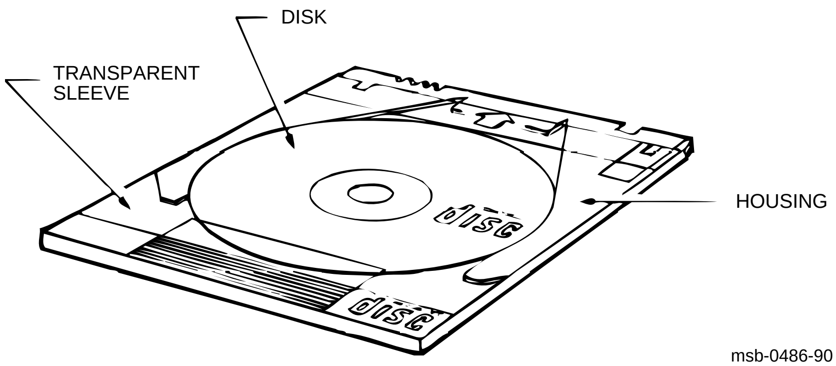
Figure A–2: Loading a Compact Disk



A.4 Cleaning Disks

The disk caddy consists of the disk, the disk housing, and the transparent sleeve. The caddy should be taken apart only if the transparent sleeve is damaged or if the disk requires cleaning. See the *RRD42 Disc Drive Owner's Manual* for cleaning instructions.

Figure A–3: Disk Caddy Parts



Appendix B
TF/TK Tape Drive Instructions

The tape drive holds one tape cartridge that contains the magnetic tape on a single reel. When a tape cartridge is inserted, the tape is automatically threaded onto a reel inside the drive. The tape must be entirely rewound before the cartridge can be removed from the drive. Rewinding can take up to 90 seconds.

The TF85 and TK70 can read data from a tape that was written by a TK50, but they cannot overwrite a tape originally written by a TK50. A TK50, however, cannot read data from a tape written by a TF85 or TK70.

B.1 Controls and Indicators

The tape drive has lights that indicate device operation, a beeper, an unload button, and a cartridge insert/release handle. Table B–1 lists the functions of TF85 tape drive controls and indicators that are shown in Figure B–1.

Figure B–1: TF85 Tape Drive

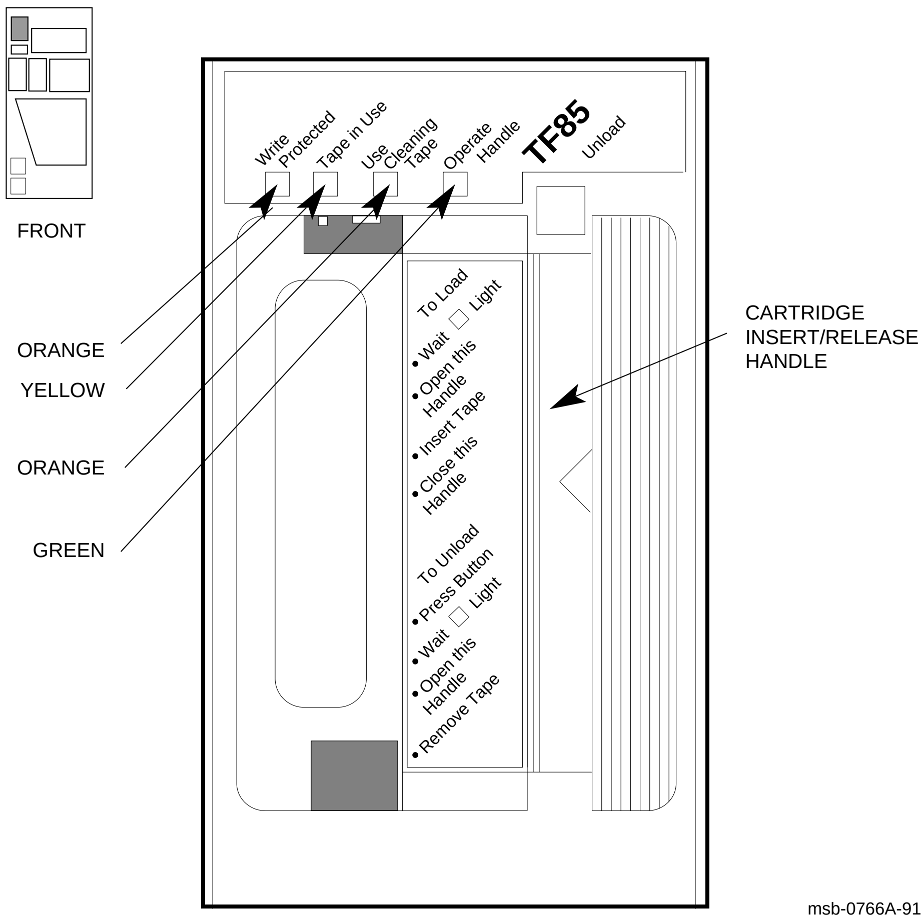


Table B–1: TF85 Light Summary

Light	State	Condition
Green	On	OK to operate handle.
(Operate Handle)	Off	Do not operate handle.
	Blinking	Defective cartridge. Pull the handle to the open position and remove cartridge. Try another cartridge.
Yellow	Steady	Drive ready.
(Tape in Use)	Blinking	Drive in use.
Orange [1]	On	Tape write protected.
(Write Protected)	Off	Tape write enabled.
Orange	On	Drive needs cleaning.
(Use Cleaning Tape)	Off	Drive cleaning unnecessary.
All four lights	Blinking	Drive fault. Attempt to reset the fault by pressing the unload button.

[1]This orange light is on when any of the following conditions exist:
- Cartridge write protect switch is in the protected position.
- Cartridge is software write protected.
- Attempt was made to mount or initialize a cartridge previously written in a TK50 drive.

B.2 Loading a Tape

To load a tape, follow these steps:

1. When the green light is on steadily, pull the handle to the open position.
2. With the label facing out, insert the tape cartridge.
3. Push the cartridge in until it is completely inside the drive.
4. Push the handle to the closed position. The yellow light blinks, indicating that the tape is loading. When the yellow light stays on steadily, the drive is ready for use.

NOTE: *If the green light blinks or if all lights blink, the loading has failed.*

B.3 Unloading a Tape

To unload a tape, follow these steps:

1. Press the unload button or execute an appropriate operating system unload command. The yellow light blinks as the tape rewinds.
2. When the green light turns on and the beep sounds, pull the handle to the open position. The cartridge will partially eject.
3. Remove the cartridge.
4. Push the handle to the closed position.

NOTE: *If all lights blink, the unload has failed.*

B.4 Write-Protecting Your Tape Cartridge

Write-protecting a tape cartridge prevents accidental erasure of information stored on the tape. To write-protect a tape, slide the tape's write-protect switch to the left so that the small orange rectangle is visible, as shown in Figure B–2.

Figure B–2: Tape Cartridge

B.5 Labeling a Tape Cartridge

To label your tape cartridge:

- Write your identifying information on the label. Note the recording density: CompacTape III = 2.6 Gbytes.
- Put the label into the slot on the front of the cartridge. See Figure B–2.
- Use only the labels supplied with the tape cartridge. Stick-on labels applied to the top, bottom, or sides of the cartridge can loosen and jam or damage the tape drive.
- Write only on the label. Do *not* write on the tape cartridge with a pen or pencil.

B.6 Tape Handling and Storage Guidelines

To add life to your tapes and protect data, follow these guidelines:

- Do not drop or bang the cartridge.
- Store tape cartridges upright in a dust-free environment.
- Keep tape cartridges away from direct sunlight, heaters, and other sources of heat. Store tape cartridges in an even temperature between 10^{o} and 40^{o}C (50^{o} to 104^{o}F).
- Keep tape cartridges away from sources of electromagnetic interference, such as terminals, motors, and video or X-ray equipment.

Appendix C

Device Type Code Assignments

Table C–1 lists XMI device type codes. See Appendix D for VAXBI device type codes. Device type code assignments are shown in the output of the SHOW CONFIGURATION console command (see Section 5.19).

Table C–1: XMI Device Type Code Assignments

Code	Device	Function
2001	DWMBA/A	XMI-to-VAXBI adapter
2002	DWMBB/A	XMI-to-VAXBI adapter
2002	DWMVA/A	XMI-to-VMEbus adapter
4001	MS62A	Memory
4001	MS65A	Memory
8001	KA62A	Processor
8001	KA62B	Processor
8080	KA65A	Processor
8082	KA64A	Processor
8087	KA66A	Processor
0C03	DEMNA	XMI-to-NI adapter
0C05	CIXCD	XMI-to-CI adapter
0C22	KDM70	RA disk and TA tape interface
0810	KFMSA	RF disk and TF tape interface
0823	DEMFA	XMI-to-FDDI adapter

Appendix D

VAXBI Options and Adapters

This appendix describes VAXBI options and adapters available with a VAX 6000 series system.

VAXBI options are supported by the XMI-to-VAXBI adapter, called the DWMBB adapter. The DWMBB adapter maps data between the XMI and VAXBI buses. The VAXBI, in turn, passes data between the system and peripheral devices. Up to six DWMBB adapters are supported.

D.1 Supported VAXBI Adapters

Table D–1 lists some of the VAXBI devices supported by VAX 6000 series systems. Note that some VAXBI adapters have more than one module, requiring more than one slot on the VAXBI.

Table D–1: VAXBI Adapters

Adapter	No. Slots	Device Code	Function
CIBCA	2	0108	CI port interface; connects a system to a Star Coupler.
DEBNA	1	410F	Ethernet port interface; connects a system to the Ethernet.
DEBNI	1	0118	Ethernet port interface; connects a system to the Ethernet.
DHB32	1	0109	Communication device; supports up to 16 terminals.
DMB32	1	0109	Interface for 8-channel asynchronous communications for terminals, one synchronous channel, and a parallel port for a line printer.
DRB32	1 or 2	0101	Parallel port.
DSB32	1	010A	Two-channel synchronous communication device.
DWMBB/B	1	210F	VAXBI-to-XMI interface.

Table D–1 (Cont.): VAXBI Adapters

Adapter	No. Slots	Device Code	Function
KDB50	2	010E	DSA disk adapter; enables connection to disk drives.
RBV20/ RBV64	1	0103	Write-once optical drive controller; uses the KLESI–B.
TBK50	1	410E	TK50 tape drive controller; connects the TK to the system.
TBK70	1	410B	TK70 tape drive controller; connects the TK to the system.
TM32	2	011F	Gapless tape controller.
TU81E	1	0103	TU81E controller; local (nonclustered) tape subsystem; uses the KLESI–B.

See Digital's *Systems and Options Catalog* or the *VAXBI Options Handbook* for more information on VAXBI adapters.

D.2 Supported Boot Devices

Table D–2 lists the boot devices supported by the VAXBI.

Table D–2: VAXBI Boot Devices

Device	Location
CI disk	Disk located on system's HSC controller connected to the system by the CIBCA adapter on the VAXBI.
Ethernet disk	Disk connected to the system over the Ethernet, through the VAXBI and DEBNI or DEBNA Ethernet port interface.
Local tape	TK50 or TK70 tape drive used as console load device; used for booting standalone backup.
Local disk	Disk connected to the system through the VAXBI; regular boot procedure specifies this disk as default boot device.

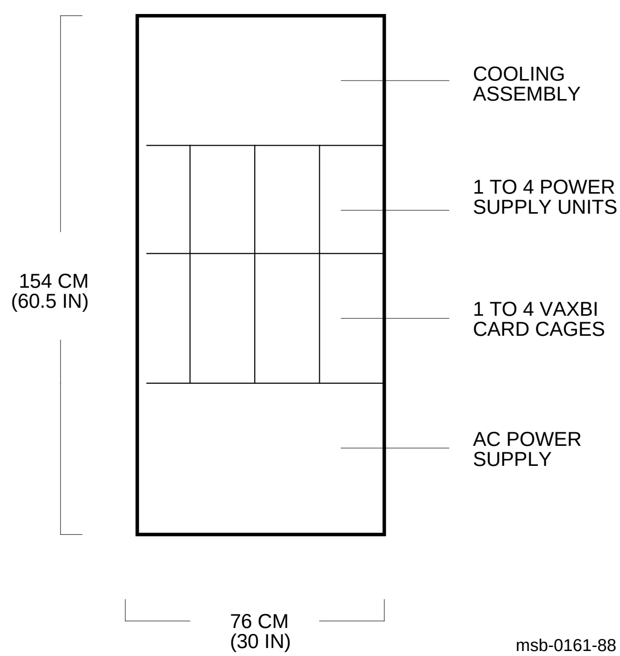
D.3 VAXBI Expander Cabinet

A VAXBI expander cabinet (Figure D–1) allows you to attach additional VAXBI channels, each with its required DWMBB/B.

The cabinet holds one to four VAXBI card cages, each with its own power supply. Two blowers cool the cabinet, and an AC power controller completes the power system.

For instructions on installing the VAXBI expander cabinet, see the *VAXBI Expander Cabinet Installation Guide* or the *VAX 6000 Series Installation Guide*.

Figure D–1: VAXBI Expander Cabinet



D.4 Power for the VAXBI Option

Table D–3 lists the system power available for the 12-slot in-cabinet VAXBI card cage.

Table D–3: In-Cabinet VAXBI Power

DC Voltage	Available VAXBI Current	Note
+5V	130.0 A	Main logic
+5VBB	Connected to +5V	Not battery backed up
+12V	4.0 A	RS–232
–12V	2.4 A	RS–232
–5.2V	20.0 A	ECL logic
–2V	7.0 A	ECL logic

Appendix E

EVUCA Program

This appendix describes the EVUCA program, the VAX 6000 EEPROM update utility. EVUCA allows you to:

- Update a processor's EEPROM contents
- Load boot primitives

E.1 EVUCA Program Overview

The EVUCA program runs under the VAX Diagnostic Supervisor (VAX/DS) in console mode. Table E–1 lists the processor-specific data files used to update the EEPROM contents. Each data file consists of the latest patch file for each major version of the PROM and all loadable boot primitives. EEPROMs are also updated on secondary processors if they are selected and attached under VAX/DS.

Table E–1: EEPROM Update Data Files

Data File	Processor	Model Number
EXUCA.BIN	KA66A	600
EMUCA.BIN	KA65A	500
ERUCA.BIN	KA64A	400
ELUCB.BIN	KA62B	300
ELUCA.BIN	KA62A	200

The following conditions must be met before running EVUCA:

- All processors and memory modules must pass self-test.
- Each processor must have console and diagnostic ROMs at the same major revision level.

E.2 Updating EEPROM Contents

Example E–1 shows a sample EEPROM update of a Model 600 two-processor system. The boot processor is in slot 1 of the XMI card cage and the secondary processor is in slot 2.

❶ At the console prompt, boot VAX/DS.

❷ At the VAX/DS prompt, load the EVUCA program.

❸ Attach the boot processor.

❹ Attach the secondary processor.

❺ Issue SELECT ALL to select all devices in the system configuration. SET TRACE enables printing of test numbers and names when EVUCA runs.

❻ Issue the START command to begin the EVUCA program.

❼ In this example the EXUCA.BIN data file is loaded to update KA66A processors. The EVUCA program uses the data file that corresponds to the processor type (see Table E–1). Press Return to continue.

❽ The EEPROM updates are verified.

❾ Exit VAX/DS.

Example E–1: Updating EEPROM Contents

```
>>> BOOT /XMI:n /R5:10 DUxx  ❶

             [the VAX Diagnostic Supervisor banner appears]

DS> LOAD EVUCA  ❷

DS> ATTACH KA66A HUB KA0 1  ❸

DS> ATTACH KA66A HUB KA1 2  ❹

DS> SELECT ALL  ❺

DS> SET TRACE

DS> START  ❻

.. Program: EVUCA - VAX 6000 EEPROM Update Utility, revision 2.0, 4
tests, at 00:05:57.61.
Testing: _KA0 _KA1

Node 02, booting.
Test 2: Load data from media

Data file to be loaded? <EXUCA.BIN>  ❼   ! Model 600

Searching...
Load complete.

Update rev:   0200

Test 3: Determine typecodes updated

Test 4: Update EEPROM data
Creating new eeprom in memory.
Writing to EEPROM.
Verifying EEPROM on node 01
EEPROM rev:   0200

Verification complete on node 01  ❽
Secondaries are being updated, please wait maximum of 20 seconds.
Secondary CPU 02 Done.
Verification complete.

The primary was successfully updated.
Secondary CPU 02, was successfully updated.
.. End of run, 0 errors detected, pass count is 1,
    time is  1-NOV-1991 00:07:05.94

DS> EXIT  ❾

>>>
```

Appendix F

Control Flags for Booting

With the console BOOT command, you can control various phases of booting by setting bits in General Purpose Register R5:

```
BOOT /R5:n
```

where *n* is in hexadecimal notation. For example, to set bit 4 in R5 when booting, you would enter:

```
BOOT /R5:10
```

The R5 bit functions are defined by VMB and by the operating system. The value –1 in R5 is reserved for Digital.

Table F–1: R5 Bit Functions for VMS

Bit	Function
0	Conversational boot. The secondary bootstrap program, SYSBOOT, prompts you for system parameters at the console terminal.
1	Debug. If this flag bit is set, the operating system maps the code for the XDELTA debugger into the system page tables of the running operating system.
2	Initial breakpoint. If this flag bit is set, VMS executes a breakpoint (BPT) instruction early in the bootstrapping process.
3	Secondary boot from boot block. The secondary boot is a single 512-byte block whose logical block number is specified in General Purpose Register R4.
4	Boots the VAX Diagnostic Supervisor. The secondary loader is an image called DIAGBOOT.EXE.
5	Boot breakpoint. This stops the primary and secondary loaders with a breakpoint (BPT) instruction before testing memory.

Table F–1 (Cont.): R5 Bit Functions for VMS

Bit	Function
6	Image header. The transfer address of the secondary loader image comes from the image header for that file. If this flag is not set, control shifts to the first byte of the secondary loader.
8	File name. VMB prompts for the name of a secondary loader.
9	Halt before transfer. VMB executes a HALT instruction before transferring control to the secondary loader.
13	No effect, since console program tests memory.
15	Reserved for the VAX Diagnostic Supervisor.
16	Do not discard CRD pages.
31:28	Specifies the top-level directory number for system disks.

Table F–2: R5 Bit Functions for ULTRIX

Bit	Function
0	Forces ULTRIXBOOT to prompt the user for an image name (the default is VMUNIX).
1	Boots the ULTRIX kernel image in single-user mode.
3	Must be set, and R4 must be zero.
16	Must be set.

Appendix G

Console Commands

Table G–1 gives a summary of the console commands. Chapter 5 gives a full description of each command, its qualifiers, and examples.

Table G–1: Console Commands and Qualifiers

Command and Qualifiers	Function
BOOT /R3:n /R5:n /XMI:n /BI:m /NODE:n /FILENAME:xyz /DSSI_NODE:n /PORT:x	Initializes the system, causing a self-test, and begins the boot program.
CLEAR EXCEPTION	Cleans up error state in XBER, XBEER, and CPU-specific registers.
CONTINUE	Begins processing at the address where processing was interrupted by a CTRL/P console command.
DEPOSIT /B /G /I /L /M /N /P /Q /V /VE /W	Stores data in a specified address.
EXAMINE /B /G /I /L /M /N /P /Q /V /VE /W	Displays the contents of a specified address.
FIND /MEMORY /RPB	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.

Table G–1 (Cont.): Console Commands and Qualifiers

Command and Qualifiers	Function
INITIALIZE [n] /BI:n	Performs a system reset, including self-test.
REPEAT	Executes the command passed as its argument.
SET BOOT	Stores a boot command by a nickname.
SET CPU [n] /ENABLED /ALL /NOENABLED /NEXT_PRIMARY /PRIMARY /ALL /NOPRIMARY /VECTOR_ENABLED /NOVECTOR_ENABLED	Specifies eligibility of processors to become the boot processor or disables a vector processor.[1]
SET LANGUAGE ENGLISH INTERNATIONAL	Changes the output of the console error messages between numeric code only (international mode) and code plus explanation (English mode).
SET MEMORY /CONSOLE_LIMIT:n /INTERLEAVE:(n+n...) /INTERLEAVE:DEFAULT /INTERLEAVE:NONE	Designates the method of interleaving the memory modules; supersedes the console program's default interleaving.
SET TERMINAL /BREAK /NOBREAK /HARDCOPY /NOHARDCOPY /SCOPE /NOSCOPE /SPEED:n	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.

[1]Vector processors are supported only on Model 400 and 500 systems.

Table G–1 (Cont.): Console Commands and Qualifiers

Command and Qualifiers	Function
SHOW DSSI	Displays DSSI bus numbers, node numbers, and unit numbers.
SHOW ETHERNET	Displays Ethernet hardware addresses for all Ethernet adapters on the system and FDDI hardware addresses for all FDDI adapters.
SHOW FIELD[2]	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 /BI:n	Halts the specified node.
TEST /RBD	Passes control to the self-test diagnostics.
UPDATE	Copies contents of the EEPROM on the processor executing the command to the EEPROM of another processor.
Z /BI:n	Logically connects the console terminal to another processor on the XMI bus or to a VAXBI node.
!	Introduces a comment.

[2]The SHOW FIELD command is available only on Model 500 and 600 systems.

Appendix H

Console Error Messages (Model 400 and Higher)

Table H–1 lists Model 400/500/600 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 H–2 lists other console error messages that are common to Model 400 and higher systems. Table H–3 lists console error messages that apply only to Models 500 and 600. Table H–4 lists console error messages unique to Model 600.

* On Model 400, system error messages and number codes appear as:
`?nn <message>`

Table H–1: Console Error Messages Indicating Halt (Model 400 and Higher)

Error Message	Meaning
?0002 External halt (CTRL/P, break, or external halt).	CTRL/P 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.
?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.

Table H–1 (Cont.): Console Error Messages Indicating Halt (Model 400 and Higher)

Error Message	Meaning
?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 H–2: Standard Console Error Messages (Model 400 and Higher)

Error Message	Meaning
?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.

Table H–2 (Cont.): Standard Console Error Messages (Model 400 and Higher)

Error Message	Meaning
?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).
?002D For Secondary Processor *n*.	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.

Table H–2 (Cont.): Standard Console Error Messages (Model 400 and Higher)

Error Message	Meaning
?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 Bad explicit interleave list — 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.
?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.

Table H–2 (Cont.): Standard Console Error Messages (Model 400 and Higher)

Error Message	Meaning
?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: *n n ...*	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.
?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.

Table H–2 (Cont.): Standard Console Error Messages (Model 400 and Higher)

Error Message	Meaning
?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 *x.xx*.	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.
?0054 EEPROM revision mismatch. Secondary processor has revision *x.xx/y.yy*.	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.

Table H–2 (Cont.): Standard Console Error Messages (Model 400 and Higher)

Error Message	Meaning
?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.
?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.

Table H–2 (Cont.): Standard Console Error Messages (Model 400 and Higher)

Error Message	Meaning
?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 = *x.xx/y.yy*.	The EEPROM contents are at revision *x.xx* with revision *y.yy* 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 = *x.xx/y.yy*.	The EEPROM image on the TK tape is at revision *x.xx* with revision *y.yy* patches.
?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.

Table H–2 (Cont.): Standard Console Error Messages (Model 400 and Higher)

Error Message	Meaning
?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 *n*	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 *n*	The I/O adapter at node *n* is configured improperly.
?007D Vector module is disabled—check KA64A revision at XMI node *n*	The vector module is attached to a KA64A module that is not at the revision level required.
?0083 Loading system software.[1]	The console is attempting to load the operating system in response to a BOOT command, power-up, or restart failure.
?0084 Failure.[1]	An operation did not complete successfully. Should be issued with another message to clarify failure.
?0085 Restarting system software.[1]	The console is attempting to restart the in-memory copy of the operating system following a power-up or serious error.

[1]No numbered prefix appears with these messages in English language mode. These numbers are used for these messages in International mode.

Table H–2 (Cont.): Standard Console Error Messages (Model 400 and Higher)

Error Message	Meaning
?00A0 Initializing system.[1]	The console is resetting the system in response to a BOOT command.
?00A1 Now updating the EEPROM of node *n*[1]	The console is updating the EEPROM.
?00A6 Console halting after unexpected machine check or exception.[1]	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.[1]	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.[1]	The previous attempt to bootstrap the system failed.
?00A9 Restart failed due to previous error.[1]	The previous attempt to restart the system failed.
Node *n: ?xxxx*	Error message *?xxxx* was generated on secondary processor *n* and was passed to the primary processor to be displayed.

[1] No numbered prefix appears with these messages in English language mode. These numbers are used for these messages in International mode.

Table H–3: Console Error Messages for Models 500 and 600

Error Message	Meaning
?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.	Model 500 CPU chip failed its microcoded self-test.
?011E Uncorrectable memory errors discovered - long memory test must be performed on node *n*	Memory array in node *n* contains an uncorrectable error. The console must perform a full test to locate all the failing locations.
?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 and higher.
?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.

Table H–4: Console Error Messages Unique to Model 600

?0201 One or more power-up tests have been bypassed.	A test normally run by the processor at power-up has been bypassed.
?0203 Hardware compatibility group mismatch—secondary/primary: *x/y*.	Indicates hardware version mismatches between the primary CPU and an indicated secondary CPU.
?0205 Error locating ROM boot code, run diagnostics.	Indicates a hardware problem reading the CPU's ROM code.
?0206 EEPROM in error or contains unsupported PCS, processor disabled.	Indicates an out of rev or faulty EEPROM image. Use the EVUCA program to upgrade the EEPROM for the indicated CPU.

Appendix I

Console Error Messages for Model 300

Table I–1 lists messages ?02 through ?1F which appear when the processor halts and the console gains control. Each message is followed by a "PC = xxxxxxxx" message giving the address where the processor was executing when it halted; these messages designate the reasons for the halt.

Table I–2 lists the standard console error messages ?20 through ?7C.

Table I–1: Model 300 Console Error Messages Indicating Halt

Error Message	Meaning
?02 External halt.	CTRL-P or STOP command.
?04 Interrupt stack not valid.	Interrupt stack pointer contained an invalid address.
?05 Machine check during exception.	A machine check occurred while handling another error condition.
?06 Halt instruction executed.	The CPU executed a Halt instruction.
?07 SCB vector bits <1:0> = 11.	An interrupt or exception vector in the System Control Block contained an invalid address.
?08 SCB vector bits <1:0> = 10.	An interrupt or exception vector in the System Control Block contained an invalid address.
?0A CHMx executed while on interrupt stack.	A change-mode instruction was issued while executing on the interrupt stack.
?0B CHMx to interrupt stack.	The System Control Block vector for a change-mode instruction indicated service on the interrupt stack.
?0C SCB read error.	The System Control Block was not accessible in memory.
?10 ACV or TNV during machine check.	An access violation or translation-not-valid error occurred while handling another error condition.

Table I–1 (Cont.): Model 300 Console Error Messages Indicating Halt

Error Message	Meaning
?11 ACV or TNV during KSP not valid.	An access violation or translation-not-valid error occurred while handling another error condition.
?12 Machine check during machine check.	A machine check occurred while handling another error condition.
?13 Machine check during KSP not valid.	A machine check occurred while handling another error condition.
?19 PSL <26:24>= 101 during interrupt or exception.	An exception or interrupt occurred while on the interrupt stack but not in kernel mode.
?1A PSL <26:24>= 110 during interrupt or exception.	An exception or interrupt occurred while on the interrupt stack but not in kernel mode.
?1B PSL <26:24>= 111 during interrupt or exception.	An exception or interrupt occurred while on the interrupt stack but not in kernel mode.
?1D 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.
?1E 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.
?1F 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 I–2: Model 300 Standard Console Error Messages

Error Message	Meaning
?20 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.
?21 Illegal command.	The command was not recognized, contained the wrong number of parameters, or contained unrecognized or inappropriate qualifiers.
?22 Illegal address.	The specified address was recognized as being invalid, for example, a general purpose register number greater than 15.
?23 Value is too large.	A parameter or qualifier value contained too many digits.
?24 Conflicting qualifiers.	A command specified recognized qualifiers that are illegal in combination.
?25 Checksum did not match.	The checksum calculated for a block of X command data did not match the checksum received.
?26 Halted.	The processor is currently halted.
?27 Item was not found.	The item requested in a FIND command could not be found.
?28 Timeout while waiting for characters.	The X command failed to receive a full block of data within the timeout period.
?29 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.
?2A Unexpected machine check or interrupt.	A valid operation within the console caused a machine check or interrupt.
?2B Command is not implemented.	The command is not implemented by this console.
?2C Unexpected exception.	A valid operation within the console caused an exception.

Table I–2 (Cont.): Model 300 Standard Console Error Messages

Error Message	Meaning
?2D For Secondary Processor *n*	This message is a preface to second message describing some error related to a secondary processor. This message indicates which secondary processor is involved.
?2E Specified node is not an I/O adapter.	The referenced node is incapable of performing I/O or did not pass its self-test.
?30 Write to Z command target has timed out.	The target node of the Z command is not responding.
?31 Z connection terminated by ^P.	A CTRL/P was typed on the keyboard to terminate a Z command.
?32 Your node is already part of a Z connection.	You cannot issue a Z command while executing a Z command.
?33 Z connection successfully started.	You have requested a Z connection to a valid node.
?34 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.
?36 Command too long.	The command length exceeds 80 characters.
?37 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.
?38 Waiting for a CR to terminate the command.	The command has not yet been issued by a carriage return.
?39 Console patches are not useable.	The console patch area in EEPROM is corrupted or contains a patch revision that is incompatible with the console ROM.
?3B Error encountered during I/O operation.	An I/O adapter returned an error status while the console boot primitive was performing I/O.

Table I–2 (Cont.): Model 300 Standard Console Error Messages

Error Message	Meaning
?3C 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.
?3D Error initializing I/O device.	A console boot primitive needed to perform I/O, but could not initialize the I/O adapter.
?3E Timeout while sending message to secondary.	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.
?3F Key switch must be at "Update" to update EEPROM.	A SET command needs to update the EEPROM, but the key switch is not set to allow updates.
?40 Specified node is not a bus adapter.	A command that accesses a VAXBI node specified an XMI node that was not a bus adapter.
?41 Invalid terminal speed.	The SET TERMINAL command specified an unsupported baud rate.
?42 Unable to initialize node.	The INITIALIZE command failed to reset the specified node.
?43 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.
?44 Unable to stop node.	The STOP command failed to halt the specified node.
?45 Memory interleave set is inconsistent: *n n ...*	The listed nodes do not form a valid memory interleave set. One or more of the nodes might not be a memory array, or the set could contain an invalid number of members. Each listed array that is a valid memory will be configured uninterleaved.

Table I–2 (Cont.): Model 300 Standard Console Error Messages

Error Message	Meaning
?46 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.
?47 Uncorrectable memory errors—long memory test must be performed.	A memory array contains an unrecoverable error. The console must perform a slow test to locate all the failing locations.
?49 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.
?4A Internal logic error in console.	The console encountered a theoretically impossible condition.
?4B 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.
?4C Invalid node for new primary.	The SET CPU command failed when attempting to make the specified node the primary processor.
?4D Specified node is not a processor.	The specified node is not a processor, as required by the command.
?4E System serial number has not been initialized.	No CPU in the system contains a valid system serial number.
?4F 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.
?50 Secondary processor returned bad response message.	A secondary processor returned an unintelligible response to a request made by the console on the primary processor.
?51 ROM revision mismatch. Secondary processor has revision *x.y*.	The revision of console ROM of a secondary processor does not match the primary.

Table I–2 (Cont.): Model 300 Standard Console Error Messages

Error Message	Meaning
?52 EEPROM header is corrupted.	The EEPROM header has been corrupted. The EEPROM must be restored from the TK tape drive.
?53 EEPROM revision mismatch. Secondary processor has revision *x.y/x.y*.	A secondary processor has a different revision of EEPROM or has a different set of EEPROM patches installed.
?54 Failed to locate EEPROM area.	The EEPROM did not contain a set of data required by the console. The EEPROM may be corrupted.
?55 Console parameters on secondary processor do not match primary.	Console parameters do not match on the primary and secondary processors.
?56 EEPROM area checksum error.	A portion of the EEPROM is corrupted. It may be necessary to reload the EEPROM from the TK tape drive.
?57 Saved boot specifications on secondary processor do not match primary.	Saved boot specifications do not match on the primary and secondary processors.
?58 Invalid unit number.	A BOOT or SET BOOT command specifies a unit number that is not a valid hexadecimal number between 0 and FF.
?59 System serial number mismatch. Secondary processor has xxxxxxxx.	The indicated serial number of a secondary processor does not match the primary.
?5A 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.
?5B No HELP is available.	The HELP command is not supported when the console language is set to International.
?5C No such boot spec found.	The specified saved boot specification was not found in the EEPROM.
?5D Saved boot spec table full.	The maximum number of saved boot specifications has already been stored.

Table I–2 (Cont.): Model 300 Standard Console Error Messages

Error Message	Meaning
?5E EEPROM header version mismatch.	The primary and a secondary processor have different versions of the EEPROM. The requested operation cannot be performed.
?5F Bad transfer length.	The primary processor attempted to send EEPROM data to a secondary processor using an invalid length.
?60 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.
?61 Illegal node number.	The specified node number is invalid.
?62 Unable to locate console tape device.	The console could not locate the I/O adapter that controls the TK tape.
?63 Operation only applies to secondary processors.	The command can only be directed at a secondary processor.
?64 Insufficient memory to buffer EEPROM image.	The SAVE, RESTORE, and PATCH EEPROM commands require working memory, but not enough was found.
?65 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.
?66 Read of EEPROM image from tape failed.	The EEPROM image was not successfully read from tape.
?67 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.
?68 EEPROM not changed.	The EEPROM contents were not changed.
?69 EEPROM changed successfully.	The EEPROM contents were successfully patched or restored.

Table I–2 (Cont.): Model 300 Standard Console Error Messages

Error Message	Meaning
?6A Error changing EEPROM.	An error occurred in writing to the EEPROM. The EEPROM contents may be corrupted.
?6B EEPROM saved to tape successfully.	The EEPROM contents were successfully written to the TK tape.
?6C EEPROM not saved to tape.	The EEPROM contents were not completely written to the TK tape.
?6D EEPROM Revision = *x.x/y.y*.	The EEPROM contents are at revision *x.x* with revision *y.y* patches.
?6E Major revision mismatch between tape image and EEPROM.	The TK tape contains an EEPROM image with a major revision different from that found in the EEPROM. The image cannot be restored.
?6F Tape image Revision = *x.x/y.y*.	The EEPROM image on the TK tape is at revision *x.x* with revision *y.y* patches.
?74	CONSOLE_LIMIT value too small for proper operation. Value ignored.
?75	Error writing to tape. Tape may be write-locked.
?83	See *Loading system software* below.
?84	See *Failure* below.
?85	See *Restarting system software* below.
?B0	See *Initializing system* below.
Failure.	The console failed in a restart or boot operation. Shows as ?84 in SET LANGUAGE INTERNATIONAL mode.
Initializing system.	The console is resetting the system in response to a BOOT command. Shows as ?B0 in SET LANGUAGE INTERNATIONAL mode.

Table I–2 (Cont.): Model 300 Standard Console Error Messages

Error Message	Meaning
Loading system software.	The console is attempting to load the operating system in response to a BOOT command, power-up, or restart failure. Shows as ?83 in SET LANGUAGE INTERNATIONAL mode.
Node: *n ?xx*	Error message *?xx* was generated on secondary processor *n* and was passed to the primary processor to be displayed.
Restarting system software.	The console is attempting to restart the in-memory copy of the operating system following a power-up or serious error. Shows as ?85 in SET LANGUAGE INTERNATIONAL mode.

Appendix J

Boot Status and Error Messages (Models 500 and 600)

This appendix lists status and error messages for Ethernet boots, local disk and tape boots, and cluster boots (Models 500 and 600). 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.

J.1 Ethernet Boot Messages

1. [Start boot]

    ?002E Specified node is not an I/O adapter
    ?0100 Specified adapter failed self-test
    ?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

    ?0113 No traffic was detected on the Ethernet—aborting boot procedure
    ?0115 Aborting boot process—adapter failed attempting to execute port command
    ?011F Aborting boot process—adapter failed attempting to execute boot command

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

J.2 Local Disk Boot Messages

1. [Start Boot]

   ?002E Specified node is not an I/O adapter
   ?0100 Specified adapter failed self-test
   ?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 *or*
   * Reading bootblock from disk

   ?0102 Controller error detected—aborting
   ?0103 Drive error detected—aborting
   ?010E Specified unit offline — No media mounted or disabled via RUN/STOP switch setting
   ?0114 Serious exception reported—aborting
   ?0116 Specified unit is inoperative
   ?0117 Specified unit offline
   ?0118 Specified unit offline—Unit unknown, online to another controller or port disabled via A,B switches

5. * Passing control to transfer address

J.3 Local Tape Boot Messages

1. [Start boot]

   ?002E Specified node is not an I/O adapter
   ?0100 Specified adapter failed self-test
   ?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 *or*
   * Reading bootblock from tape *or*

* Rewinding tape

?0101 BVP port error reported—aborting
?0102 Controller error detected—aborting
?0103 Drive error detected—aborting
?010E Specified unit offline—No media mounted or disabled via RUN/STOP switch setting
?0114 Serious exception reported—aborting
?0116 Specified unit is inoperative
?0117 Specified unit offline
?0118 Specified unit offline—Unit unknown, online to another controller or port disabled via A,B switches

5. * Passing control to transfer address

J.4 CI and DSSI Boot Messages

1. [Start boot]

    ?002E Specified node is not an I/O adapter
    ?0109 Illegal adapter specified for CI boot
    ?011A Illegal adapter specified for DSSI 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. * Previous operation failed—retrying DSSI boot
7. * Port received a "no path" error—retrying the init sequence

    ?0110 Port received a "no path" error after 6 retries—aborting the boot process

8. * Connecting to MSCP server layer
9. * Previous operation failed—retrying CI boot
10. * Connecting to boot disk *or*
    * Connecting to shadow unit—will fail over to physical after 6 attempts.

    ?0102 Controller error detected—aborting
    ?0103 Drive error detected—aborting

?010E Specified unit offline—No media mounted or disabled via RUN/STOP switch setting
?0114 Serious exception reported—aborting
?0116 Specified unit is inoperative
?0117 Specified unit offline
?0118 Specified unit offline—Unit unknown, online to another controller or port disabled via A,B switches

11. * Failure to connect to shadow unit—retrying on physical unit

12. * Reading bootblock from disk

?0102 Controller error detected—aborting
?0103 Drive error detected—aborting
?010E Specified unit offline—No media mounted or disabled via RUN/STOP switch setting
?0114 Serious exception reported—aborting
?0116 Specified unit is inoperative
?0117 Specified unit offline
?0118 Specified unit offline—Unit unknown, online to another controller or port disabled via A,B switches

13. * Passing control to transfer address

Glossary

Adapter

A node that interfaces other buses, communication lines, or peripheral devices to the 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 VAX 6000 series 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 CD-ROMs 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 where the processor is not running the operating system but allows a console terminal operator to communicate with nodes on the XMI bus.

DEBNI

VAXBI adapter; Ethernet port interface.

DEMFA

XMI adapter to the FDDI (fiber distributed data 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.

DSB32

VAXBI adapter communication device; provides two synchronous lines.

DSSI

Digital Storage System Interconnect. A Digital Storage Architecture interconnect used by the KFMSA adapter and RF and TF series integrated storage elements to transfer data and to communicate with each other.

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 VAX 6000 series system must have a DWMBB adapter.

Ethernet-based compact disk server

The RRD 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 RA disks; enables connection to disk drives.

KDM70

XMI adapter for RA disks; enables connection to disk drives.

KFMSA

XMI adapter for RF disks and TF tapes; enables connection to nodes on a DSSI bus. Each KFMSA adapter supports two DSSI buses.

ISE (integrated storage element)

All DSSI storage devices, such as RF disks and TF tapes, are ISEs.

Memory interleaving

Method to optimize memory access time; the VAX 6000 series 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 MS65A. Memory is a global resource equally accessible by all processors on the XMI. See also *MS65A*.

Module

A single 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").

MS65A

XMI memory array; a memory subsystem of the XMI. Memory is a global resource equally accessible by all processors on the 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 XMI node is a single module that occupies one of the 14 logical and physical slots on the 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 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 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.

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 commmand, 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 DWMBB adapters and of VAXBI nodes. See also *DWMBB*.

XMI

The 64-bit, high-speed system bus.

XMI Corner

The portion of an XMI module that connects to the backplane and provides an electrically identical interface for every XMI node.

Index

A

Airflow sensor, 2–15, 3–13
Architecture, 1–4

B

Battery backup unit, 2–9, 2–17
  location, 1–8, 1–10
  status indicator light, 3–11
Baud rate, 5–3, 5–8, 5–57
  synchronizing, 5–8
BOOT command, 5–12 to 5–15
  default, 4–7
  description, 5–15
  examples, 5–13
  nickname, 4–7
  parsing, 4–6
  qualifiers, 5–13
  storing, 4–7
  syntax, 5–15
Boot devices, 4–4, 4–5, D–2
Boot device selection, 4–8 to 4–9
Booting, 4–2 to 4–11
  and Star Couplers, 4–13
  bootblock, 4–3
  boot code, 4–11
  boot device, 4–3 to 4–5
    local disk, 4–5
    TK tape drive, 4–5
    VAXcluster disk, 4–5
  boot primitives, 4–3
  control flags for, F–1 to F–2
  Ethernet-based, 4–16 to 4–19
  flowchart, 4–2
  Model 500 and 600 boot error messages, J–1 to J–4

Booting (Cont.)
  Model 500 and 600 boot status messages, J–1 to J–4
  procedure, 4–3
  regular procedure, 4–6 to 4–7
  sample commands, 4–7
  saved boot specification, 4–7
  troubleshooting, 6–18 to 6–19
  troubleshooting flowchart, 6–18
  ULTRIX, 4–12
  VAXcluster, 4–12 to 4–15
  VMB, 4–3
Boot processor
  designation on self-test results, 6–11
  forcing a, 6–20 to 6–21
  selection of, 4–10 to 4–11, 6–3
  troubleshooting during booting, 6–19

C

CIBCA, D–2
Circuit breaker, 2–8, 3–5, 3–12 to 3–13
CIXCD, 4–3, 4–4, 4–14 to 4–15
Configuration, 1–3
Console, 5–1 to 5–73
  baud rate, 5–8
  CCA (console communications area), 5–3
  changing terminal characteristics, 5–9
  definition of, 5–2 to 5–3
  functions, 5–5
  load devices, 2–2
  mode, 5–6 to 5–7
    definition, 5–6

Console
- mode (Cont.)
  - entering, 5–6
  - exiting, 5–7
  - prompt, 5–3
- program, 4–11, 5–3
- sample session, 5–72 to 5–73
- terminal
  - baud rate, 5–3, 5–57
  - defaults, 5–57
  - description, 5–3
  - port, 5–3
  - suspending output, 5–9

Console commands
- !, 5–70 to 5–71
- abbreviation of, 5–10
- BOOT, 5–12 to 5–15
- CLEAR EXCEPTION, 5–16 to 5–17
- command recall, 5–10
- CONTINUE, 5–18 to 5–19
- control characters, 5–8 to 5–9
  - Break key, 5–9
  - Carriage return, 5–9
  - CTRL/C, 5–9
  - CTRL/O, 5–9
  - CTRL/P, 5–9
  - CTRL/Q, 5–9
  - CTRL/R, 5–9
  - CTRL/S, 5–9
  - CTRL/U, 5–9
  - Delete key, 5–9
  - Escape key, 5–9
- DEPOSIT, 5–20 to 5–23
- editing, 5–9
- EXAMINE, 5–24 to 5–27
- FIND, 5–28 to 5–29
- functions, 5–4
- HALT, 5–30 to 5–31
- HELP, 5–32 to 5–33
- INITIALIZE, 5–34 to 5–35
- list of, 5–1
- maximum length, 5–10
- numbers, 5–10
- qualifier placement, 5–10

Console commands (Cont.)
- REPEAT, 5–36 to 5–37
- RESTORE EEPROM, 5–38 to 5–39
- SAVE EEPROM, 5–40 to 5–41
- SET BOOT, 5–44 to 5–45
- SET commands, 5–42 to 5–55
- SET CPU, 5–46 to 5–49
  - and self-test designation, 6–11
  - and troubleshooting booting, 6–19
- SET LANGUAGE, 5–50 to 5–51
- SET MEMORY, 5–52 to 5–54
  - and self-test results, 6–13
- SET TERMINAL, 5–54 to 5–55
- SHOW, 5–56 to 5–57
- SHOW BOOT
  - and troubleshooting, 6–19
- SHOW CONFIGURATION
  - and self-test results, 6–17
- SHOW FIELD, 5–56 to 5–57
- START, 5–58 to 5–59
- STOP, 5–60 to 5–61
- summary chart, G–1
- suspending output, 5–9
- syntax, 5–10 to 5–11
- TEST, 5–62 to 5–63
- UNJAM, 5–64 to 5–65
- UPDATE, 5–66 to 5–67
- Z, 5–68 to 5–69

Console commands and qualifiers, G–1 to G–3

Console mode, 3–6

Control panel, 3–2 to 3–3
- keys, 3–2
- labels, 3–2 to 3–3
- location, 1–8
- lower key switch, 3–2, 3–6 to 3–7
- Restart button, 3–5, 3–8 to 3–9
- status indicator lights, 3–10 to 3–11
  - Battery light, 3–11
  - Fault light, 3–11
  - location, 3–10

Control panel
status indicator lights (Cont.)
Run light, 3–11
upper key switch, 3–2, 3–4 to 3–6
Auto Start position, 3–7
Enable position, 3–5
Halt position, 3–7
Off position, 3–5
Secure position, 3–5
Standby position, 3–5
Update position, 3–7
Cooling system, 2–14 to 2–15
airflow pattern, 2–14
airflow sensor, 2–15
blowers, 2–15
location, 1–8, 1–10
space requirements, 2–15
thermostat, 2–15

D

DEBNA, D–2
DEBNI, D–2
DEMFA
booting, 4–23
DEMNA
booting, 4–23
DEPOSIT command, 5–20 to 5–23
Device type codes, C–1
VAXBI, D–1
DHB32, D–2
Disk drives, 2–17
DMB32, D–2
DRB32, D–2
DSB32, D–2
DSSI, 2–7
DWMBB, D–2
self-test, 6–3
self-test results, 6–17
DWMBB/B, D–2
DWMVA adapter, 1–5

E

EEPROM
EEPROM (Cont.)
and self-test, 6–15
patch level, 6–15
restoring, 5–38 to 5–39
saving to tape, 5–40 to 5–41
updating, E–1 to E–2
version number, 6–15
Error messages
changing format, 5–50 to 5–51
console
Model 300, I–1 to I–10
Model 400 and higher, H–1 to H–12
format, 5–7
Ethernet-based compact disk server, 2–4 to 2–5
booting, 4–16 to 4–19
Ethernet booting, 4–16 to 4–27
Ethernet connector, 2–13
EVUCA program, 5–67, E–1 to E–2
EXAMINE command, 5–24 to 5–27

F

FV64A, 5–72

H

Hardcopy output, 5–55
HELP for console commands, 5–32 to 5–33

I

I/O bulkhead connections, 2–12 to 2–13
Ethernet port, 2–13
location, 2–13
panel, 2–13
terminal port, 2–13
tray, 2–13
Initializing the system, 5–35
Input voltages, 2–8
Interleaving
and self-test results, 6–13
SET MEMORY command, 5–52 to 5–54

Internationalizing error messages, 5–50 to 5–51

K

KDB50, D–2
KFMSA, 4–3
KLESI–B, D–2

M

Memory, 1–5
  self-test, 6–3
  self-test results, 6–11, 6–13
Memory size
  determining, 6–13

P

Power
  troubleshooting during booting, 6–19
Power regulators
  location, 1–8, 1–10
Power supply
  distribution, 2–8
Power system, 2–8 to 2–9
  AC power controller, 2–8
  battery backup unit, 2–8
  circuit breaker, 3–5
  DEC power bus, 3–13
  field service port, 3–13
  power and logic box, 2–8
  switched outlets, 3–13
Primary processor
  See Boot processor
Printing terminal output, 5–55
Processor, 1–5
  forcing a boot, 6–20 to 6–21
  primary, 4–11
  secondary, 4–11
  self-test, 4–11, 6–3
  self-test results, 6–11
Program mode, 3–6
  definition, 5–6
  entering, 5–7
Progress trace, 6–6

R

RBV20/RBV64, D–2
RRD compact disk drive, A–1 to A–4

S

Secondary processors
  See Processor
Self-test
  DWMBB, 6–3
  EEPROM patch level, 6–15
  EEPROM version, 6–15
  explanation of sample configuration, 6–5
  line
    identification, 6–14 to 6–15
    ILV, 6–12 to 6–13
    Mb, 6–12 to 6–13
    XBI, 6–16 to 6–17
  lines
    BPD, 6–10 to 6–11
    ETF, 6–10 to 6–11
    NODE #, 6–8 to 6–9
    progress trace, 6–6 to 6–7
    STF, 6–8 to 6–9
    TYP, 6–8 to 6–9
  module self-test status, 6–9
  module types, 6–9
  node numbers, 6–9
  order of testing, 6–2
  overview, 6–2 to 6–3
  processor and memory
    designation on self-test results, 6–11
  progress trace, 6–6 to 6–7
  sample, 6–4 to 6–5
  system identification, 6–15
  VAXBI, 6–3
  VAXBI module test results, 6–17
  when invoked, 6–4
SET CPU command, 5–46 to 5–49

System
airflow sensor, 2–15
architecture, 1–4
configuration, 1–3
footprint, 1–3
front view, 1–8
initialization, 5–35
rear view, 1–10
serial number, 6–15
thermostat, 2–15
typical, 1–7

T

Tape cartridge, B–1 to B–5
handling and storage, B–5
labeling, B–5
write protecting, B–4
Tape drive, in-cabinet, 2–6 to 2–7, B–1 to B–5
controls and indicators, B–2 to B–3
loading a tape, B–3
location, 1–8
unloading a tape, B–4
TBK50, D–2
TBK70, D–2
Temperature, 3–13
Terminal connector, 2–13
TF85 tape drive, B–2
TF tape drive, in-cabinet, 2–6 to 2–7
TK tape drive, in-cabinet, 2–6 to 2–7
TM32, D–2
Troubleshooting, 6–1 to 6–21
during booting, 6–18 to 6–19
forcing a boot processor, 6–20 to 6–21
TU81E, D–2

U

ULTRIX booting, 4–12, F–2

V

VAXBI adapters, D–1 to D–2
self-test, 6–17
VAXBI bus, 1–3 to 1–5
VAXBI card cages, 2–16 to 2–17
location, 1–8, 1–10
VAXBI expander cabinet, D–3
VAXBI modules
self-test, 6–17
VAXBI node
initializing, 5–34
VAXcluster booting, 4–12 to 4–15
Vector processor, 5–73
VMB (virtual memory boot), 4–3
VMEbus, 1–5
VMS booting, F–1
Voltages, input,
VAXBI, D–4
XMI, 2–9

X

XMI adapters, 1–12 to 1–13
XMI bus, 1–3 to 1–5, 2–10 to 2–11
node number of boot processor, 4–11
XMI card cage, 2–10 to 2–11
configuration, 2–11
door, 2–11
interlock switch, 2–11
location, 1–8, 1–10
slot numbers, 2–11
XMI modules
LEDs, 2–11, 6–3
self-test, 6–8
XMI node
ID numbers, 2–11
initializing, 5–34
XMI-to-VAXBI adapter
self-test results, 6–17
XTC module
location, 1–10