VAX 6300
VAX Fileserver 6300

Owner's Manual

Order Number EK–620AC–OM–003

This manual covers the daily operations of a VAX 6300 and is intended for the system manager or system operator. It also applies to VAX 6200 systems.

digital equipment corporation
maynard, massachusetts

First Printing, May 1988
Revised, May 1989

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

Chapter 4 Booting

Chapter 5 Console

Chapter 6 System Self-Test and Troubleshooting

Appendix A TK70 Tape Drive Instructions

Appendix B Console Error Messages

Appendix C Control Flags for Booting

Appendix D Console Commands

Appendix E Device Type Code Assignments

Glossary

Index

Examples

Figures

Tables

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.

Document Structure

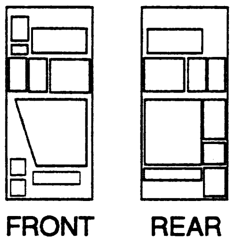
This manual uses a structured documentation design. There are many topics, organized into small sections for efficient reference. Each topic begins with an abstract. You can quickly gain a comprehensive overview by reading only the abstracts. Next is an illustration or example, which also provides quick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

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

- **Chapter 1, The VAX 6300 System Family**, and **Chapter 2, System Components**, give you a basic introduction to your system and its parts.
- **Chapter 3, Controls and Indicators**, describes how the VAX 6300 presents information and how you use the switches.
- **Chapter 4, Booting**, explains how you turn the VAX 6300 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 TK tape drive instructions, Appendix B has the console error messages, Appendix C lists control flags for booting, Appendix D summarizes the console commands, and Appendix E shows the XMI and VAXBI device type code assignments.
- 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 6300 systems. A shaded area in the icon shows the location of the component or part being discussed.



VAX 6300 Documents

The VAX 6300 documentation set includes the following documents:

Title	Order Number
VAX 6300 Installation Guide	EK–620AC–IN
VAX 6300 Owner's Manual	EK–620AC–OM
VAX 6300 Mini-Reference	EK–620AC–HR
VAX 6300 Options and Maintenance	EK–620AB–MG
VAX 6300 System Technical User's Guide	EK–620AB–TM

Associated Documents

Other documents that relate to the VAX 6300 include:

Title	Order Number
CIBCA User Guide	EK–CIBCA–UG
H4000 DIGITAL Ethernet Transceiver Installation Manual	EK–H4000–IN
H7231 Battery Backup Unit User's Guide	EK–H7231–UG
H9657–EU Installation Guide	EK–VBIEU–IN
KDB50 Disk Controller User's Guide	EK–KDB50–UG
RA82 Disk Drive User's Guide	EK–ORA82–UG
RA90 Disk Drive User's Guide	EK–ORA90–UG
RV20 Optical Disk Owner's Manual	EK–ORV20–OM
SC008 Star Coupler User's Guide	EK–SC008–UG
TK70 Streaming Tape Drive Owner's Manual	EK–OTK70–OM
TU81/TA81 and TU81 PLUS Subsystem User's Guide	EK–TUA81–UG
VAX Architecture Reference Manual	EY–3459E–DP
VAX Systems Hardware Handbook — VAXBI Systems	EB–31692-46
VAXBI Expander Cabinet Installation Guide	EK–VBIEA–IN
VAXBI Options Handbook	EB–32255-46
VMS Installation and Operations: VAX 6200, 6300 Series	AA–LB36B–TE
VMS Networking Manual	AA–LA48A–TE
VMS System Manager's Manual	AA–LA00A–TE
VMS VAXcluster Manual	AA–LA27A–TE

Chapter 1

The VAX 6300 System Family

The VAX 6300, a general-purpose computer system designed for growth, is configured for many different applications. Like other VAX systems, the VAX 6300 can support many users in a time-sharing environment. The VAX 6300 does the following:

- Supports a full set of VAX applications
- Functions as a stand-alone system, a member of a VAXcluster, or a boot node of a local area VAXcluster
- Allows for expansion of processors, memory, and I/O
- Implements multiprocessing where all processors have equal access to memory
- Uses the VAXBI bus (VAX Bus Interconnect) as the I/O interconnect
- Uses a high-bandwidth internal system bus designed for multiprocessing
- Interleaves memory bank accesses in a user-definable sequence
- Performs automatic self-test on power-up, reset, reboot, or system initialization

This chapter describes system packages and introduces the location of components in the cabinet—both front and rear views. Sections include:

- System characteristics
- System architecture
- Typical system
- System front view
- System rear view
- VAXBI expander cabinet
- Supported VAXBI adapters

1.1 System Characteristics

The VAX 6300 family has packages with from 1 to 6 processors. Each 60-inch system cabinet includes one 14-slot high-bandwidth internal system bus backplane (XMI) and two 6-slot VAXBI backplanes. All systems share the same system characteristics as shown in Table 1-1.

Table 1–1: System Characteristics

Physical		cm (in)
	Height	154 (60.5)
	Width	78 (30.5)
	Depth	76 (30.0)
	Weight	318 kg (700 lbs)
Environmental		
Heat dissipation (max)		5440 Btu/hr
Operating temperature	TK not in use	10° to 40°C (50° to 104°F)
	TK in use	15° to 32°C (59° to 90°F)
Operating humidity	TK not in use	10 to 90% relative humidity
	TK in use	20 to 80% relative humidity
Altitude	Non-operational	0 to 9.1 km (0 to 30,000 ft)
	Operating	0 to 2.4 km (0 to 8000 ft)

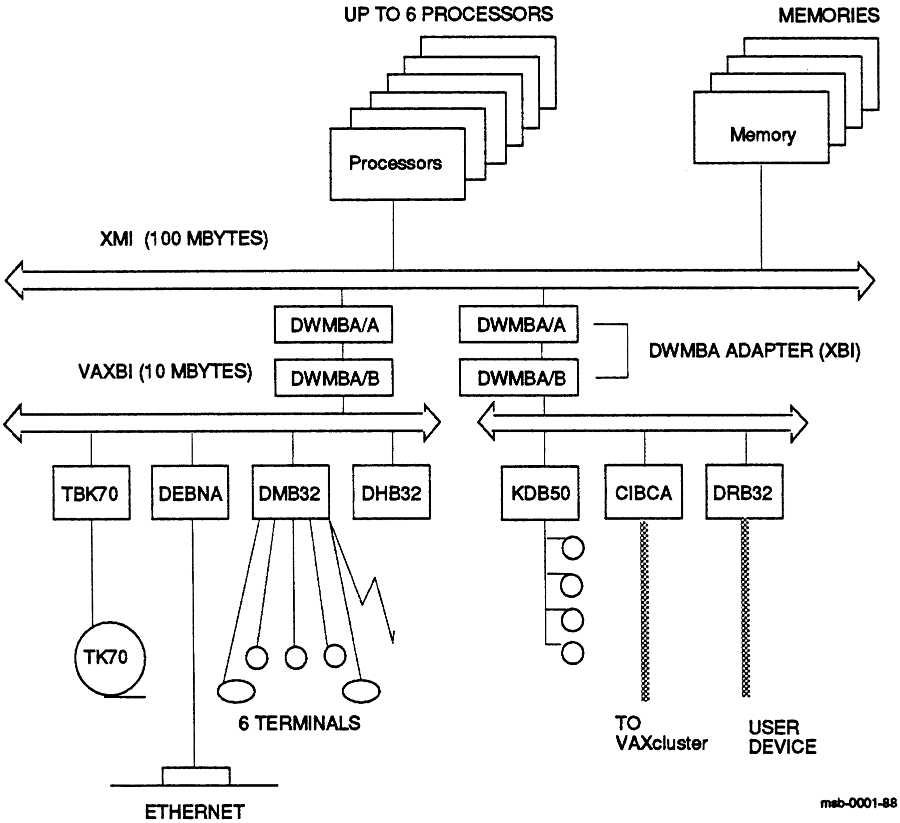
Table 1–1 (Cont.): System Characteristics

Electrical		
AC power consumption (max)		1.6 kW
AC current (max)	60 Hz	8 A (208 V)
	50 Hz	4 A (416 V), 4.5 A (380 V)
Voltage input	60 Hz	3-phase 208 V RMS
	50 Hz	3-phase 380/416 V RMS
Frequency tolerance		47–63 Hz
Surge current		60 A

1.2 System Architecture

The system uses a high-speed bus, called the XMI bus, to interconnect its processors and its memory modules. All I/O devices connect to the VAXBI bus. The system supports multiprocessing with up to six processors.

Figure 1–1: VAX 6300 System Architecture



The XMI is the system bus; the VAXBI bus supports the I/O subsystem. The XMI is a 64-bit system bus[1] that interconnects the central processors, memory modules, and VAXBI I/O adapters.

The VAXBI and XMI share similar but incompatible connector and module architecture. Both the VAXBI and XMI buses use 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 (KA62B), memory nodes (MS62A), and the XMI-to-VAXBI I/O adapters (DWMBA).

A **processor node**, called a KA62B for timeshare systems and a KA62B-S for server systems, is a single-board VAX processor. It contains a central processor unit (CPU) chip with its own cache, a floating-point processor, a secondary cache, a writable PROM for system parameters, and a custom gate array for interfacing to the XMI bus.

Processors communicate with main memory over the XMI bus. The system supports multiprocessing with up to six processors. One processor becomes the boot processor during power-up, and that boot processor handles all communication. The other processors become secondary processors and receive system information from the primary processor (see Section 4.5).

A **memory node** is an MS62A. Memory is a global resource equally accessible by all processors on the XMI. Each MS62A module has 32 Mbytes of memory, consisting of MOS 1–Mbit dynamic RAMs, ECC logic, and control logic. The memories are automatically interleaved for maximum performance, or may be custom set by console command. An optional battery backup unit protects memory in case of power failure.

An **XMI-to-VAXBI adapter**, called a **DWMBA**, is a 2-board adapter that maps data between these two buses. The DWMBA/A module is installed on the XMI bus; it communicates with the DWMBA/B module on the VAXBI using a 120-pin cable. Every VAXBI on this system must have a DWMBA adapter. Therefore, systems with two VAXBI channels have two DWMBA/A modules on the XMI bus, and each VAXBI has a DWMBA/B module in its card cage. System error messages and self-test results refer to the pair of DWMBA modules as XBI.

The VAXBI, in turn, passes data between the system and the peripheral devices.

[1] The XMI has a 64-nanosecond bus cycle, with a maximum throughput of 100 Mbytes per second.

1.3 Typical System

A typical system has a main cabinet with a TK tape drive, a console terminal and printer, a disk drive cabinet, an accessories kit, and a set of documentation, including this manual. The system may have additional tape or disk drives and may be a member of a VAXcluster.

Figure 1–2: Typical System

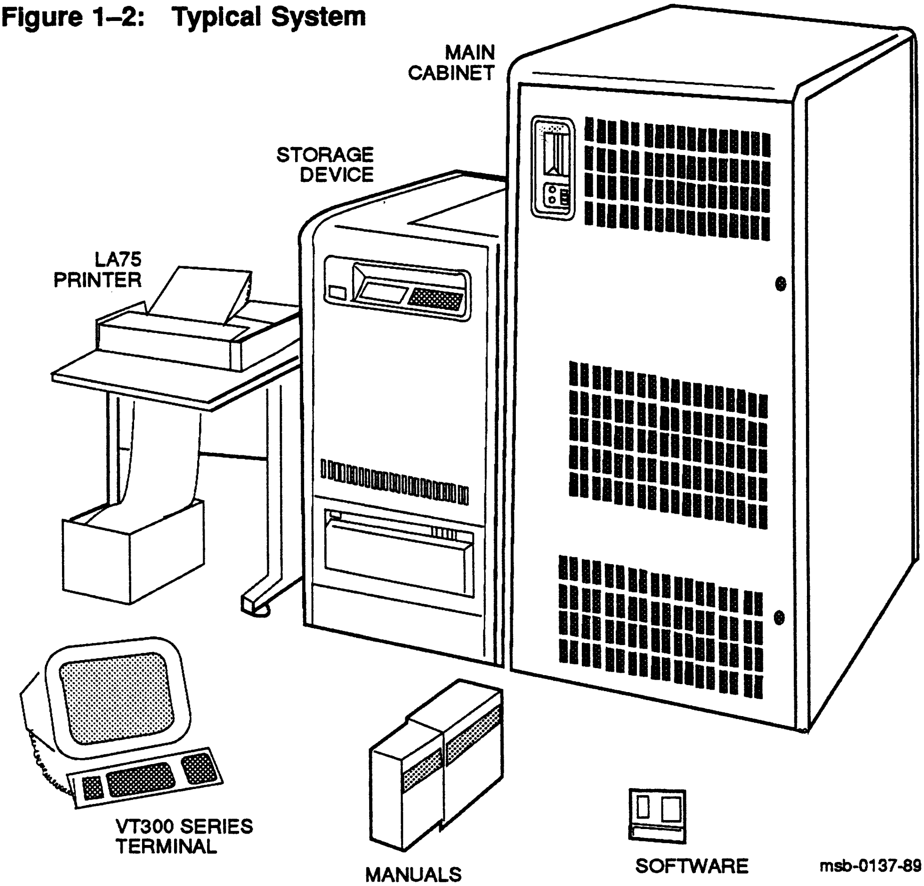


Table 1–2: Typical System

Component	Function
Main cabinet	Houses system components
TK tape drive	Software distribution; stores and transfers data
Console terminal	Manages system and its resources
Console printer	Provides hardcopy of console transactions
System documentation	See the Preface for full list of documentation related to VAX 6300 systems
Disk expansion cabinet	Provides storage capacity

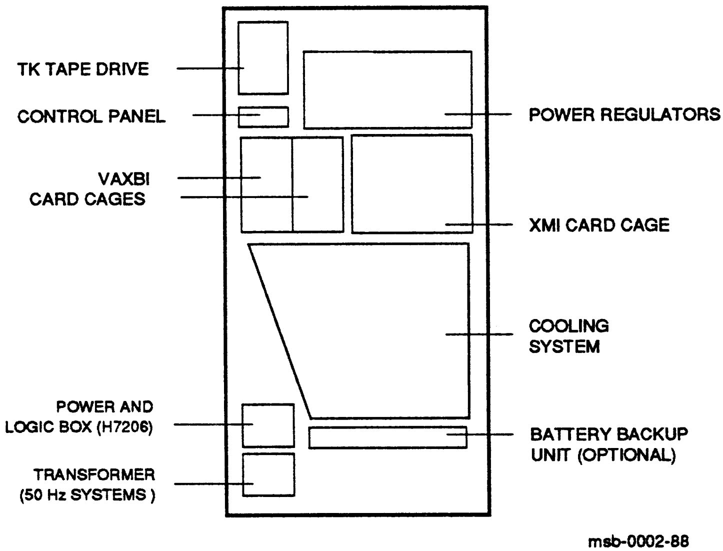
Your DIGITAL field service representative has installed your system and verified that it is running properly. Before you turn on the system, familiarize yourself with its components:

- **The main cabinet** houses a TK tape drive, the XMI card cage (which contains the processors and memories), two VAXBI card cages, the control panel switches, status indicators, and restart controls.
- **The TK tape drive** in the main cabinet is used for installing operating systems, software, and some diagnostics.
- **The disk drive cabinet** has local storage and archiving capability.
- **The console terminal and printer** are used for booting and for system management operations.
- **VAX 6300 documentation** includes:
  - *VAX 6300 Installation Guide*
  - *VAX 6300 Owner's Manual*
  - *VAX 6300 Mini-Reference*

1.4 System Front View

The TK tape drive and control panel are on the front of the system cabinet, accessible with the doors closed. With the front door open, field service representatives can access the VAXBI and XMI card cages, the cooling system, the battery backup unit, if present, and power regulators.

Figure 1–3: System Front View



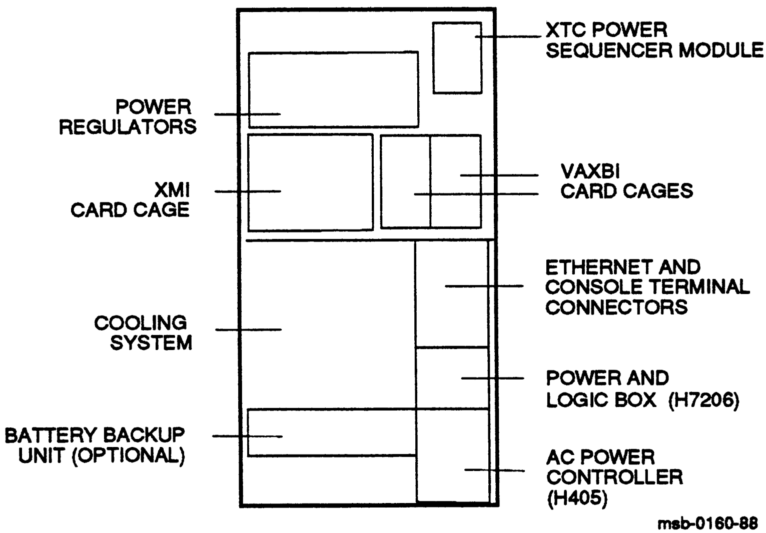
These components are visible from the inside front of the cabinet (see Figure 1–3 for their location):

- TK tape drive
- Control panel
- Power regulators
- Two VAXBI card cages
- XMI card cage
- Cooling system
  One of the two blowers is visible from the front of the cabinet.
- Battery backup (if installed)

1.5 System Rear View

With the rear door open, field service representatives can access the power regulators; power sequencer module (XTC); cooling system; power and logic box; battery backup unit, if present; AC power controller; terminal, disk, and console connectors; and the I/O bulkhead space.

Figure 1–4: System Rear View



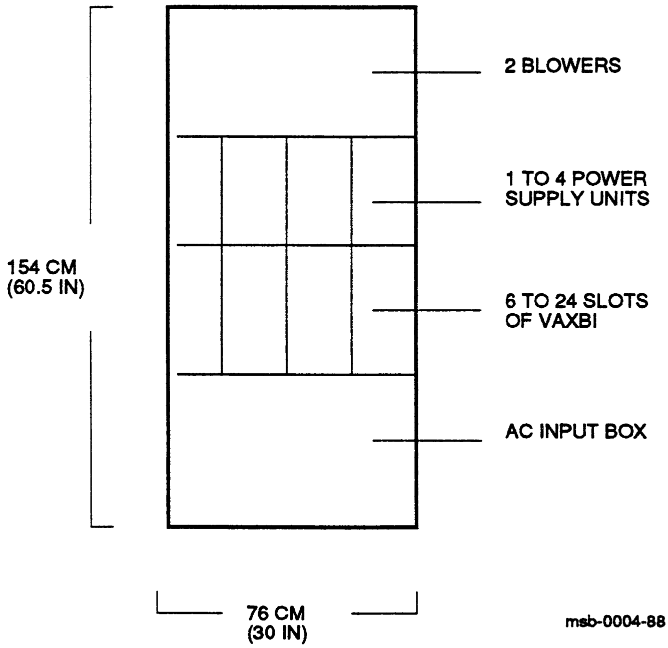
These components are visible from the rear of the cabinet (see Figure 1–4):

- Five field-replaceable power regulators
- Power sequencer module (XTC) located on the back of the TK tape drive and control panel unit
- 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.
- Cooling system, with open grid over a blower
- VAXBI and XMI adapter bulkhead cables
- Terminal, disk, and console connectors
- Power and logic box (H7206)
- Battery backup unit (optional)
- AC power controller (H405)

1.6 VAXBI Expander Cabinet

A VAXBI expander cabinet can be ordered to increase the system's VAXBI I/O slots. With a VAXBI expander cabinet 5 to 20 additional slots are available for I/O.

Figure 1–5: VAXBI Expander Cabinet



A VAXBI expander cabinet (Figure 1–5) allows you to attach additional VAXBI channels, each with its required DWMBA/B.

The cabinet is 154 centimeters (60.5 inches) high by 76 centimeters (30 inches) wide. Four power supply units provide power to the VAXBI backplanes. 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 6300 Installation Guide*.

1.7 Supported VAXBI Adapters

The system supports the use of the following VAXBI adapters: CIBCA, DEBNA, DHB32, DMB32, DRB32, DSB32, KDB50, RBV20/RBV64, TBK70, TU81E, and DWMBA.

Figure 1–6: VAXBI Adapters

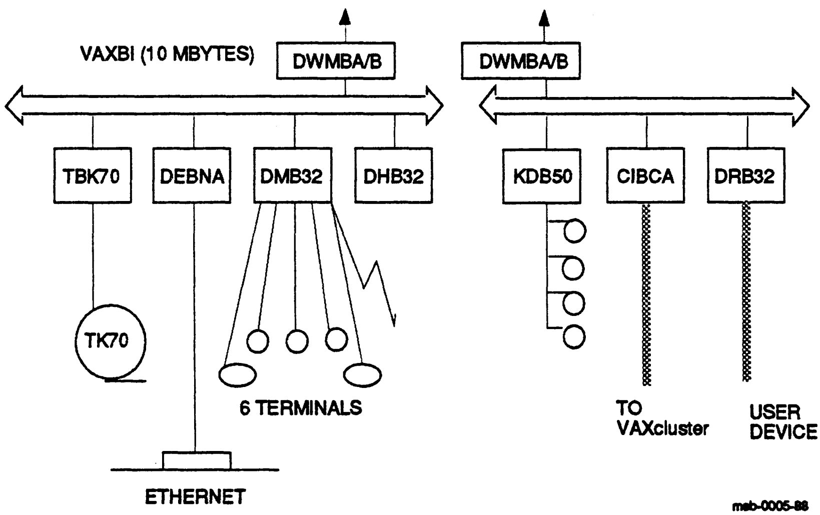


Table 1–3 lists some of the VAXBI devices supported by VAX 6300 systems. Note that some VAXBI adapters have more than one module, requiring more than one slot on the VAXBI. Consult your field service representative for configuration rules.

Table 1–3: VAXBI Adapters

Adapter	Std Opt'l	No. Slots	Function
CIBCA	O[1]	2	VAXcluster port interface; connects a system to a VAXcluster.
DEBNA	S	1	Ethernet port interface; connects a system to the Ethernet.
DHB32	O	1	Communication device; supports up to 16 terminals.
DMB32	O	1	Interface for 8-channel asynchronous communications for terminals with one synchronous channel for a line printer.
DRB32	O	1 or 2	Parallel port.
DSB32	O	1	Two-channel synchronous communication device.
DWMBA	S	1	VAXBI-to-XMI interface.
KDB50	O[1]	2	DSA disk adapter; enables connection to disk drives.
RBV20/RBV64	O	1	Write-once optical drive controller.
TBK70	S	1	TK70 tape drive controller; connects the TK to the system.
TU81E	O	1	TU81 controller; local (nonclustered) tape subsystem.

[1]One disk or VAXcluster adapter is standard. You may add additional disk adapters.

See Appendix E in this book, the *VAX Systems and Options Catalog*, or the *VAXBI Options Handbook* for more information on VAXBI adapters.

Chapter 2

System Components

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

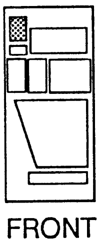
- TK tape drive
- Power system
- XMI card cage
- VAXBI card cage
- I/O connections
- Cooling system

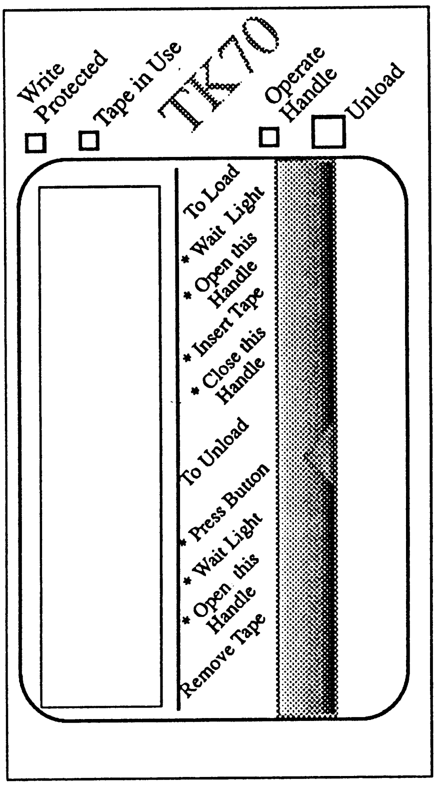
WARNING: *The inside of a VAX 6300 cabinet is not designed to be accessed by the customer. The information in this chapter is for your information only. The cabinet doors are to be opened only by field service representatives.*

2.1 TK Tape Drive

The TK tape drive is mounted at the front of the system cabinet in the upper left corner. You use the TK tape drive for software installation and diagnostics. User applications may use the TK as an I/O device.

Figure 2-1: TK Tape Drive





The TK tape drive is used for:

- Installing or updating software
- Loading diagnostics
- Interchanging user data
- Saving, restoring, and updating contents of the EEPROM
- Loading stand-alone backup

The TBK70 adapter is located in the VAXBI and is the interface to the TK tape. For more information on how to use the TK tape drive, see Appendix A, TK Tape Drive Instructions, or the *TK70 Streaming Tape Drive Owner's Manual*.

2.2 Power System

The power system consists of an AC power controller (H405E/F) with circuit breaker, the power and logic box (H7206), five power regulators for the XMI and VAXBI backplanes, and an optional battery backup unit.

Figure 2–2: Power System (Rear View)

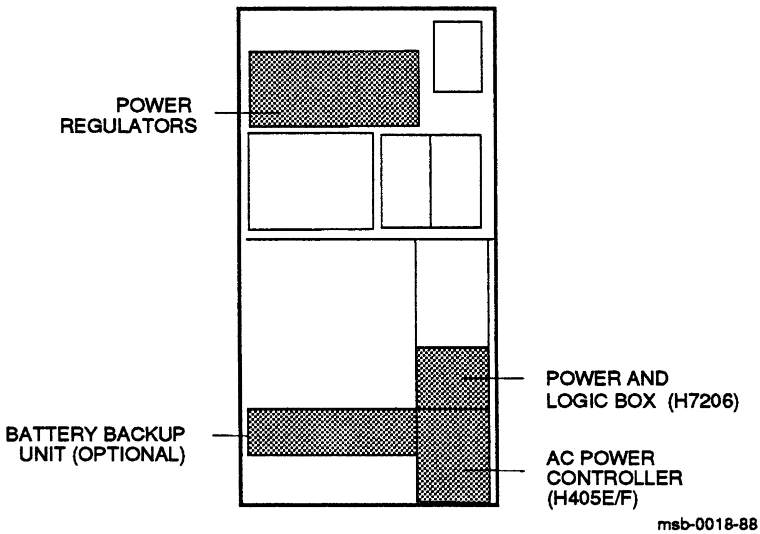


Table 2–1: Input Voltage

Model No.	Hz	Nominal Input Voltages	Phase
H405E	60	208 V	3
H405F	50 50	380 V 416 V	3 3

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 for the XMI and VAXBI card cages.

The power supply is made up of two 200-watt and three 600-watt power regulators. One 200-watt unit and one 600-watt unit supply the power to the VAXBI; one 200-watt unit and two 600-watt units supply the power to the XMI. See Table 2–2. The power supply includes sufficient power for any combination of available XMI modules.

Table 2–2: Power Supply Available for VAXBI Options

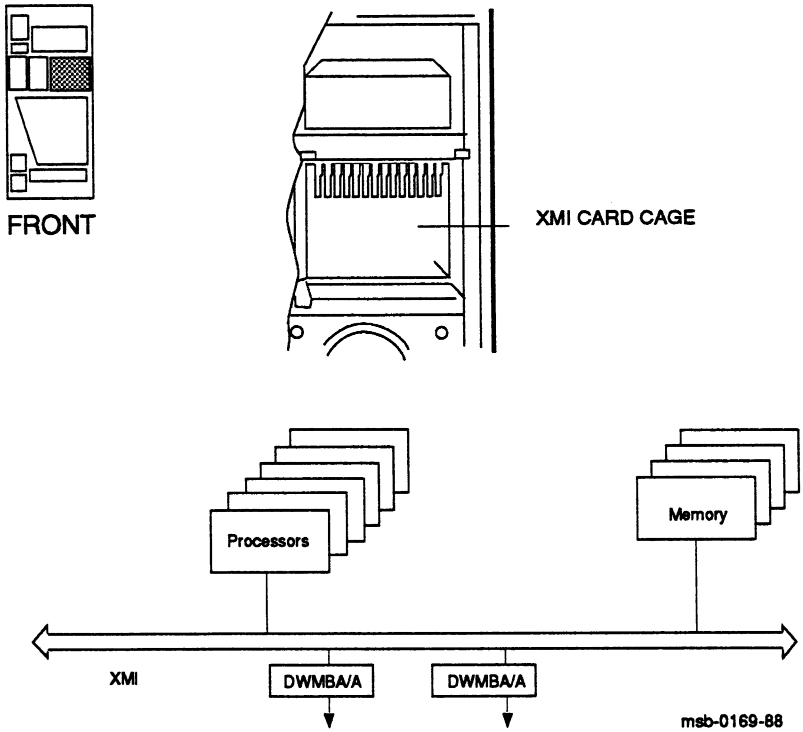
DC Voltage	Available VAXBI Current	Note
+5V	86.0 A	Main logic
+5VBB	Connected to +5V	Not battery backed up
+12V	4.0 A	RS–232
–12V	2.4 A	RS–232
–5.2V	20.0 A	ECL logic
–2V	7.0 A	ECL logic

The optional H7231 battery backup unit, if present, is located in the rear left lower third of the cabinet, near the airflow funnel. This unit supplies power to sustain memory for up to 10 minutes following power interruption to system memory. The control panel on the front of the system indicates the status of the battery backup unit. Two power connections (60 Hz systems only) are on the back face of the power controller and are fuse-protected. When the system is powered down, the devices attached at these switches also are powered down. Three neon lights on the AC power controller (60 Hz systems only) indicate the presence of the 3-phase voltages at the input to the power controller.

2.3 XMI Card Cage

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

Figure 2–3: XMI System Bus



The XMI is a limited-length, pended, synchronous bus with centralized arbitration. The XMI bus 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 XMI 14-slot 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 XMI 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. (See the *VAX 6300 Options and Maintenance* manual for details of module indicator lights.)

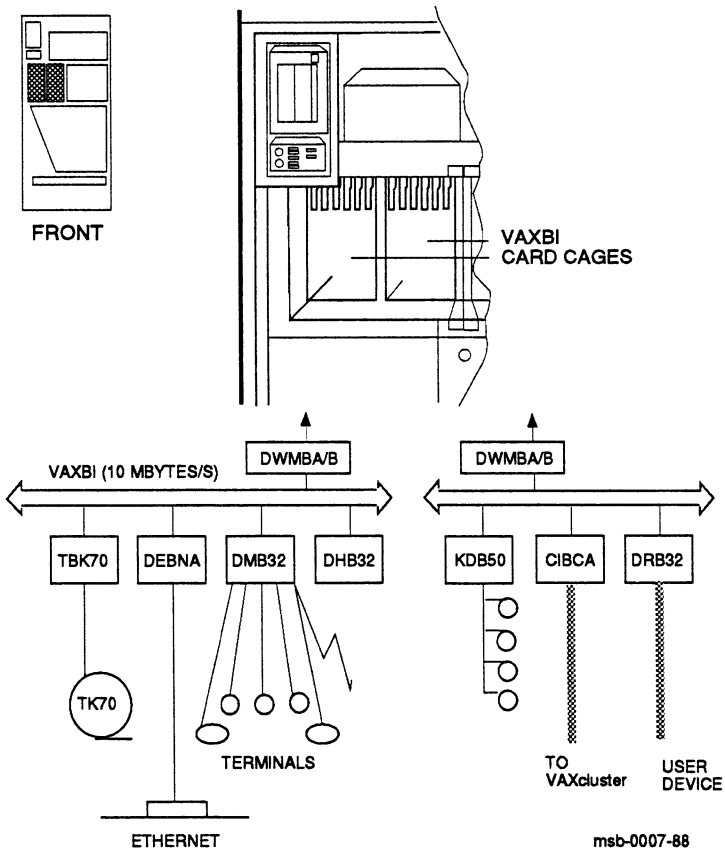
Each slot of the XMI card cage is hard-wired to a 4-bit node ID code that corresponds to the physical slot number in the card cage. The node ID number of the module is its slot position. The slots are numbered 1 through E (hexadecimal) from right to left, as you view the card cage from the front of the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

For information on installing modules in the XMI card cage, see the *VAX 6300 Options and Maintenance* manual. For technical information and configuration rules, see the *VAX 6300 System Technical User's Guide*.

2.4 VAXBI Card Cage

The VAXBI is the I/O interface. The VAXBI card cages house modules that connect the system to the Ethernet, VAXclusters, multiple terminals, and other peripherals.

Figure 2–4: VAXBI I/O Interface Bus



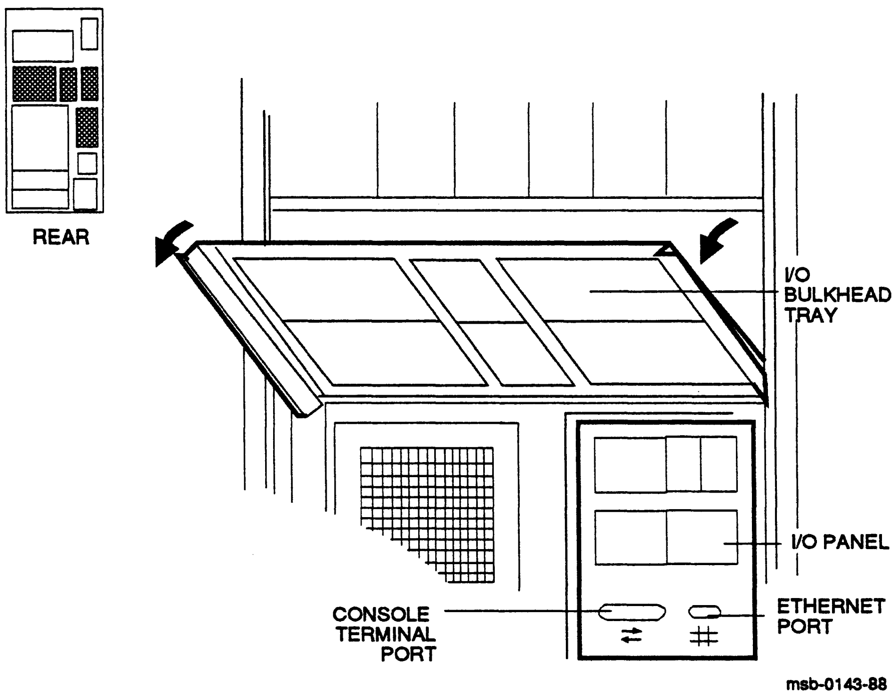
The VAXBI bus is a high-performance 32-bit bus that is the system's I/O interface. Two 6-slot VAXBI card cages are located in the upper third of the cabinet on the left side, as viewed from the front of the cabinet. A clear latched door protects the components housed in the VAXBI card cage and helps to direct the airflow over the modules. Indicator lights on the modules in the VAXBI card cage can be viewed through this clear front door (see the *VAX 6300 Options and Maintenance* manual).

The VAXBI is available in the system only as a 6-slot, fixed-length, nonexpandable card cage. The VAX 6300 system has two 6-slot VAXBI card cages. You may also add a VAXBI expander cabinet (see Section 1.6).

2.5 I/O Connections

I/O connections are installed on the bulkhead connections tray and the I/O connection 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 and VAXBI backplanes. 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 2–5: Console and Terminal Connectors



The I/O bulkhead connections tray is located in the rear of the cabinet, above the cooling system and standard I/O connections 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.

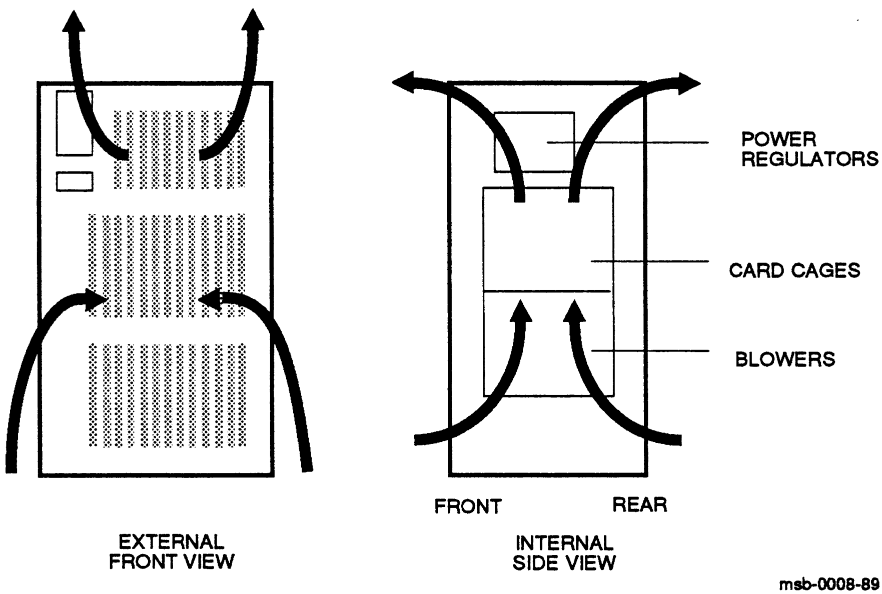
The I/O tray and panel have 30 panel units designed to accommodate a variety of I/O connectors.

The Ethernet and console terminal connectors are at the bottom of the I/O panel. The Ethernet DEBNA 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 2–5.

2.6 Cooling System

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

Figure 2–6: 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 VAXBI and XMI card cages. 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 separate 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. If your unit experiences extreme temperatures, the temperature thermostat shuts off all output power at the AC power controller except for power to the battery backup unit. 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 field service representative.

Chapter 3

Controls and Indicators

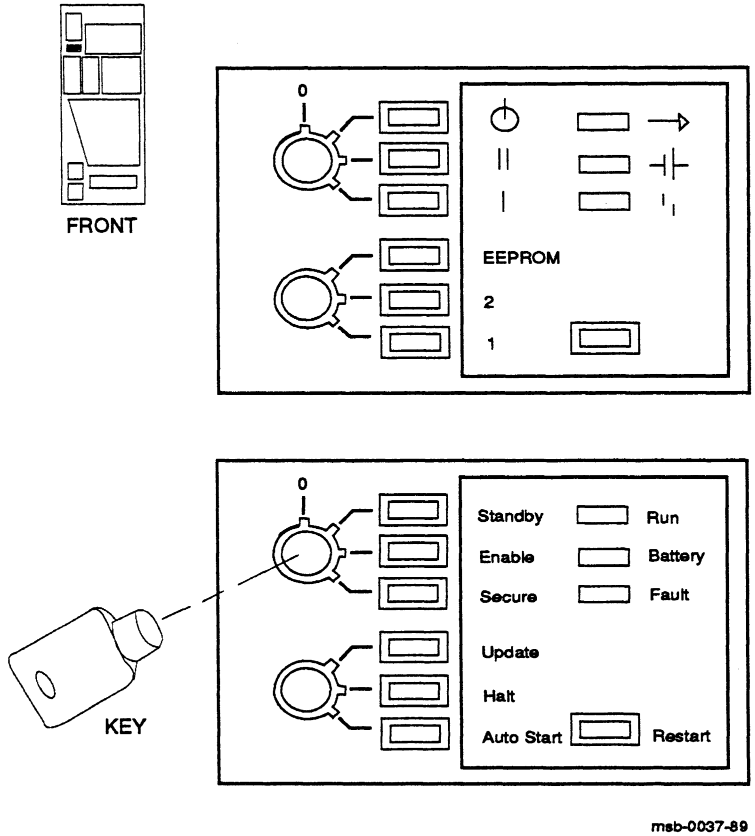
This chapter introduces system controls and indicators. Sections include:

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

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 on the upper left corner 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.

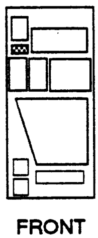
Table 3–1: Control Panel Symbols

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)

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)



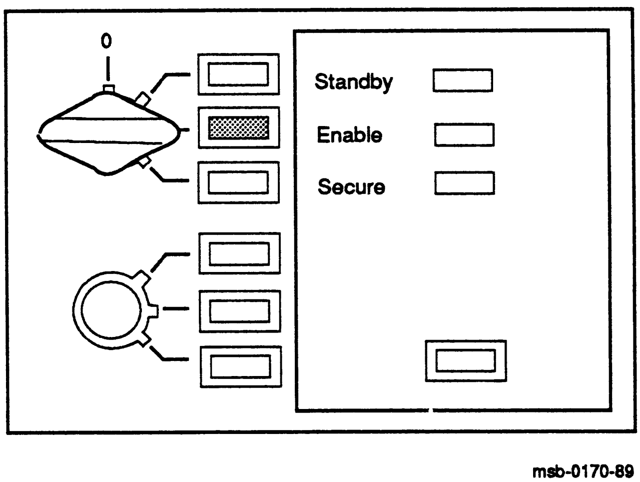


Table 3–2: Upper Key Switch

Position	Effect	Light Color
O (Off)	Removes all power, except to the battery backup unit.	No light
Standby	Supplies power only to memory and blowers.	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)	Maintains power to the whole system; console terminal is disabled. Used for normal system operation. Prevents console mode. Disables Restart button and causes the lower key switch to have the effect of Auto Start, regardless of its setting.	Green

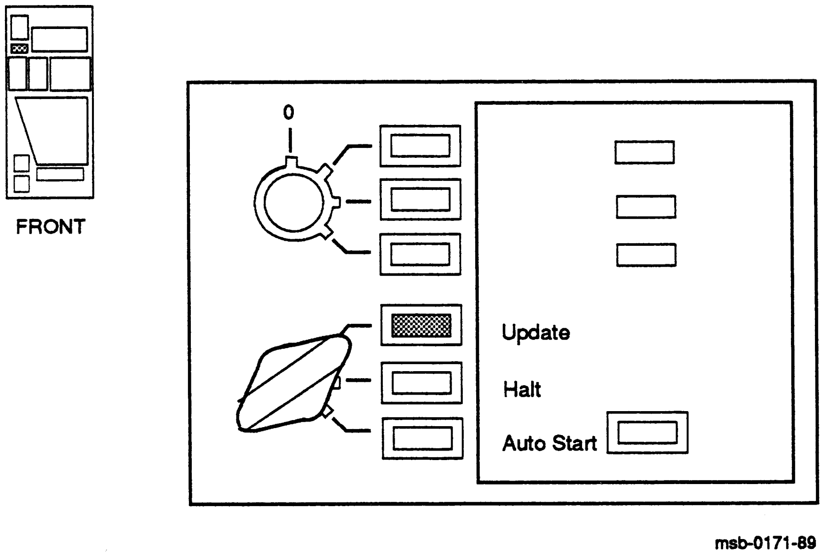
The upper key switch has four positions: Off, Standby, Enable, and Secure. You change the position of the upper key switch by inserting and turning a key. A light to the right of each key position lights to show what mode is in operation. When the switch is set to Off, no lights are lit. Each position 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 backup unit. To ensure total absence of power in the machine, pull the circuit breaker and unplug the machine. See Section 3.6.
- **Standby** powers only the memory regulator and the blowers.
- **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** maintains power to the system. During normal operation the switch is set to Secure. With the switch in the Secure position, you can use the console terminal only in program mode (as a user terminal). You cannot type CTRL/P to enter 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 activates the primary processor. 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 EEPROM on primary processor. Halts boot processor in console mode on power-up or when Restart button is pressed. Used for updating parameters stored in the processor (such as SET BOOT or UPDATE console commands) and to prevent 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.

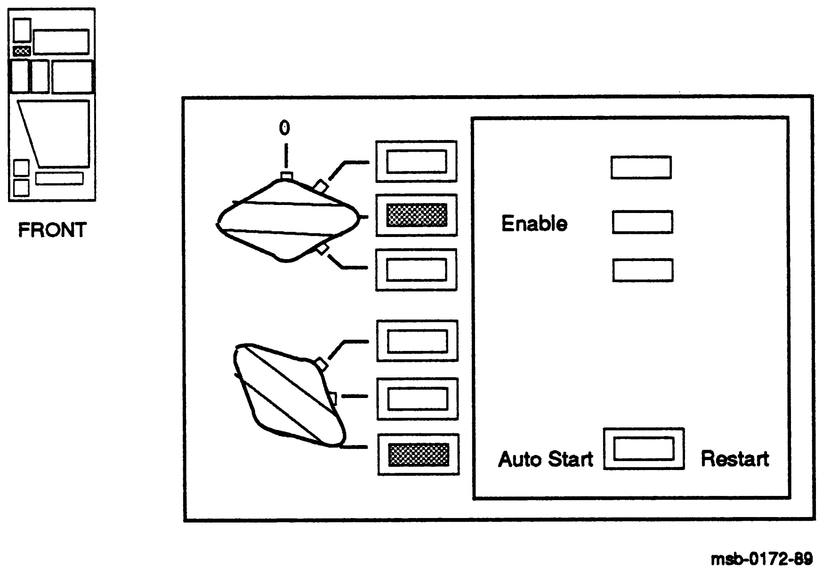
Each position engages the primary processor in a different way.

- **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, RESTORE EEPROM, SAVE EEPROM, and all the SET commands (see Section 5.17.1 through Section 5.17.5). Field service representatives and self-maintenance customers use Update for updates to the console, self-test, and diagnostics programs. See also Section 5.23, Section 5.15, and Section 5.16. 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 (see Section 4.4).

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 restart. If the restart fails, then it reboots the operating system. 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 VAX 6300 system.

Figure 3–5: Control Panel Status Indicator Lights

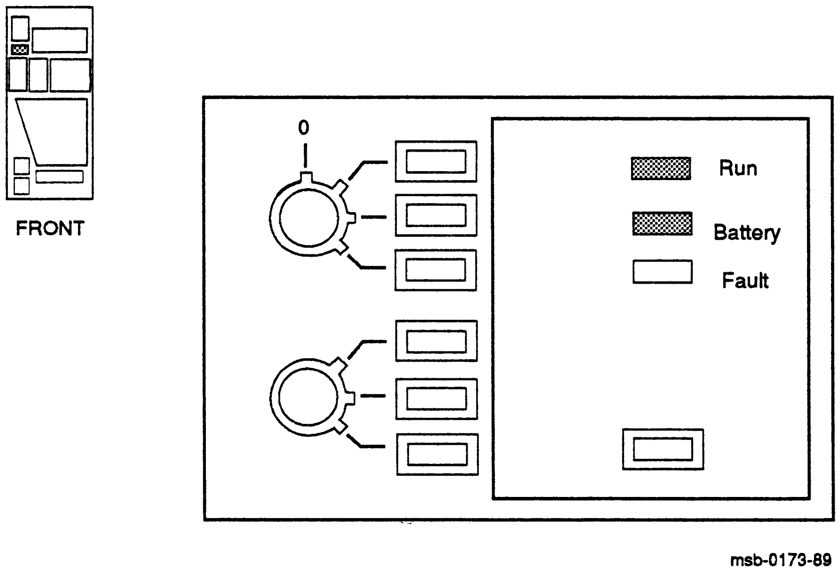


Table 3–5: Control Panel Status Indicator Lights

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 fully charged; normal operation.
		Flashing 1 x/sec	Battery backup unit is charging.
		Flashing 10 x/sec	Battery backup unit is supplying power to the system.
		Off	Either system does not have a battery backup unit or the battery backup unit is turned off.
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.

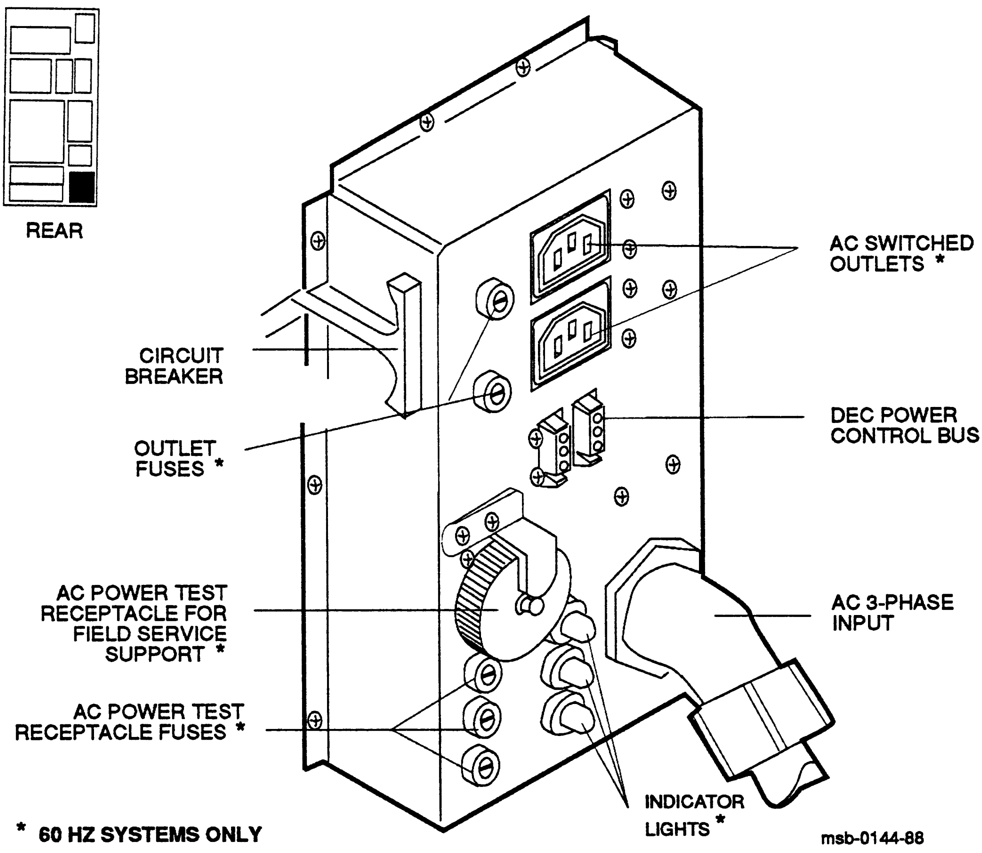
Three status indicator lights on the control panel show the state of system execution (*Run*), the presence of a battery backup unit (*Battery*), and hardware errors (*Fault*).

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.

3.6 Circuit Breaker and Power Indicator Lights

The circuit breaker and power indicator lights are located on the AC power controller, which is at the bottom right corner at the back of the cabinet.

Figure 3–6: Circuit Breaker and Power Indicator Lights



The circuit breaker and power indicator lights can be seen from the inside rear of the cabinet.

Circuit Breaker
The circuit breaker controls power to the system, including the power regulators and the blowers. 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°C (167°F), the "contactor" in the AC power controller is opened and the system powers down.

Power Indicator Lights/Fuses (60 Hz systems only)
Three power indicator lights are located on the bottom left corner of the AC power controller. Each light indicates that one phase of the 3-phase power is entering the system cabinet. If a light does not go on, you do not have all three power phases into the system.

Switched Outlets (60 Hz systems only)
Two switched outlets are on the AC power controller. The fuses for these outlets are directly to the left of the outlets. These outlets should not be used for external devices.

DEC Power Control Bus
Two DEC power control bus connectors are located on the AC power controller. These buses interconnect your system to a VAXBI Expander Cabinet or to storage cabinets. These power control bus outlets connect the power controllers, allowing power control over several units. It provides central power-up and power-down capability.

Field Service Port (60 Hz systems only)
The round capped receptacle is used by field service representatives to connect test equipment that monitors AC power.

Chapter 4
Booting

This chapter describes how to boot a VAX 6300 system. Sections include:

- How booting works
- Boot devices
- Initial boot procedure
- Regular boot procedure
- Boot processor selection
- Booting from a VAXcluster
- Booting over the Ethernet

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

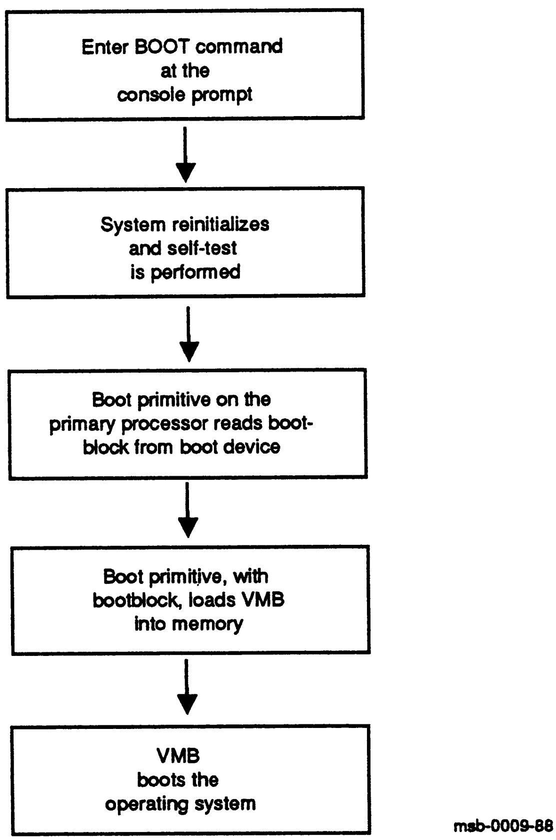


Table 4–1: Boot Procedure

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.
4	The bootblock contains a pointer to VMB. Using the information in the bootblock, the boot primitive loads VMB into the first 256-Kbyte block of available memory.
5	Once VMB is loaded into memory, the boot primitive transfers control to VMB, which in turn starts the operating system.

Boot primitives

Boot primitives are 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 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 TK tape drive, a local system disk connected through a KDB50, a disk connected to the system through a CI adapter (CIBCA) , or a disk connected to the system through the Ethernet.

Bootblock

The bootblock is 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 TK tape drive, a local system disk, a disk connected to the system through a CIBCA adapter, or a disk connected to the system through Ethernet.

Figure 4–2: Boot Devices

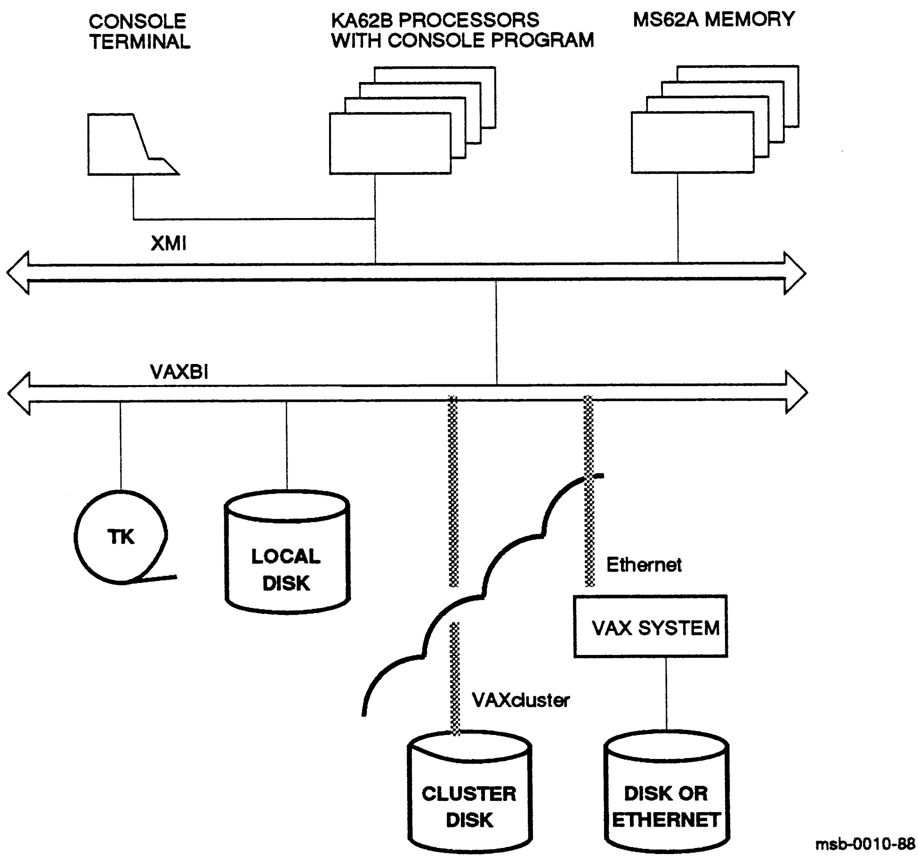


Table 4–2: Boot Devices

Device	Location
TK tape drive	Upper left corner of the system cabinet; used for booting stand-alone backup or diagnostics.
Local disk	Disk connected to the system through the VAXBI; regular boot procedure specifies such a disk as default boot device for individual systems that are not VAXclustered or networked.
VAXcluster disk	Disk located on system's VAXcluster connected to the system by an HSC controller.
Ethernet disk	Disk connected to the system over Ethernet, through the VAXBI and the DEBNA Ethernet port interface.

You have a choice of four boot device locations. You can boot from the TK tape drive located in the system cabinet, from a local disk connected to the system, from a disk on the VAXcluster interface (CI), or by Ethernet from a remote disk on another system.

The TK tape drive can be used to perform the following tasks:

- Boot stand-alone backup
- Install or upgrade operating systems
- Initial boot following installation
- Boot 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.

4.3 Initial Boot Procedure

The first time you boot the system, load the TK console tape containing stand-alone backup. Use the command BOOT CSA1.

Example 4–1: Initial Boot Command

```
>>> BOOT CSA1                   ! Entering initial BOOT command;
                                ! system reinitializes and responds
                                ! with self-test results:
Initializing system.

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

    A   A   .   .   M   M   .   .   .   .   .   .   P   P       TYP
    o   o   .   .   +   +   .   .   .   .   .   .   +   +       STF
    .   .   .   .   .   .   .   .   .   .   .   .   E   B       BPD
    .   .   .   .   .   .   .   .   .   .   .   .   +   +       ETF
    .   .   .   .   .   .   .   .   .   .   .   .   E   B       BPD

.   .   .   .   .   .   .   .   .   +   +   .   +   .   +   .   XBI D +
.   .   .   .   .   .   .   .   .   +   .   +   .   +   +   .   XBI E +

    .   .   .   .   A2  A1  .   .   .   .   .   .   .   .       ILV
    .   .   .   .   32  32  .   .   .   .   .   .   .   .       64Mb

ROM = 4.1           EEPROM = 2.0/4.1        SN = SG91234567

Loading system software.
   VAX/VMS Version XXX     Major version id = XX    Minor version id = XX

%SMP-I-CPUBOOTED, CPU #2 has joined the PRIMARY CPU in multiprocessor
                  operation
Please remove the volume "CONSOL" from the console device.
Insert the first stand-alone system volume and enter "YES" when ready:
! Remove the console tape cartridge.  Insert the tape cartridge labeled
! VAX/VMS A52A TK70. When the distribution tape cartridge is loaded,
! type YES and press RETURN to indicate that you are ready to continue.
! System response:

Resuming load operation on volume 'SYSTEM', please stand by...

   VAX/VMS Version XXX     Major version id = CC    Minor version id = XX

PLEASE ENTER DATE AND TIME (DD-MMM-YYYY  HH:MM)   31-DEC-1989 15:00

! Enter date and time information. Operating system continues with boot.
```

The first time you boot the system after installation, you must use the TK tape drive to run the stand-alone backup program that was shipped with your machine. (Appendix A gives instructions on using the TK tape drive.)

After you have inserted the TK tape cartridge containing the stand-alone backup program, invoke the booting procedure by using this command:

```
BOOT CSA1
```

where CSA1 is the name for the console storage device (the TK tape drive) that is preprogrammed in ROM. The command BOOT CSA1 is designed for the initial boot procedure following system installation.

The system reinitializes and performs self-test. The results of self-test are displayed on your console terminal. (See Section 6.2 for a description of the self-test results display.)

You can now install your operating system. See your operating system manual for further installation instructions.

4.4 Regular Boot Procedure

With the system in console mode, you issue a BOOT command. The qualifiers to the BOOT command determine the boot device.

Figure 4–3: Regular Boot Procedure

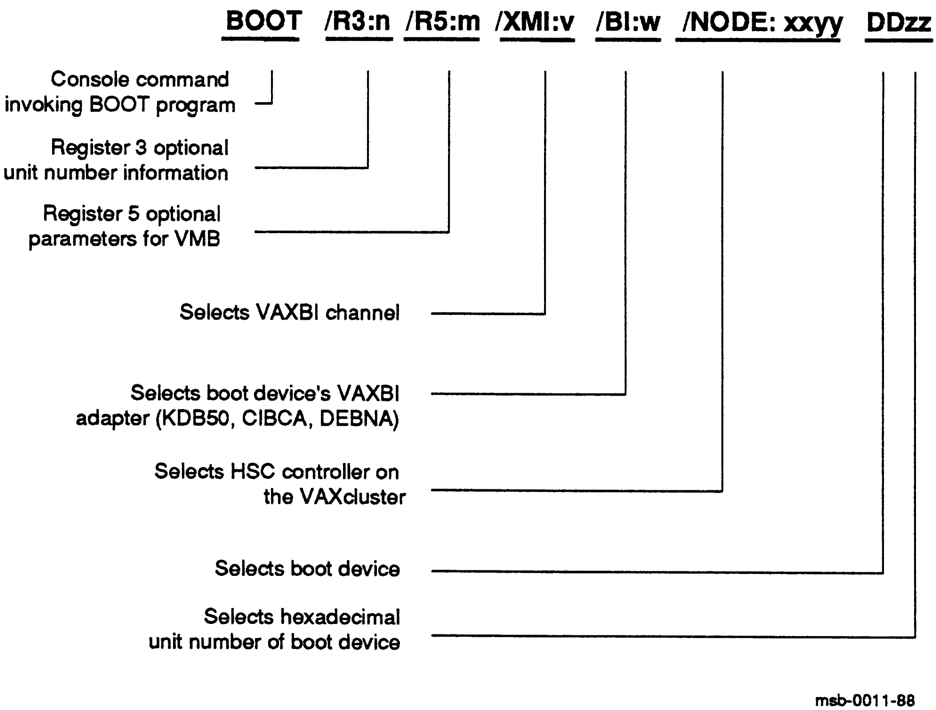


Figure 4–3 shows the components of the BOOT command. Section 5.6 describes the BOOT command in detail.

When using the /R5:m qualifier, see Appendix C for the values of *m*. The /R3:n qualifier is used with VMS when you boot from a shadow set, where *n* 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.

Since the /XMI node number you enter is a DWMBA adapter, you must specify a node on the VAXBI. You would use both the /XMI and the /BI qualifiers to specify a boot device located on the VAXBI, such as a local disk or the TK tape drive.[1]

If you designate a VAXcluster disk as your boot device, you use the /XMI, /BI, and /NODE qualifiers with their respective node numbers. The /XMI node number must be a DWMBA adapter, the /BI node number must be the number of the CIBCA adapter, /NODE's node number carries the CI node number of one or two HSC controllers and DU*zz*, where DU specifies the device type as a disk and *zz* is the hexadecimal unit of the disk boot device.

Once you have decided on your boot device, you can store the BOOT command under a nickname, using the command:

```
SE[T] B[OOT] xxxx
```

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.

You can store up to 10 saved boot specifications, in addition to the default specification. See Section 5.17.1 for details on creating nicknames for boot devices. See Section 5.6 for information on the BOOT command.

Table 4–3: Sample BOOT Commands

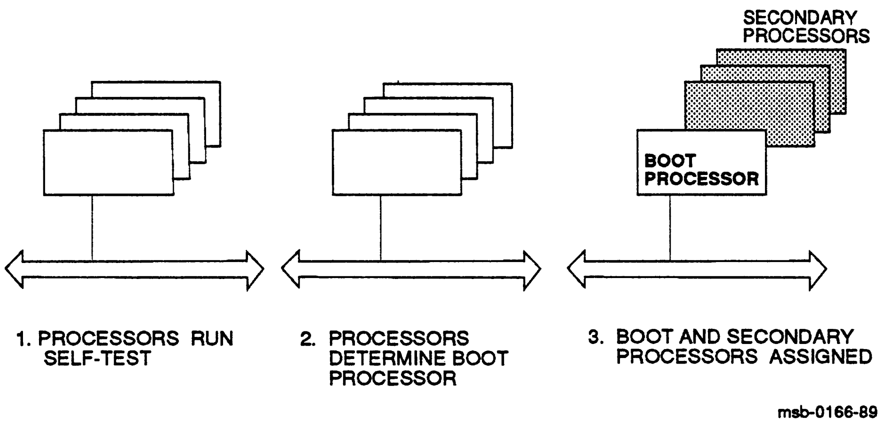
Boot Procedure	BOOT Command	Refer to
Boot from TK tape drive	BOOT CSA1	Section 4.3
Boot from local disk	BOOT /XMI:v/BI:w DUzz	Section 5.6
Boot from a VAXcluster	BOOT /R5:m/XMI:v/BI:w/NODE:xxyy DUzz	Section 4.6
Boot over the Ethernet	BOOT /XMI:v/BI:w ET0	Section 4.7
Boot VDS from TK	BOOT /R5:10 CSA1	Appendix C
Boot VDS from disk	BOOT /R5:10/XMI:v/BI:w DUzz	Appendix C
Conversational boot	BOOT /R5:1/XMI:v/BI:w DUzz	Appendix C
Boot from VMS shadow set	BOOT /R3:n/XMI:v/BI:w/NODE:xxyy DUzz	Section 5.6

[1] You can also designate the TK tape drive as your boot device when you use the BOOT CSA1 command described in Section 4.3.

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. Therefore, one processor must be designated as the boot processor (or primary processor) to become 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 5.17.2).

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. Normally, the boot processor's console program communicates with the console terminal and system control panel. However, you can force the boot processor to redirect console commands to a secondary processor by using the Z command (see Section 5.24).

Since the console terminal is connected only to the boot processor, any console commands that affect processor registers are executed by the boot processor. You can change the current boot processor by putting the control panel switch in Update and using the command:

```
SET CPU n
```

where *n* is the hexadecimal node number[1] of the processor you are designating to be the boot processor. After you enter the SET CPU command, the next console prompt that you receive comes from this newly designated boot processor. For this change to remain in effect after a system reset, the lower key switch must be set to Update when the SET CPU command is issued.

You can designate a processor to be ineligible to be the boot or primary processor by putting the control panel switch in Update and using the command:

```
SET CPU /NOPRIMARY n
```

where *n* is the hexadecimal node number of the processor you wish to designate as ineligible.

---

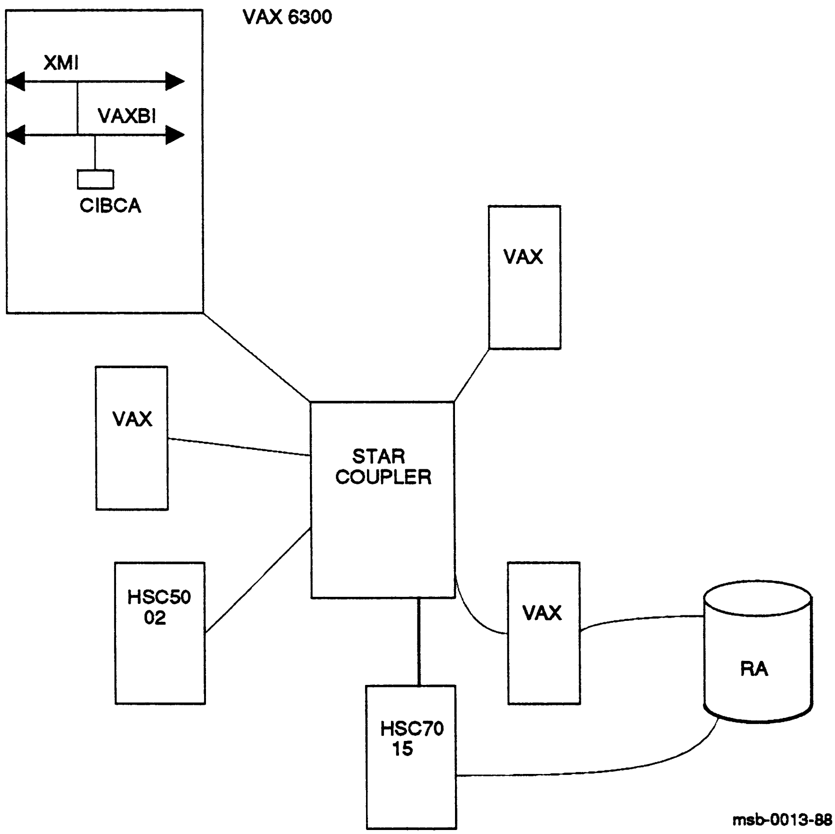
1 The node number and the ID number of the XMI slots are identical. See Section 2.3.

4.6 Booting from a VAXcluster

4.6.1 VAXcluster Boot Overview

The system supports booting from a VAXcluster. This allows all systems on a VAXcluster to share a system disk.

Figure 4–5: Booting from a VAXcluster



When you boot 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 on a tag on the front of the HSC cabinet. It 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 the pathname of the top level directory on the system.

Figure 4–5 shows a sample VAXcluster 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 normally want it to boot from the cluster system disk, you may set its default boot specification to the cluster disk using the SET BOOT console command (see Section 5.17.2).

4.6.2 Sample VAXcluster Boot

This section shows a sample boot from the VAXcluster configuration shown in Figure 4-5.

Example 4-2: Sample VAXcluster Boot

```
>>> SHOW CONFIGURATION ❶            ! Enter command.
                                     !
     Type              Rev           !
 1+  KA62B    (8001)  8002           ! Find the XMI and VAXBI address
 9+  MS62A    (4001)  0001           ! of the CIBCA, which is the
 D+  DWMBA/A  (2001)  0001           ! VAXcluster interface.
 E+  DWMBA/A  (2001)  0001           ! Here, the CIBCA connects
                                     ! to the system bus through the
                                     ! DWMBA/A at XMI node E. ❶
                                     !
 XBI D                               !
 1+  DWMBA/B  (2107)  0007           !
 6+  DEBNA    (410F)  023B           !
                                     !
 XBI E ❶                             !
 1+  DWMBA/B  (2107)  0007           !
 4+  CIBCA    (0108)  00A8 ❷         ! The CIBCA is VAXBI node 4. ❷
 6+  TBK70    (410B)  0307           !
                                     !

>>> BOOT /XMI:E❶ /BI:4❷ /R5:70000000❸ /NODE:0E02❹ DU0❺

Initializing system. ❻

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

    A   A   .   .   .   M   .   .   .   .   .   .   .   P       TYP
    o   o   .   .   .   +   .   .   .   .   .   .   .   +       STF
    .   .   .   .   .   .   .   .   .   .   .   .   .   E       BPD
    .   .   .   .   .   .   .   .   .   .   .   .   .   +       ETF
    .   .   .   .   .   .   .   .   .   .   .   .   .   B       BPD

.   .   .   .   .   .   .   .   .   +   .   .   .   .   +   .   XBI D +
.   .   .   .   .   .   .   .   .   +   .   +   .   .   +   .   XBI E +

    .   .   .   .   .   A1  .   .   .   .   .   .   .   .       ILV
    .   .   .   .   .   32  .   .   .   .   .   .   .   .       32 Mb

ROM = 4.1         EEPROM = 2.0/4.1       SN = SG91234567
```

Example 4-2 Cont'd. on next page

Example 4–2 (Cont.): Sample VAXcluster Boot

```
Loading system software. ❼
        .
        .
[operating system banner appears]
```

❶ SHOW CONFIGURATION displays the positions of modules and adapters. The DWMBA/A, which connects to the VAXBI with the CIBCA, is located at XMI node E. Enter this value E in the BOOT command as the argument to the /XMI qualifier.

❷ The CIBCA adapter is VAXBI node 4. Enter this value as the argument to the BOOT command qualifier /BI.

❸ In the BOOT command, the system root is the argument to the /R5 qualifier. In this case, SYS7 is specified.

❹ The arguments to /NODE are hexadecimal HSC node numbers 0E (decimal 14) and 02 (decimal 2). Listing two arguments in this manner 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.

❺ DU0 indicates that you are booting from a disk with unit number 0.

❻ System runs self-test.

❼ Booting of the operating system proceeds.

4.7 Booting over the Ethernet

4.7.1 Ethernet Boot Overview

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

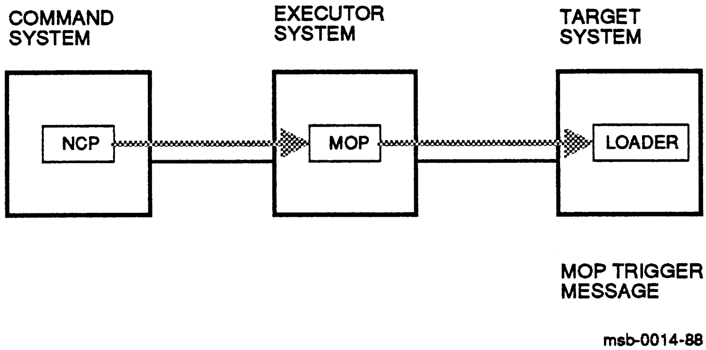
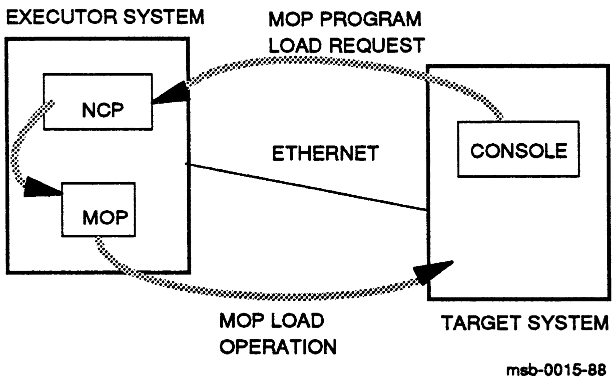


Figure 4–7: Target-Initiated Booting by Ethernet



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 in turn sends the command over the Ethernet to the executor system, which causes a boot in the target system, a VAX 6300. The target system loads its boot program from the boot device that is designated as the default. The target system must have its control panel key switch in the Auto Start position and be turned on for the boot to succeed. Commands are issued only from the command node, not from the target machine.

Target-initiated booting (Figure 4–7) is initiated by a command from the target VAX 6300 system. The BOOT command indicates the method of booting is Ethernet by using ET as the device type in the BOOT command:

```
BOOT /R5:m /XMI:v /BI:w /NODE:xxyy ET0
```

(See Section 4.4 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 6300 target node, the executor node's Maintenance Operations Protocol (MOP) volatile database requires an entry for the target node.

Section 4.7.2 shows an example of a target-initiated Ethernet boot.

4.7.2 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 Maintenance Operations Protocol volatile database on the executor node, and (3) issue a BOOT command from the target node. Example 4–3 through Example 4–5 show this procedure.

4.7.2.1 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 6300.

Example 4–3: Step 1, Ethernet Boot

```
>>> SHOW ETHERNET ❶             ! Enter command on target machine.
  XMI:D BI:6    08-00-2B-08-3D-64 ❷
                                 ! The DEBNA is BI node 6 of the VAXBI
                                 ! attached to the XMI at node D.  Its
                                 ! hardware Ethernet 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 4.7.2.2).

❷ The system reports the hardware Ethernet address for the DEBNA. The DEBNA is attached to the XMI at node D; the DEBNA adapter is node 6 on that VAXBI. The XMI/VAXBI address of the DEBNA is node D on the XMI and node 6 on the VAXBI.

The system manager of the target system and the system manager of the executor system assign a 6-letter node name and a network address for the target machine. In the example in Section 4.7.2, the assigned node name is TARGET, and the node network address is 9.961.

4.7.2.2 Step 2, Enter Information into Executor's MOP Volatile Database

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

Example 4–4: 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 address 9.961 ❷
NCP> set node TARGET hardware address 08-00-2B-08-3D-64 ❸
NCP> set node TARGET tertiary loader sys$system:tertiary_vmb.exe ❹

NCP> show node TARGET char ❺           ! 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            = BNA-0 ❻
    Hardware address           = 08-00-2B-08-3D-64
    Tertiary loader            = sys$system:tertiary_vmb.exe

NCP>                                   ! Prompt returns; show
NCP> sho circuit bna-0 char ❼          ! circuit characteristics.

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

    Circuit = BNA-0

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

Example 4–4 Cont'd. on next page

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

```
NCP>                                   ! Prompt returns; show
NCP> sho line BNA-0 char ❽             ! line characteristics.

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

    Line = BNA-0

    Receive buffers          = 6
    Controller               = normal
    Protocol                 = Ethernet
    Service timer            = 5000 ❽
    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 node name (in this example, TARGET) and the network address (9.961).

❸ Enter the hardware address found in Step 1, Section 4.7.2.1.

❹ Enter the tertiary loader pathname. The tertiary loader comes from a directory pointed to by MOM$load or from SYS$SYSTEM. Check your operating system documentation for details.

❺ Check your work, using the SHOW NODE command.

❻ The service circuit code is dependent on your hardware.

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

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

---

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

4.7.2.3 Step 3, Boot from the Target Node

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

Example 4–5: Step 3, Booting from the Target Node

```
>>> BOOT /XMI:D /BI:6 ET0          ! Enter BOOT command

Initializing system.

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

    A   A   .   .   M   M   .   .   .   .   .   .   P   P       TYP
    o   o   .   .   +   +   .   .   .   .   .   .   +   +       STF
    .   .   .   .   .   .   .   .   .   .   .   .   E   E       BPD
    .   .   .   .   .   .   .   .   .   .   .   .   +   +       ETF
    .   .   .   .   .   .   .   .   .   .   .   .   E   B       BPD

.   .   .   .   .   .   .   .   .   +   +   +   +   .   +   .   XBI D +
.   .   .   .   .   .   .   .   .   +   .   +   .   .   +   .   XBI E +

    .   .   .   .  A2  A1   .   .   .   .   .   .   .   .       ILV
    .   .   .   .  32  32   .   .   .   .   .   .   .   .       64 Mb

ROM = 4.1          EEPROM = 2.0/4.1       SN = SG91234567

Loading system software.

        .
        .
        .

[operating system banner appears]
```

Enter the BOOT command from the VAX 6300 console terminal in console mode. Use the XMI and VAXBI node numbers that you found in Step 1, Section 4.7.2.1, describing the path of the Ethernet controller.

Because the target VAX 6300 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.

Chapter 5

Console

The VAX 6300 console follows VAX standards for consoles, 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. Individual sections include:

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

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

- Sample console session

5.1 Description of Console

The VAX 6300 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: VAX 6300 Console

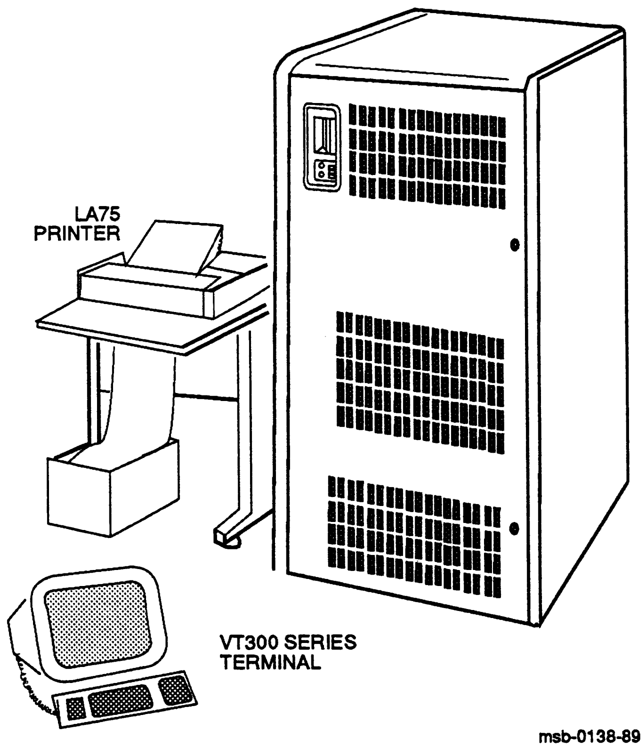


Table 5–1: VAX 6300 Console Parts and Functions

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.

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.

The console terminal port connection is located on the right rear I/O distribution panel above the AC power controller (see Section 2.5). 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).

You designate a terminal as the console terminal by connecting it to the console terminal port. See Section 2.5 for a description of the terminal port.

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, change memory interleave, and start and halt programs running on the processor.

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	CONTINUE, START
Deposit or change memory	FIND, DEPOSIT, INITIALIZE, SET MEMORY
Deposit or change registers	DEPOSIT
Designate primary processor	SET CPU
Display system parameters	SHOW
Examine memory	FIND, EXAMINE
Examine registers	EXAMINE
Execute ROM diagnostics	TEST
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 specific processor	CTRL/P, HALT, STOP
Store boot specifications	SET BOOT

You use the console terminal to control the system manually, correct errors, determine the status of machine circuits, 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.

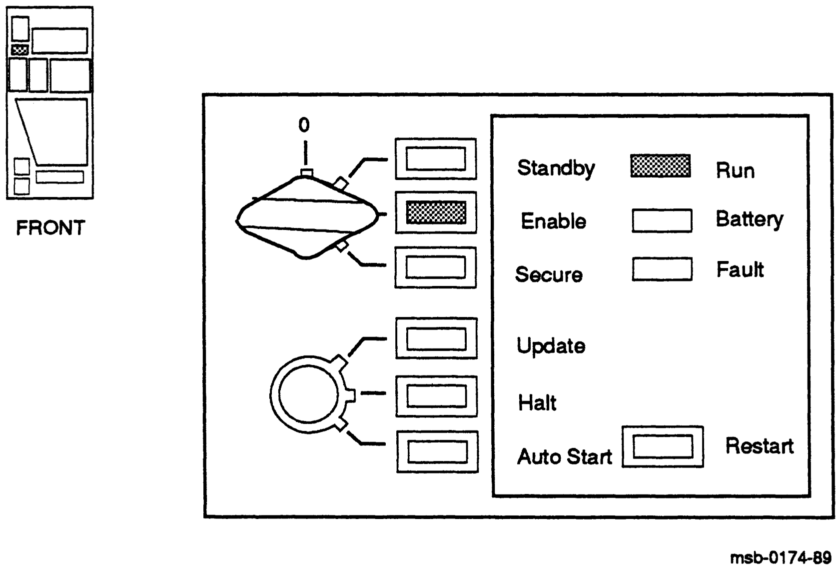
Nonprimary processors are called secondary processors. Secondary processors communicate with the console terminal through the primary processor when performing I/O during a console session. For information on designating a primary processor, see Section 4.5.

Appendix D lists 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 suspends 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. If you want to halt a secondary processor, you can issue a STOP command (see Section 5.20).

To resume program mode, use any of these commands:

CONTINUE	Resumes the program that was interrupted by the CTRL/P
START	Restarts the primary processor at a specified address
BOOT	Starts a bootstrap of the operating system

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

```
?nn <message>
```

where *nn* is a 2-digit number in hexadecimal format. These codes indicate an error or a halted processor. See Appendix B for a listing of these codes.

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 send the error message:

```
Node 5:  ?06 Halt instruction executed.

Node 5:  PC = 12345678
```

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.
CTRL/C	Causes the console to abort processing of a command.
CTRL/O	Causes 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. Hit BREAK repeatedly until the console prompt ">>> " appears, or use the SET TERMINAL command. The baud rates are 300, 600, 1200, 2400, 4800, 9600, 19200, and 38400. It is not recommended to run over 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 "$".

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	Appear in hexadecimal format.
No characters	Treated as a null command; no action taken.

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:

```
?36 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.

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

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.

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 6300.

5.6.1 BOOT Command Examples and Qualifiers

Example 5–1: BOOT Command

❶

```
>>> BOOT/XMI:E/BI:4 DU0        ! Boots from a disk with hex unit
                               ! number 0 connected via a
                               ! controller on VAXBI node 4,
                               ! accessible through the DWMBA/A
                               ! at XMI node E.
```

❷

```
>>> BOOT /XMI:E/BI:6/R5:70000000/NODE:0E02 DU0
                               ! Boots on a VAXcluster from an
                               ! HSC controller dual-ported at
                               ! unit numbers 0E and 02 with a
                               ! system root of SYS7.  See
                               ! Section 4.6.2.
```

❸

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

❹

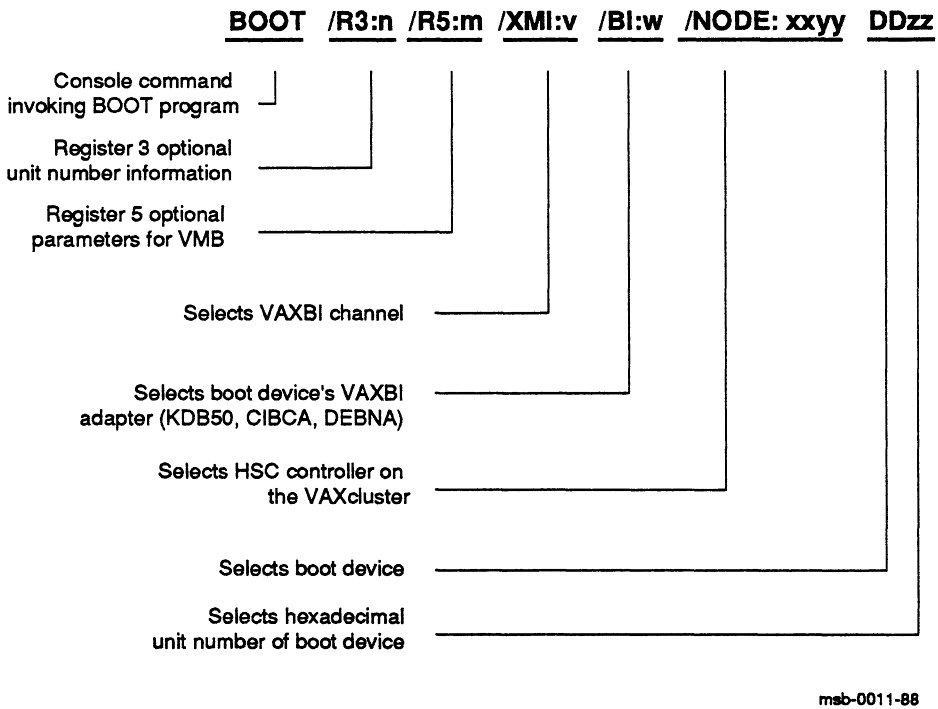
```
>>> BOOT                       ! Boots from the special boot
                               ! specification named DEFAULT.
```

Table 5–5: BOOT Command Qualifiers

Qualifier	Function
/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.
/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 C.
/X[MI]:number	Specifies the XMI node number of the node that connects the boot device.
/B[I]:number	Specifies the VAXBI node that connects the boot device. The /XMI qualifier must have selected a node containing a DWMBA/A.
/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.

5.6.2 BOOT Command Description

Figure 5–3: BOOT Command



The BOOT command syntax is:

```
B[OOT][/qualifier:number] [<device>]
```

where <device> is a string of the form *ddnn*. The variable *dd* is a 2-character mnemonic for the device type (MU for tape, DU for disk, ET for Ethernet, or CSA1 for TK tape), 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 being used.

The variable *nn* number indicates the node number of the module being specified by the qualifier. (See Table 5–5.) Number is a required argument to the qualifier. If you do not specify a number, you receive an error message in the form:

```
?21 Illegal command
```

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.

You can also use <device> for 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 the /XMI, /BI, /NODE, /R3, and /R5 qualifiers. 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 5.17.1.

If you omit <device>, 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.

5.7 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 5–2: CONTINUE Command

❶
```
$ ^P                            ! Stops processing on boot proces-
                                ! sor; processor enters console mode.
                                !
    ?02 External halt
    PC = 80159035               ! System responds with error message;
                                ! system has halted with address
                                ! 80159035 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 80159035.
```
❷
```
$ ^P                            ! CTRL/P stops boot processor;
                                ! processor enters console mode.
                                !
    ?02 External halt
    PC = 20044957               ! System responds with error message
                                ! that the system has halted with
                                ! address 20044957 in the program
                                ! counter (PC).
>>> [console session begins]    !
    .                           !
    .                           !
    .                           !
>>> C                           ! Processing resumes at 20044957,
                                ! where CTRL/P interrupted.
```

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 notifies you of the address in the PC when you halt the system by using CTRL/P. The message is of the form:

```
PC = hexadecimal number
```

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.

5.8 DEPOSIT

The DEPOSIT command stores data in a specified address.

Example 5–3: DEPOSIT Command

```
❶ >>> D/P 27 0                      ! Deposits the value of zero
                                    ! to physical address 27.
❷ >>> D/N:8 R0 FFFF                 ! Loads registers R0-R8 with FFFF.
❸ >>> DEPOSIT/P/B/N:1FF 0 0         ! Deposits zeros to the first
                                    ! 512 bytes of physical memory
                                    ! beginning with address 0.
```

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.
/N:<count>	Defines the address space as the first of a range.[1] <count>is a required value with /N.
/P	Defines the address space as physical memory.
/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.[2]
/W	Defines data size as a word.

[1]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).

[2]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 numeric 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.
- +, 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.

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.

5.9 EXAMINE

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

Example 5–4: EXAMINE Command

```
❶
  >>> E/N:8 R0                     !  Examines registers R0-R8.
❷
  >>> EXAMINE/P/B/N:1FF            !  Examines the first 512 bytes.
❸
  >>> EXAMINE/N:5/W/P -            !  Examines the previous word
                                   !  in the physical address space
                                   !  and the next five words.
❹
  >>> E/I 3E                       !  Examines the system ID register.
                                   !  System responds with output.
   I    0000003E    0A000002
```

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.
/N:<count>	Defines the address space as the first of a range.[1]
/P	Defines the address space as physical memory.

[1]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.

Table 5–7 (Cont.): EXAMINE Command Qualifiers

Qualifier	Meaning
/V	Defines the address space as virtual memory. All access and protection checking occur.[2]
/W	Defines data size as a word.

[2]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.

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

```
<TAB><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.
- M — Machine-dependent address space. This identifier is returned when the PSL is examined.

5.10 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.

Example 5–5: FIND Command

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

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

❸
```
>>> F/RP                       ! 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 the VAX 6300 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).

5.11 HALT

The HALT command is a null command for the VAX 6300 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 5–6: HALT Command

```
>>> HALT                      ! You enter the HALT command.
                              !
    ?26 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 the VAX 6300 system, HALT has no effect, because the boot processor is already halted as a requisite condition for console mode.

This command is included for the convenience of users who are familiar with its function in other VAX systems.

5.12 HELP

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

Example 5–7: HELP Command

❶

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

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

❷

```
>>> 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.
```

❸

```
>>> 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.
```

❹

```
>>> 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. Example 5–7 shows some of the HELP files.

The HELP command operates when the console program error messages are set in English mode (see Section 5.17.3). 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:

```
?5B
```

From the error messages in Table B–1, you can see that ?5B indicates that no help is available.

5.13 INITIALIZE

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

Example 5–8: INITIALIZE Command

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

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

❸
```
  >>> I                     ! Resets the entire system.
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   o   .   .   +   +   +   +   .   .   +   +   +   +       STF
    .   .   .   .   .   .   .   .   .   .   E   D   E   B       BPD
    .   .   .   .   .   .   .   .   .   .   +   +   +   +       ETF
    .   .   .   .   .   .   .   .   .   .   E   D   E   B       BPD

.   .   .   .   .   .   .   .   .   +   +   .   +   .   +   .   XBI D +
.   .   .   .   .   .   .   .   .   +   .   +   .   +   +   .   XBI E +

    .   .   .   .   A4  A3  A2  A1  .   .   .   .   .   .       ILV
    .   .   .   .   32  32  32  32  .   .   .   .   .   .       128 Mb

ROM = 4.1       EEPROM = 2.0/4.1        SN = SG91234567

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

Table 5–9: INITIALIZE Command Qualifiers

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

The INITIALIZE command syntax is:

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

where <node> is the XMI node number of the node to be initialized. If the node number you designate does not have a module in it, you receive an error message of the form:

```
?43 Unable to initialize node.
```

The node is reset by setting its node reset (NRST) bit. If <node> is omitted, the entire system is reset by using the IORESET processor register.

The INITIALIZE command and the Restart button on the system control panel perform the same function: they both reset the machine. See Section 6.2 through Section 6.7 for how to interpret self-test results.

5.14 REPEAT

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

Example 5–9: REPEAT Command

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

❷
```
>>> REPEAT TEST/RBD           ! Repeat TEST.
                              !
    ?21 Illegal command       ! Error message indicates this
                              ! is an illegal command.
```

The REPEAT command syntax is:

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

where <command> is any command other than REPEAT or TEST.

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.

You will receive an error message if you try to use REPEAT with either the TEST or REPEAT command.

5.15 RESTORE EEPROM

The RESTORE EEPROM command copies the TK tape's EEPROM image to the EEPROM of the boot processor. This command is used with the SAVE EEPROM command.

Example 5–10: RESTORE EEPROM Command

```
                   ! 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

?6D EEPROM Revision = x.x/y.y
?6F Tape image Revision = x.x/y.y

                   ! 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 update of EEPROM? (Y or N) >>> Y      — 30 seconds

>>>                ! 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 5.16). 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. Press the unload button. This rewinds the tape to the beginning, so that restoration proceeds at this point. When the operate handle light goes on, open the handle, reinsert the tape cartridge, and then close the handle. See Appendix A for additional information on the TK tape drive operation.

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/PRIMARY command. (See Section 4.5 and Section 5.17.2.)
5. At the prompt, enter RESTORE EEPROM. This operation overwrites any existing information on the TK cartridge, so be sure you have inserted an appropriate tape.
6. The console program queries you, requiring your confirmation to proceed with the RESTORE EEPROM operation.
7. Enter Y to indicate your intention to proceed. The restore process takes about 3 minutes to complete.
8. When the console prompt returns, the restore operation is complete. Restored information includes:

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

9. Rewind your TK tape by pushing the control button out, and remove the tape.
10. Reset the system using the INITIALIZE command or the control panel Restart button. All restored changes are visible following a system reset.
11. Use the SHOW command to verify the contents of the EEPROM.

NOTE: *To restore EEPROM contents on secondary processors, use the UPDATE command. See Section 5.23.*

5.16 SAVE EEPROM

The SAVE EEPROM command copies the EEPROM contents of the boot processor to the TK tape. This command is used with the RESTORE EEPROM command.

Example 5–11: SAVE EEPROM Command

❶

```
                              ! 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

?6C EEPROM saved to tape successfully.
>>>                           ! System confirms SAVE is complete.
```

❷

```
>>> SA E                      ! SAVE EEPROM to the TK tape.
                              !
?3D  Error initializing I/O device.
                              ! TK tape not initialized.
                              ! Reload.
                              ! Press the load/unload button 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 field 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 into the TK tape drive and press the control panel button. This rewinds the tape so it records from the beginning of the tape. See Appendix A for additional information on the TK 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 5.17.2.)
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 about 3 minutes 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:

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

8. Press the TK tape drive's load/unload button to 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 A.4).

5.17 The 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.

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

```
?3F  Key switch must be at "Update" to update EEPROM.
```

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 5.20 and Section 5.13.)

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.

This section describes the following SET commands:

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

5.17.1 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.

Example 5–12: SET BOOT Command

❶
```
                      ! Turn key switch to Update.
>>> SET BOOT DIAG /XMI:D /BI:4 /R5:10 DU0
>>>                   ! This creates a saved boot specification
                      ! called DIAG that boots the disk unit 0
                      ! from VAXBI node 4 on the VAXBI connected
                      ! to XMI node D.  The hexadecimal
                      ! value 10 is supplied for boot flags in R5.
                      !
                      ! SHOW BOOT command displays all saved
                      ! BOOT specifications.
>>> SHO BOOT
  DEFAULT  /BI:6 DU1
  DIAG     /R5:00000010 /XMI:D /BI:4 /R5:10 DU0
```

❷
```
>>> SET BOOT DIAG     ! Removes the boot specification saved
                      ! under the name DIAG.
                      !
>>> SHO BOOT          ! SHOW BOOT command confirms DIAG is
  DEFAULT  /BI:6 DU1  ! removed from saved BOOT specifications,
                      ! from example one above.
```

❸
```
>>> SET BOOT DEFAULT /XMI:E /BI:4 /NODE:0405 /R5:40000000 DU0
                      ! Sets the default boot device for the
                      ! system to be disk unit 0 (DU0) on the
                      ! VAXcluster.
                      !
                      ! SHOW BOOT command confirms DEFAULT
                      ! is changed in saved BOOT specifications.
>>> SHO BOOT
  DEFAULT  /R5:40000000 /XMI:E /BI:4 /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. You can use any name. Use DEFA, the first four letters of DEFAULT, for the specification which is used when you enter the BOOT command without a nickname.

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 field service representative 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 with battery backup 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 5.18 for additional information on the SHOW BOOT command.

5.17.2 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.

Example 5–13: SET CPU Command

❶
```
>>> SET CPU/PRIMARY 1       ! The processor at XMI node 1 may
                            ! become the primary processor at
                            ! the next system reset.
```
❷
```
>>> SE C/P 1                ! Same as above; this is
                            ! the abbreviation.
```
❸
```
>>> SE CPU 1                ! Processor at XMI node 1 immedi-
                            ! ately becomes the new primary
                            ! processor.  The next system prompt
                            ! is generated from the processor at
                            ! node 1.
```

Table 5–10: SET CPU Command Qualifiers

Qualifier	Meaning
/E[NABLED]	Processor is included in the system configuration and is eligible to become the boot processor.
/NOE[NABLED]	Processor is immediately excluded from the system configuration; START, BOOT, and CONTINUE commands are ignored.
/NEX[T_PRIMARY]	Processor will be the primary (boot) processor at the next system reset.
/P[RIMARY]	Processor will be eligible to be selected as the primary (boot) processor at the next system reset.
/NOP[RIMARY]	Processor will not be eligible to be selected as the primary (boot) processor at the next system reset.
No qualifier	Processor immediately becomes the new primary processor; the next system prompt comes from the new primary processor.

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 are in effect at the next system reset. The /ENABLE and /NOENABLE qualifiers modify the EEPROM and take effect immediately.

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 must issue SET CPU/PRIMARY commands for all processors.

The effect of the SET CPU command qualifiers is shown on line 4 of the system self-test, marked BPD (see Section 6.2).

Table 5–11: SET CPU Command Qualifiers' Effect on Self-Test Results (with Key Switch at Update)

Qualifier	BPD Value at Next Reset
/NEX[T_PRIMARY]	B for boot processor; must be chosen the boot processor at the next system reset.
/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.
None	B for boot processor.

5.17.3 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 B.

Example 5–14: SET LANGUAGE Command

❶
```
>>> SET LANGUAGE INTERNATIONAL ! Lower key switch must be in
                               ! Update position.
                               ! 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.
?02                            ! System error message indicates
                               ! external halt; message is in
                               ! INTERNATIONAL format.
```
❷
```
>>> 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.
?02  External halt             ! System error message indicates
                               ! external halt; message is ENGLISH
                               ! format.
```
❸
```
>>> SET LANGUAGE
?21 Illegal command            ! Illegal command; requires parameter.
```

Table 5–12: SET LANGUAGE Command Parameters

Parameter	Meaning
ENGLISH	Error messages reported as 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:

```
?5B
```

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:

```
?02 External halt
```

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

```
?02
```

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

```
?21
```

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

```
?21 Illegal command
```

5.17.4 SET MEMORY

The SET MEMORY command designates the method of interleaving memory.

Example 5–15: SET MEMORY Command

```
❶ >>> 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.

❷ >>> SE M/I:(9+7+6+5, 8, A)
                    ! Explicitly specifies what is created
                    ! as the system memory configuration.
```

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>
```

Interleaving memory modules produces faster memory access time. The console program automatically interleaves your memory modules for optimum performance. You change the interleave by using SET MEMORY.

An interleave set can only consist of one, two, four, or eight memory modules. 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.

This command modifies the configuration stored in the EEPROM. The new configuration takes effect the next time the system is reset or powered up. You cannot change the memory interleave without performing a system reset.

The default action interleaves memory so that the largest interleave factor is obtained for each group of like-sized 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 one module remains. 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
.   .   .   .   .   32  .   .   .   .   .   .   .   .   .   .   32 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 may contain one, two, four, or eight 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.

5.17.5 SET TERMINAL

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

Example 5–16: SET TERMINAL Command

```
❶ >>> SHOW TERMINAL                     ! Enter SHOW TERMINAL.
    /SCOPE    /SPEED: 1200   /BREAK
                                        ! System responds with
                                        ! the parameters stored for
                                        ! /SCOPE, /SPEED, and /BREAK.

❷ >>> SET TERM/HARDCOPY                 ! SET TERM changes system
                                        ! parameters.
                                        !
  >>> SHOW TERM                         ! Enter the command.
    /HARDCOPY /SPEED: 1200   /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. /BREAK is the default setting.
/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. /SCOPE is the default setting.
/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. /SPEED:1200 is the default setting.

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 and stores the new value in the EEPROM if you have the lower key switch set to Update. If you have not set any terminal characteristics in EEPROM, the defaults are /SPEED:1200, /BREAK, and /SCOPE. Recommended baud rate is 1200 or under.

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

5.18 SHOW

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

Example 5–17: SHOW Command

```
❶ >>> SHOW CONFIGURATION           ! Enter the command.
                                    !
     Type           Rev             ! System shows XMI device node,
  1+ KA62B   (8001) 8002            ! self-test status, device type,
  2+ KA62B   (8001) 8002            ! and revision register contents
  A+ MS62A   (4001) 0001            ! of the device.
  D- DWMBA/A (2001) 0001            !
  E+ DWMBA/A (2001) 0001            !
                                    !
  XBI D                             ! DWMBA at node D failed self-test.
  XBI E                             ! Shows the VAXBI connected through
  1+ DWMBA/B (2107) 0007            ! DWMBA/A on node E; gives VAXBI
  4+ CIBCA   (0108) 41C1            ! node, self-test status, device
  6+ TBK70   (410B) 0307            ! type, and revision register con-
                                    ! tents of the device. See Appendix
                                    ! E for device code tables.

❷ >>> SHOW CPU                      ! Gives the XMI node of current
                                    ! primary processor and status of
   Current Primary: 1               ! each processor.  CPU at node 1 is
   /NOENABLED-                      ! boot processor.  The CPU at node 2
   /NOPRIMARY- 2                    ! is ineligible to become a primary
                                    ! processor.

❸ >>> SHOW MEMORY                   ! Displays the memory lines from the
                                    ! system self-test.
  F  E  D  C  B  A  9  8  7  6  5  4  3  2  1  0   NODE #

  .  .  .  .  .  A1 .  .  .  .  .  .  .  .  .  .   ILV
  .  .  .  .  .  32 .  .  .  .  .  .  .  .  .  .   32 Mb
   /INTERLEAVE:DEFAULT

❹ >>> SHOW TERMINAL                 ! Gives baud rate and terminal param-
                                    ! eters created by using SET TERM.
    /HARDCOPY /SPEED: 1200 /BREAK

❺ >>> SHOW BOOT                     ! Shows the boot commands and their
                                    ! nicknames created by using SET BOOT.
    DEFAULT /XMI:E /BI:4 DU0
    DSK1    /BI:6 DU1
    HSC     /R5:40000000 /XMI:D /BI:2 /NODE:0405
```

Example 5–17 Cont'd. on next page

Example 5–17 (Cont.): SHOW Command

```
❻ >>> SHO ETHERNET                   ! Displays the XMI and BI node numbers
                                     ! for the DEBNA adapter and Ethernet hard-
  XMI:E BI:6 08-00-2B-08-3D-64       ! ware address of system running the
                                     ! console program.
```

The SHOW command syntax is:

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

where <object> is one of these commands:

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

All system responses to the SHOW command are formatted and displayed on the console terminal.

SHOW ALL displays all the information given in Example 5–17, in the order shown in Example 5–17.

SHOW BOOT lists all BOOT commands and their nicknames that have been created by using the SET BOOT command.

SHOW CONFIGURATION displays the hardware device type and revision level for each XMI and VAXBI node and indicates whether the node passes or fails self-test. See Appendix E for device type code assignments.

SHOW CPU displays the /NOENABLED and /NOPRIMARY values for each node. The SET CPU command assigns these values.

SHOW ETHERNET locates all Ethernet adapters on the system and displays their Ethernet hardware addresses.

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. 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 presents the current parameters set for the console terminal: baud rate and terminal characteristics.

5.19 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.

Example 5–18: START Command

```
❶ $ ^P                          ! CTRL/P stops processing;
                                ! system enters console mode.
                                !
      ?02 External halt         ! System responds with error
          PC = 80159035         ! message that the system has
                                ! halted with address 80159035
                                ! in the program counter (PC).
  >>> [console session begins]  !
      .                         !
      .                         !
      .                         !
  >>>  START                    ! Starts the system at
                                ! address 80159035.

❷ $ ^P                          ! Stops processing; system enters
                                ! console mode.
                                !
      ?02 External halt         ! System responds with error
          PC = 20044957         ! message that the system has
                                ! halted with address 20044957
                                ! 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.

5.20 STOP

The STOP command halts the specified XMI node by setting its XMI Bus Error Register (XBER) <NHALT> bit. If the target node is a processor, the processor enters console mode.

Example 5–19: STOP Command

```
❶ >>> STOP/BI:6 D          ! Stops the adapter at node 6 on the
                           ! VAXBI accessible through the
                           ! DWMBA/A at node D.
❷ >>> STO/B:6  D           ! Same as above command.
❸ >>> STOP 6               ! Stops the adapter at node 6 on the
                           ! XMI. If the adapter at node 6 is a
                           ! DWMBA/A, then all nodes on the
                           ! VAXBI connected through node 6 are
                           ! also stopped.
```

Table 5–15: 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.
<node>	Specifies an XMI node, and stops the processor or DWMBA/A at the given node number.

The STOP command syntax is:

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

where <node> is the node number of the XMI node to be halted.

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

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

if <node> is a DWMBA/A, all nodes on the VAXBI are halted. If <node> is a processor module, then only that node is halted.

A STOP affects only the current activity on the node. The module is reset at the next self-test. If you stop a processor that is currently running, you receive this message:

```
Noden:  ?02 External halt
Noden:       PC = nnnnnnnn
```

where *n* is the node you stopped, and *nnnnnnnn* 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.

The STOP command does not apply to memories.

5.21 TEST

The TEST command passes control to the system self-test diagnostics, which can start a reset in part of the system.

Example 5–20: TEST Command

```
>>> TEST/RBD                    ! Requests testing of ROM-based
                                ! diagnostics.
                                !
RBD4> ST 0/TR                   ! New prompt indicates you are in
                                ! the RBD monitor program working
                                ! from the device with node number 4.
                                ! Command starts RBD test 0, tracing
                                ! results. Results follow:
;XCPST        0.08

; T0001  T0003  T0004  T0005  T0006  T0007  T0008  T0009  T0010  T0011
; T0012  T0013  T0014  T0015  T0016  T0017  T0018  T0019  T0020  T0021
; T0022  T0023  T0024  T0025  T0026  T0027  T0028  T0029  T0030  T0031
; T0034

;      P       4     8001       1
;00000000 00000000 00000000 00000000 00000000 00000000 00000000

RBD4> ^Z                        ! You enter CTRL/Z to return
                                ! to console mode.

?06 Halt instruction executed.
    PC = 20060117               ! System error message indicates
                                ! RBD program has been halted.
                                !
>>>                             ! Console prompt returns.
```

Table 5–16: 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, control transfers to the self-test diagnostics. See the *VAX 6300 Options and Maintenance* manual for more information on diagnostics.

5.22 UNJAM

The console program accepts the UNJAM command. However, UNJAM has no effect on the system, since the VAX 6300 system does not have an independent I/O bus reset option.

Example 5–21: UNJAM Command

```
>>> 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 in the VAX 6300 system, since the only bus reset is accomplished with a full system reset.

5.23 UPDATE

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

Example 5–22: UPDATE Command

❶
```
>>> UPDATE 1                !Lower key switch must be set to Update.
?64 Operation only applies to secondary processors.
                            ! The module at node 1 is not a
                            ! secondary processor.
```

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

❸
```
>>> UPDATE 2                ! Update the processor at node 2.
>>>                         ! There is a pause while this
                            ! command executes.  When the
                            ! console prompt returns, update
                            ! is complete.
```

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

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 receiving the contents of the primary processor's EEPROM. The secondary processor's EEPROM is updated even if the processor is set to NOENABLED.

The UPDATE command copies the parameters that can be set plus any additional information stored in the EEPROM of the boot processor. UPDATE should be issued following any field service installation of a new EEPROM patch level. Updated information includes:

- System serial number
- Systemwide console parameters

  Baud rate
  Memory interleave specifications
  Console terminal specifications

- Saved boot specifications
- Diagnostic patches
- Console patches
- Boot primitives

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

```
?3F Key switch must be at "Update" to update EEPROM.
```

You could use the UPDATE command to update the EEPROM of a secondary processor after restoring the primary processor's EEPROM from the tape drive.

5.24 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.

Example 5–23: Z Command

```
>>> Z 6                             ! Connect to the CPU at node 6
                                    ! on the XMI.
?33 Z connection successfully started.
                                    ! System response; console
6>>                                 ! prompt indicates the new
                                    ! processor node: "6>>".
                                    !
6>> D/P 0 12345678                  ! Deposits data 12345678 to
                                    ! physical address 0.
                                    !
6>> ^P                              ! End connection to CPU at node
                                    ! 6 with a CTRL/P command.
                                    !
?31 Z connection terminated by ^P.
                                    ! System response; console prompt
>>>                                 ! indicates you are in console
                                    ! mode on the boot processor.
                                    !
>>> E/P 0                           ! Examines physical address zero;
  P 00000000  12345678              ! shows data 12345678 was
                                    ! successfully deposited.
                                    !
>>> Z/BI:6 E                        ! Connect to adapter at VAXBI node 6
                                    ! through DWMBA/A in XMI slot E.
?33 Z connection successfully started.
6>>
```

Table 5–17: Z Command Qualifiers

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

The Z command syntax is:

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

where <node> is required and is the number of the target node. When used with the /BI qualifier, <node> specifies the target node on the VAXBI. When no qualifier is present, <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 VAXBI 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. This modified prompt remains until you use a CTRL/P command to return to the primary processor.

5.25 !

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.

Example 5–24: ! Command

❶
```
>>> ! THIS IS A COMMENT
>>>
```

❷
```
>>> 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:

```
?36 Command too long.
```

5.26 Sample Console Session

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

    A   A   .   .   M   M   .   .   .   .   .   .   P   P       TYP
    o   o   .   .   +   +   .   .   .   .   .   .   +   +       STF
    .   .   .   .   .   .   .   .   .   .   .   .   E   B       BPD
    .   .   .   .   .   .   .   .   .   .   .   .   +   +       ETF
    .   .   .   .   .   .   .   .   .   .   .   .   E   B       BPD

.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   XBI D -
.   .   .   .   .   .   .   .   .   +   .   +   +   .   +   .   XBI E +

    .   .   .   .   A2  A1  .   .   .   .   .   .   .   .       ILV
    .   .   .   .   32  32  .   .   .   .   .   .   .   .       64Mb

ROM = 4.1         EEPROM = 2.0/4.1         SN = SG91234567 ❹
```

❷
```
>>> EX/N:5 R0
G 00000000  FFFFFFFF
G 00000001  20140648
G 00000002  00000000
G 00000003  00000010
G 00000004  20140800
G 00000005  00FFF5F8
```

❸
```
>>> SHOW CONFIGURATION

     Type              Rev
 1+  KA62B    (8001)  8002
 2+  KA62B    (8001)  8002
 9+  MS62A    (4001)  0002
 A+  MS62A    (4001)  0002
 D-  DWMBA/A  (2001)  0002
 E+  DWMBA/A  (2001)  0002

 XBI D
 XBI E
 1+  DWMBA/B  (2107)  0007
 3+  DRB32    (0101)  0001
 4+  KDB50    (010E)  0F1C
 6+  TBK70    (410B)  0307
```

❹
```
>>> BOOT /XMI:E/BI:5 DU3
```

❺
```
?06 Halt instruction executed.
   PC = 2004517A
Failure.
```

❻
```
>>> BOOT /XMI:E/BI:5 DU3

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

  VAX/VMS Version 5.2
```

❼
```
PLEASE ENTER DATE AND TIME (DD-MMM-YYYY  HH:MM) ❽ 16-JAN-1989 12:43
```

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 console prompt indicates that the terminal is in console mode. Enter an EXAMINE command to examine the contents of register 0 and five additional registers. Output displays the contents for R0, R1, R2, R3, R4, and R5, respectively.

❸ 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.

The second paragraph of output gives information on the VAXBI connected through the DWMBA/A located on node E of the XMI. It indicates VAXBI node number, self-test status, the code for the VAXBI device type, and the contents of the revision register of each device. See Appendix E for more information on device type code assignments.

❹ Enter BOOT command, using a boot device of a disk (DU3) located through node 5 of the VAXBI that is connected by a DWMBA/A on node E of the XMI.

❺ System issues an error message. Because you did not get to the operating system banner, 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.

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

❼ The operating system software (in this case, VMS) prompts for date and time information.

❽ Enter date and time information in requested format. The operating system continues to load.

Chapter 6

System Self-Test and Troubleshooting

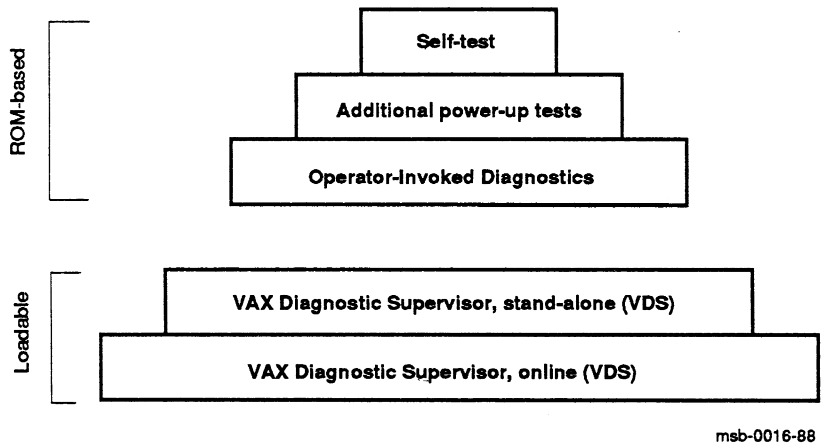
This chapter discusses the VAX 6300 diagnostics and self-tests; it includes the following sections:

- Diagnostics design
- Sample self-test
- Self-test lines NODE #, TYP, and STF
- Self-test lines BPD and ETF
- Self-test lines XBI
- Self-test lines ILV and Mb
- Self-test identification line
- Troubleshooting during booting
- Forcing a boot processor

6.1 Diagnostics Design

The VAX 6300 system has two types of diagnostics: (1) ROM-based Diagnostics (RBDs) which include self-test, additional power-up tests, and operator-invoked diagnostics and (2) loadable diagnostics which run under the VAX Diagnostic Supervisor (VDS) in stand-alone or on-line mode. The VDS tests are used by DIGITAL field service representatives.

Figure 6–1: Diagnostics Design



At power-up or system reset, the processors and memory modules run their own self-tests. The processor self-test completes in 10 seconds, and the memory test completes within 60 seconds.

While self-test is running, the Fault light on the front panel is on (see Section 3.5). In a fully working system, self-test completes within 10 to 15 seconds, the Fault light goes off, and the console printout of self-test begins. If the Fault light goes out but no console message is printed, check the terminal power, connection, and characteristics. If the Fault light stays lit, wait 1 minute and the console message should begin. If the console message does not begin, see Section 6.9 and call your field service representative.

Each XMI module has a yellow LED on the outer edge of the module that lights when the module passes self-test. Results of the complete system self-test, including I/O devices on the VAXBI bus, are written to the console terminal (see Section 6.2). You can view the lights from the front of the cabinet with the front door open. If a module fails self-test, its yellow LED does not light, and a failure is indicated on the self-test results.

The VAXBI follows a similar self-test procedure, with each node on the VAXBI running its own self-test. The VAXBI adapters for the Ethernet port and the TK tape drive adapter run through their self-test at this point.

Next, having passed self-test, the processors run additional tests on the processor and the memory modules. The boot processor then tests the DWMBA/A and DWMBA/B modules. (See Section 4.5 for information on the boot processor.)

As each self-test runs, the results are written to the console terminal. When all self-tests complete and all self-test results have printed, the system enters console mode unless Auto Start is enabled (see Section 3.3). From console mode, you boot the operating system or use RBD mode to run diagnostics.

6.2 Sample Self-Test

The VAX 6300 has self-test diagnostics in ROM on each processor. These self-tests check each module at power-up, when the system is reset, during booting, or when the self-tests are invoked from the RBD monitor program. Self-test results are printed on the console terminal, as shown in Example 6–1.

Example 6–1: Self-Test Results

```
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   o   .   .   +   +   +   +   .   .   +   +   +   -       STF    ❸
    .   .   .   .   .   .   .   .   .   .   E   B   D   E       BPD    ❹
    .   .   .   .   .   .   .   .   .   .   +   -   +   -       ETF    ❺
    .   .   .   .   .   .   .   .   .   .   B   E   D   E       BPD

.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   XBI D -❻
.   .   .   .   .   .   .   .   .   +   +   +   .   -   +   .   XBI E +

    .   .   .   .   B2  B1  A2  A1  .   .   .   .   .   .       ILV    ❼
    .   .   .   .   32  32  32  32  .   .   .   .   .   .       128 Mb ❽

ROM = 4.1 ❾      EEPROM = 2.0/4.1 ❿       SN = SG91234567 ⓫

>>>
```

Following power-up and system reset, self-test runs and results are printed on the console terminal. Self-test results are also printed during booting. Once self-test begins, there is no way to interrupt until the process completes.

The self-test printout in Example 6–1 reflects the system configuration listed in Table 6–1. Each numbered line 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.

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

Module	XMI Node Number	Module Type
KA62B	1	Processor; fails first self-test.
KA62B	2	Processor; disabled from being boot processor.
KA62B	3	Processor; boot processor at first self-test, fails extended self-test.
KA62B	4	Processor; operating as boot processor.
MS62A	7	Memory (32 Mbytes); interleaved with memory at node 8 by a SET MEMORY console command.
MS62A	8	Memory (32 Mbytes); interleaved with memory at node 7 by a SET MEMORY console command.
MS62A	9	Memory (32 Mbytes); interleaved with memory at node A by a SET MEMORY console command.
MS62A	A	Memory (32 Mbytes); interleaved with memory at node 9 by a SET MEMORY console command.
DWMBA/A	D	I/O adapter that failed self-test.
DWMBA/A	E	I/O adapter leading to a VAXBI bus that has passing modules at nodes 1, 4, 5, and 6; the module at node 2 on the VAXBI failed self-test. These modules might be a DEBNA Ethernet adapter, a CIBCA adapter, a TBK70 adapter for a TK tape drive, and a DWMBA/B.

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

The first 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.

Example 6–2: Self-Test Results: NODE #, TYP, and STF

```
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   o   .   .   +   +   +   +   .   .   +   +   +   -       STF     ❸
    .   .   .   .   .   .   .   .   .   .   E   B   D   E       BPD
    .   .   .   .   .   .   .   .   .   .   +   -   +   -       ETF
    .   .   .   .   .   .   .   .   .   .   B   E   D   E       BPD

.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   XBI D -
.   .   .   .   .   .   .   .   .   +   +   +   .   -   +   .   XBI E +

    .   .   .   .   B2  B1  A2  A1  .   .   .   .   .   .       ILV
    .   .   .   .   32  32  32  32  .   .   .   .   .   .       128 Mb

ROM = 4.1        EEPROM = 2.0/4.1        SN = SG91234567
>>>
```

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

❶ The NODE # line lists the node numbers on the XMI and VAXBI buses. 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 Example 6–2).

Note that XMI entries use only slots 1 through E, while the VAXBI has entries in slots 0 through F. The XMI has 14 slots, and the slot and node numbers are identical. So the position of the nodes on the self-test printout reflects a map of the physical position of the modules in the XMI card cage.

The system VAXBI bus has six slots. The VAXBI slot and node numbers are not identical. Nodes may be numbered from 0 through F on the VAXBI. Node plugs (labeled 1 to 6) located in the backplane of the system VAXBI are used to identify the number of a node.

❷ The second line in the printout indicates the type (TYP) of module at each XMI node:

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

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

+ (pass)
– (fail)
o (does not apply)

Since the DWMBA adapters do not have a module-resident self-test, their entry for the STF line is always "o."

6.4 Self-Test Lines BPD and ETF

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

Example 6–3: Self-Test Results: BPD and ETF

```
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  o  .  .  +  +  +  +  .  .  +  +  +  -      STF
   .  .  .  .  .  .  .  .  .  .  E  B  D  E      BPD   4
   .  .  .  .  .  .  .  .  .  .  +  -  +  -      ETF   5
   .  .  .  .  .  .  .  .  .  .  B  E  D  E      BPD   4

.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .   XBI D -
.  .  .  .  .  .  .  .  .  +  +  +  .  -  +  .   XBI E +

   .  .  .  .  B2 B1 A2 A1 .  .  .  .  .  .      ILV
   .  .  .  .  32 32 32 32 .  .  .  .  .  .      128Mb

ROM = 4.1        EEPROM = 2.0/4.1        SN = SG91234567
>>>
```

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

❹ The BPD line 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 5.17.2).

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

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

❺ During extended self-test (ETF) all processors run additional CPU tests involving memory. In Example 6–3, results printed at this ETF line indicate:

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

6.5 Self-Test Lines XBI

The XBI lines of the self-test printout provide information on the node numbers and self-test status for modules in the VAXBI card cages connected to the XMI through DWMBA/A modules.

Example 6–4: Self-Test Results: XBI

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

 .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   XBI D - (6)
 .   .   .   .   .   .   .   .   .   +   +   +   .   -   +   .   XBI E +

     .   .   .   .   B2  B1  A2  A1  .   .   .   .   .   .       ILV
     .   .   .   .   32  32  32  32  .   .   .   .   .   .       128Mb

 ROM = 4.1       EEPROM = 2.0/4.1          SN = SG91234567
 >>>
```

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

❻ The XBI lines indicate the VAXBI self-test information and VAXBI device self-test results.

In Example 6–4, one VAXBI was accessed through the DWMBA/A on XMI node D and has failed (–) self-test (XBI D –). The other VAXBI was accessed through XMI node E and has passed (+) self-test (XBI E +).

When a DWMBA 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 DWMBA fails self-test, the failure is reported, and the VAXBI device self-tests are not displayed.

The SHOW CONFIGURATION command gives you additional information on the VAXBI nodes, listing their device type numbers (see Section 5.18).

6.6 Self-Test Lines ILV and Mb

The seventh and eighth lines of the self-test printout provide additional information on the memory modules. The ILV line details the interleaving of the memories, and the Mb line gives the size in Mbytes of each memory module and the total size of system memory.

Example 6–5: Self-Test Results: ILV and Mb

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

.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   XBI D -
.   .   .   .   .   .   .   .   .   +   +   +   .   -   +   .   XBI E +

    .   .   .   .   B2  B1  A2  A1  .   .   .   .   .   .       ILV    ❼
    .   .   .   .   32  32  32  32  .   .   .   .   .   .       128Mb  ❽
ROM = 4.1      EEPROM = 2.0/4.1            SN = SG91234567
>>>
```

Passing Memory

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

❼ This ILV line contains a memory interleave value (ILV) for each memory. If you have more than one interleave set, each set is indicated by a different letter.

In Example 6–5, a SET MEMORY command was used to create two interleaved sets of two 32–Mbyte memories each (see Section 5.17.4). 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
    .   .   .   .   .   32  32  32  32  .   .   .   .   .   .        128Mb

    /INTERLEAVE:DEFAULT
```

❽ The line after the ILV line displays the size of each memory module configured in the system and gives the total Mbytes of system memory. In Example 6–5, the total is 128 Mbytes.

Failing Memory

When a memory module does not pass its self-test, the module undergoes extensive testing and failing addresses 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
    .   .   .   .   .   32  32  32  32  .   .   .   .   .   .        128Mb

    /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
    .   .   .   .   .   .   32  32  32  .   .   .   .   .   .        96Mb

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

6.7 Self-Test Identification Line

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

Example 6–6: Self-Test Results: Identification Line

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

.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   XBI D -
.   .   .   .   .   .   .   .   .   +   +   +   .   -   +   .   XBI E +

    .   .   .   .   B2  B1  A2  A1  .   .   .   .   .   .       ILV
    .   .   .   .   32  32  32  32  .   .   .   .   .   .       128Mb

ROM = 4.1 (9)      EEPROM = 2.0/4.1 (10)       SN = SG91234567 (11)
>>>
```

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

❾ The ROM information indicates the version of read-only memory that is installed on the processors in this system. In Example 6–6, all processors have version 4.1 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 field service representative to fix the ROM levels.

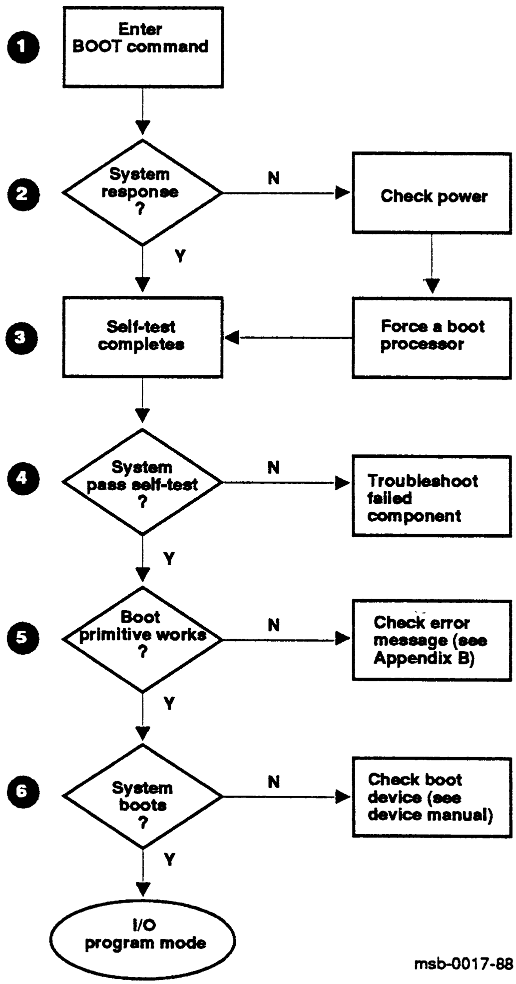
❿ The EEPROM information gives the version of EEPROM that your processors have and the patch level. In Example 6–6, the processors have level 2.0 EEPROMs with a 4.1 patch level. If you are running processors whose EEPROMs do not match, you receive an error message. Contact your field service representative.

⓫ SN gives the system serial number. The serial number of the system is carried on each processor and on the cabinet. In general, different ROM levels on processors will prevent you from booting the operating system.

6.8 Troubleshooting During Booting

When booting fails, you can check several parameters.

Figure 6–2: Troubleshooting Booting



If the boot procedure fails, check through the steps shown in Figure 6–2.

❶ 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 5.18.)

❷ 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 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 6.9).

❸ 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.

❹ 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. For troubleshooting other failed modules, run ROM-based diagnostic tests or call your field service representative.

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

❺ 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 is not working, you receive an error message, most likely:

```
?06 Halt instruction executed.
```

(For a list of error messages, see Appendix B.) If the boot primitive fails, this indicates that the boot primitive program is not able to reach the boot device. Check to see if the boot device passed self-test. Is the boot device connected to the system? Is it powered up?

❻ 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 all these steps fail, call your field service representative.

6.9 Forcing a Boot Processor

The system may hang at power-up 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, and no console commands have any effect. After you check electrical and control panel connections, force a boot processor.

Example 6–7: Forcing a Boot Processor

```
[  No self-test results or system prompt appear on console terminal.  ]

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

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

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

.   .   .   .   .   .   .   .   .   +   +   .   .   .   +   .   XBI D +
.   .   .   .   .   .   .   .   .   +   +   +   .   -   +   .   XBI E +

    .   .   .   .   A2  A1  .   .   .   .   .   .   .   .       ILV
    .   .   .   .   32  32  .   .   .   .   .   .   .   .       64Mb

ROM = 4.1          EEPROM = 2.0/4.1        SN = SG91234567

>>>                                 ! 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 you do not receive self-test results when you power up the system, followed by a system prompt, the system may be hung. When the system is hung, you cannot receive any response from the console. At this point, force a boot processor.

The system hangs for one of two reasons:

- No boot processor can be found. All processors are disabled from becoming the boot processor or fail self-test.
- No memory can be located.

If the problem is caused by inability to designate a boot processor, the system does not report self-test but hangs with no console terminal response at all. For example, in Example 6–7, there are three processors. Two of the processors at nodes 2 and 3 have been set, using the SET CPU command and qualifiers, to be ineligible to become the boot processor. The third processor on node 1 is eligible to be a boot processor, but fails self-test. At this point, the system cannot find a boot processor and hangs.

If you enter:

```
>>n
```

where *n* is the system bus node number of a processor, the console program is forced to overlook the bits set in the processors by the SET CPU command and to designate the processor at node *n* as the primary processor. Processor *n* next executes self-test results to the console terminal. The >>*n* sequence is not echoed. Example 6–7 forced the processor at node 3 to become the boot processor.

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. Only the SHOW CPU command gives the accurate state of the processor bits.

In Example 6–7, 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 the system hang is caused by inability to locate sufficient memory, self-test completes and the first four lines of the self-test print to the console terminal. You receive the output of lines NODE #, TYP, STF, and BPD. Each processor then attempts to run extended test, but finds no memory present. Each processor reports insufficient memory, and prints the ETF and BPD lines. System hangs here until the >>n command is entered. When a system hang is caused by memory 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
TK70 Tape Drive Instructions

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

A.1 Controls and Indicators

The TK70 tape drive has three lights, a beeper, an unload button, and a cartridge insert/release handle. Table A–1 lists the functions of TK tape drive controls and indicators that are shown in Figure A–1.

Figure A–1: TK Tape Drive

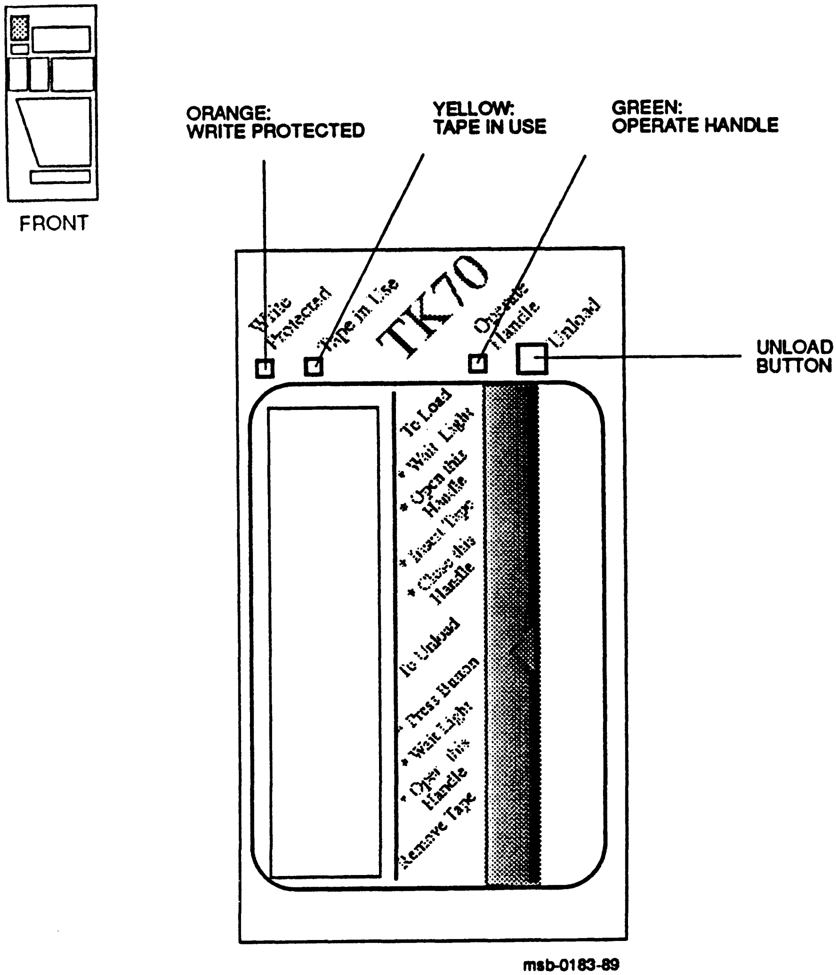


Table A–1: TK70 Light Summary

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

[1]The 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.

A.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 three lights blink, the loading has failed.)*

A.3 Unloading a Tape

To unload a tape, follow these steps:

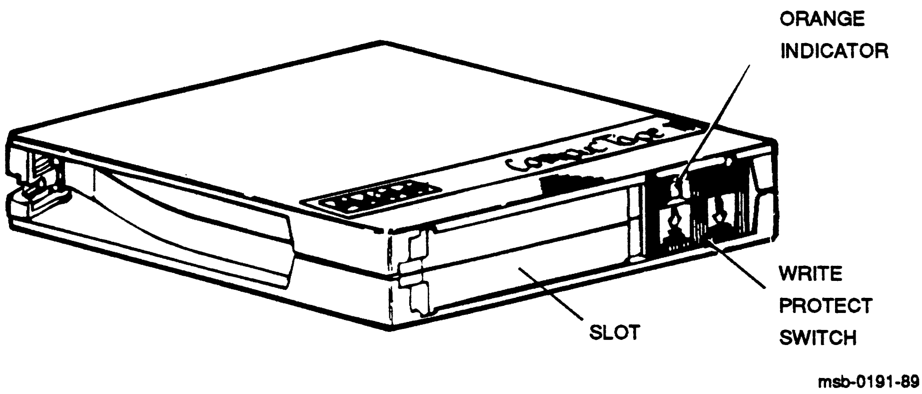
1. Press the unload button or execute an appropriate operating system unload command. The yellow light blinks as the tape rewinds.
2. When the green light turns on and the beep sounds, pull the handle to the open position. The cartridge will partially eject.
3. Remove the cartridge.
4. Push the handle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

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

A.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 A–2.

Figure A–2: TK Tape Cartridge



A.5 Labeling a Tape Cartridge

To label your tape cartridge:

- Write your identifying information on the label. Note the recording density: TK70 = 296 Mbytes.
- Put the label into the slot on the front of the cartridge. See Figure A–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.

A.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° and 40°C (50° to 104°F).
- Keep tape cartridges away from sources of electromagnetic interference, such as terminals, motors, and video or X-ray equipment.

Appendix B
Console Error Messages

Table B–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 B–2 lists the standard console error messages ?20 through ?7C.

Table B–1: 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 B–1 (Cont.): 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 B–2: 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.

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

Error Message	Meaning
?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.
?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.

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

Error Message	Meaning
?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.
?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.

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

Error Message	Meaning
?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 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.
?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. See Section 6.9.
?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.

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

Error Message	Meaning
?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.
?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.

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

Error Message	Meaning
?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 which 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.
?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.

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

Error Message	Meaning
?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.
?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.

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

Error Message	Meaning
?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.
Loading system software.	The console is attempting to load the operating system in response to a BOOT command, powerup, 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 C
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 C–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. If bit 4 is also set, the VAX Diagnostic Supervisor should start and prompt you for devices to test.
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 immediately after enabling mapping.
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 C–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.
15	Used by the VAX Diagnostic Supervisor.
31:28	Specifies the top-level directory number for system disk.

Table C–2: R5 Bit Functions for ULTRIX

Bit	Function
0	Forces ULTRIXBOOT to prompt the user for an image name (the default is VMUNIX). If bit 4 is also set, the user is prompted for the Diagnostic Supervisor image name.
1	Boots the ULTRIX kernel image in single-user mode.
3	Must be set, and R4 must be zero.
16	Must be set.

Appendix D
Console Commands

There are 24 console commands. Most commands are standard console commands. The commands that have been designed specifically for the VAX 6000 family include RESTORE EEPROM, SAVE EEPROM, SET BOOT, SET CPU, and Z. Chapter 5 gives a full description of each command, its qualifiers, and examples. Table D–1 gives a summary of the commands.

Table D–1: Console Commands

Command	Function
BOOT	Initializes the system, causing a self-test, and begins the boot program.
CONTINUE	Begins processing at the address where processing was interrupted by a CTRL/P console command.
DEPOSIT	Stores data in a specified address.
EXAMINE	Displays the contents of a specified address.
FIND	Searches main memory for a page-aligned 256-Kbyte block of good memory or for a restart parameter block.
HALT	Null command; no action is taken since the processor has already halted in order to enter console mode.
HELP	Prints explanation of console commands; operates like the HELP command in VMS.
INITIALIZE	Performs a system reset, including self-test.
REPEAT	Executes the command passed as its argument.
RESTORE EEPROM	Copies the TK tape's EEPROM contents to the EEPROM of the processor executing the command.
SAVE EEPROM	Copies to the TK tape the contents of the EEPROM of the processor executing the command.
SET BOOT	Stores a boot command by a nickname.

Table D–1 (Cont.): Console Commands

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

Appendix E

Device Type Code Assignments

XMI and VAXBI device type code assignments are shown in the output of the console command SHOW CONFIGURATION (see Section 5.18). These device types and numerical code values, and their module information, are given in Table E–1 for the XMI and in Table E–2 for the VAXBI card cage.

Table E–1: XMI Device Type Code Assignments

Code	Adapter	Standard or Optional	Function
2001	DWMBA/A	S	XMI-to-VAXBI adapter
4001	MS62A	S	Memory
8001	KA62B	S	Processor (timeshare systems)
8001	KA62B-S	S	Processor (server systems)

Table E–2: VAXBI Device Type Code Assignments

Code	Adapter	Std Opt'l	No. Slots	Function
0101	DRB32	O	1	Parallel port.
0103	KLESI–B	O	1	TU81 controller; local (nonclustered) tape subsystem. Also, RBV20/RBV64 controller; WORM disk drive.
0108	CIBCA	O[1]	2	VAXcluster port interface (VAXBI-to-CI adapter); connects a system to a VAXcluster.

[1]One disk or VAXcluster adapter is standard in timeshare systems.

Table E–2 (Cont.): VAXBI Device Type Code Assignments

Code	Adapter	Std Opt'l	No. Slots	Function
0109[2]	DHB32	O	1	Communication device; supports up to 16 terminals.
0109[2]	DMB32	O	1	Interface for 8-channel asynchronous communications; connects terminals.
010A	DSB32	O	1	Synchronous communications.
010E	KDB50	O[1]	2	KDB50 DSA disk adapter; enables connection to disk drives.
2107	DWMBA/B	S	1	VAXBI part of the XMI interface.
410B	TBK70	S	1	TK tape drive controller; connects the TK to the system.
410F	DEBNA	S	1	Ethernet port interface; connects a system to the Ethernet.

[1]One disk or VAXcluster adapter is standard in timeshare systems.

[2]0109 is the device type code for both DMB32 and DHB32 adapters; the SHOW CONFIGURATION command reports the DHB32 as a DMB32.

Glossary

Adapter
A node that interfaces other buses, communication lines, or peripheral devices to the VAXBI bus or the XMI bus.

Address space
The 1 Gbyte of physical address space supported by the VAXBI bus or the XMI bus.

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). The VAX 6300 can be booted from one of four boot devices: the system TK tape drive, a local system disk connected through a KDB50, a disk connected to the system through a CI adapter (CIBCA), 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 processor with the lowest node number that passes self-test and which has not been excluded from the system configuration with a SET CPU console command; equivalent to "primary processor."

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 VAXcluster port interface; connects a system to a VAXcluster.

Cold start

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

Console communications area (CCA)

Segment of system main memory reserved by the console program.

Console mode

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

DEBNA

VAXBI 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 with one synchronous channel for a line printer.

DRB32

VAXBI adapter; parallel port.

DSB32

VAXBI adapter communication device; provides two synchronous lines.

DWMBA

The XMI-to-VAXBI adapter, a 2-module adapter; allows data transfer from VAXBI to the XMI, with total effective throughput of 10 Mbytes/s; DWMBA/A is the module in the XMI card cage, and DWMBA/B is the VAXBI module. Every VAXBI on the VAX 6300 system must have a DWMBA adapter.

Interleaving memory

See Memory interleaving.

KDB50

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

Memory interleaving

Method to optimize memory access time; VAX 6300 console program automatically interleaves the memories in the system for fastest memory access time, 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 an even number of like-sized memories to operate in parallel.

Memory node

Also called the MS62A. Memory is a global resource equally accessible by any processors on the XMI. Each memory module has 32 Mbytes of memory, with MOS dynamic RAMs, ECC logic, and control logic.

Module

A single VAXBI or XMI 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").

MS62A

XMI memory array; a memory subsystem of the XMI; memory is a global resource equally accessible by any processors on the XMI. Each memory module has 32 Mbytes of memory, with 1–Mbit MOS dynamic RAMs, ECC logic, and control logic; see also *Memory node.*

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

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

Processor node

Also called a KA62B; a single-board 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; write-once-read-many (WORM) optical disk drive.

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.

TBK70

VAXBI adapter connecting the TK tape drive to the system.

TU81E

VAXBI adapter; TU81 controller; local (nonclustered) tape subsystem.

VAXBI bus

The 200-ns bus used by the system 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.

VAX Diagnostic Supervisor (VDS)

Software that loads and runs diagnostic and utility programs.

VMB

The virtual memory boot program (VMB.EXE) that boots the operating system. VMB is the primary bootstrap program and is stored on the system disk. The goal of booting is to read VMB from the boot device.

XMI

The VAX 6300 internal high-speed system bus; it is a 64-bit bus, whereas the VAXBI bus, which is used for I/O, is a 32-bit 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

D

E

H

I

K

M

O

P

R

S

T

U

V

X