DEC 4000 AXP

Service Guide

Order Number: EK–KN430–SV. B01

Digital Equipment Corporation
Maynard, Massachusetts

Revised, July 1993
First Printing, December 1992

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

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.

S2384

This document was prepared using VAX DOCUMENT, Version 2.1.

Contents

3 Running System Diagnostics

4 Error Log Analysis

5 Repairing the System

6 System Configuration and Setup

Examples

Figures

Tables

Preface

This guide describes the procedures and tests used to service DEC 4000 AXP systems.

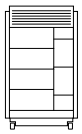
Intended Audience

This guide is intended for use by Digital Equipment Corporation service personnel and qualified self-maintenance customers.

Conventions

The following coventions are used in this guide.

Convention	Meaning
Return	A key name enclosed in a box indicates that you press that key.
Ctrl/*x*	Ctrl/*x* indicates that you hold down the Ctrl key while you press another key, indicated here by *x*. In examples, this key combination is enclosed in a box, for example, Ctrl/C.
bold type	In the online book (Bookreader), bold type in examples indicates commands and other instructions that you enter at the keyboard.
lowercase	Lowercase letters in commands indicate that commands can be entered in uppercase or lowercase.

	In some illustrations, small drawings of the DEC 4000 AXP system appear in the left margin. Shaded areas help you locate components on the front or back of the system.
Warning	Warnings contain information to prevent personal injury.
Caution	Cautions provide information to prevent damage to equipment or software.
[ ]	In command format descriptions, brackets indicate optional elements.
console command abbreviations	Console command abbreviations must be entered exactly as shown.
`boot`	Console and operating system commands are shown in this special typeface.
italic type	Italic type in console command sections indicates a variable.
< >	In console mode online help, angle brackets enclose a placeholder for which you must specify a value.
{ }	In command descriptions, braces containing items separated by commas imply mutually exclusive items.

1

System Maintenance Strategy

Any successful maintenance strategy is based on the proper understanding and use of information services, service tools, service support and escalation procedures, field feedback, and troubleshooting procedures. This chapter describes the maintenance strategy for the DEC 4000 AXP system.

- Section 1.1 provides a diagnostic strategy you should use to troubleshoot a DEC 4000 AXP system.
- Section 1.2 explains the service delivery methodology.
- Section 1.3 lists the product tools and utilities.
- Section 1.4 lists available information services.
- Section 1.5 describes field feedback procedures.

1.1 Troubleshooting the System

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

- Has the system been used before and did it work correctly?
- Have changes to hardware or updates to firmware or software been made to the system recently?
- What is the state of the system—is the operating system up?

  If the operating system is down and you are not able to bring it up, use the console environment diagnostic tools, such as RBDs and LEDs.

  If the operating system is up, use the operating system environment diagnostic tools, such as error logs, crash dumps, DEC VET and UETP exercisers, and other log files.

System problems can be classified into the following five categories:

1. Power problems
2. Problems getting to the console
3. Failures reported by the console subsystem
4. Boot failures
5. Failures reported by the operating system

Using these categories, you can quickly determine a starting point for diagnosis and eliminate the unlikely sources of the problem. Table 1–1 provides the recommended tools or resources you should use to isolate problems in each category.

Table 1–1 Recommended Troubleshooting Procedures

Description	Diagnostic Tools/Resources	Reference
1. Power Problems (Table 1–2)		
No power at system enclosure or trouble with power supply subsystem, as indicated by LEDs.	Power supply subsystem LEDs	Refer to Section 2.1.1 for information on interpreting power supply LEDs.
2. Problems Getting to Console Mode (Table 1–3)		
System powers up, but does not display power-up screen.	OCP LEDs	Refer to Section 2.1.2 for information on interpreting OCP LEDs.
	Console terminal troubleshooting flow	Refer to Table 1–3 for information on troubleshooting console terminal problems.
	Power-up sequence description	Refer to Section 2.3 and 2.3.3 for a description of the power-up and self-test sequence.
	Robust mode power-up	Refer to Section 2.2.3 for a description of robust mode power-up and its functions.

(continued on next page)

Table 1–1 (Cont.) Recommended Troubleshooting Procedures

Description	Diagnostic Tools/Resources	Reference
3. Failures Reported by the Console Program (Table 1–4)		
Power-up console screens indicate a failure.	Power-up screens	Refer to Section 2.2 for information on interpreting power-up self-tests.
	Console event log	Refer to Section 2.2 for information on the console event log.
	RBD device tests	Refer to Section 3.1 for information on running RBD device tests.
4. Boot Failures (Table 1–5)		
System fails to boot operating system.	Console commands (to examine environment variables and device parameters)	Refer to Chapter 6 for instructions on setting and examining environment variables and device parameters.
	Storage device troubleshooting flowcharts	Refer to Section 2.2.2.
	RBD device tests	Refer to Section 3.1 for information on running RBD device tests.
	Boot sequence description	Refer to Section 2.4 for a description of the boot sequence.

(continued on next page)

Table 1–1 (Cont.) Recommended Troubleshooting Procedures

Description	Diagnostic Tools/Resources	Reference
5. Failures Reported by the Operating System (Table 1–6)		
Operating system generates error logs; process hangs or operating system crashes.	Error logs	Refer to Chapter 4 for information on interpreting error logs.
	Crash dump	Refer to *OpenVMS AXP Alpha System Dump Analyzer Utility Manual* for information on how to interpret OpenVMS crash dump files.
		Refer to the *Guide to Kernel Debugging* (AA–PS2TA–TE) for information on using the DEC OSF/1 Krash Utility.
	DEC VET or UETP	Refer to Section 3.3 for a description of DEC VET, and Section 3.4 for information on running UETP software exercisers.
	Other log files	Refer to Chapter 4 for information on using log files such as SETHOST.LOG and OPERATOR.LOG to aid in troubleshooting.

Use the following tables to identify the diagnostic flow for the five types of system problems:

- Table 1–2 provides the diagnostic flow for power problems.
- Table 1–3 provides the diagnostic flow for problems getting to console mode.
- Table 1–4 provides the diagnostic flow for problems reported by the console program.
- Table 1–5 provides the diagnostic flow for boot problems.
- Table 1–6 provides the diagnostic flow for errors reported by the operating system.

Table 1–2 Diagnostic Flow for Power Problems

Symptom	Action	Reference
No AC power at system as indicated by AC present LED.	Check the power source and power cord.	
AC power is present, but system does not power on.	Check the system AC circuit breaker setting.	
	Check the DC on/off switch setting.	
	Examine power supply subsystem LEDs to determine if a power supply unit or fan has failed, or if the system has shut down due to an overtemperature condition.	Section 2.1.1

Table 1–3 Diagnostic Flow for Problems Getting to Console Mode

Symptom	Action	Reference
Power-up screens (or console event log) are not displayed.	Check OCP LEDs for a failure during self-tests. If two OCP LEDs remain lit, either option could be at fault.	Section 2.1.2
	Check baud rate setting for console terminal and system. The system default baud rate setting is 9600.	Section 6.5
	Try connecting the console terminal to the auxiliary console port. Note: No console output is directed to the auxiliary console port untill the power-up self-tests have completed and you press the Enter key or Ctrl/x.	
	For certain situations, power up under robust mode to bypass the power-up script and get to a low-level console. From console mode, you can then edit the nvram file, set and examine environment variables, or initialize individual phases of drivers.	Section 2.2.3

Table 1–4 Diagnostic Flow for Problems Reported by the Console Program

Symptom	Action	Reference
Power-up screens are displayed, but tests do not complete.	Use power-up display and/or OCP LEDs to determine error.	Section 2.2 and Section 2.1.2
Console program reports error.	Examine the console event log to check for embedded error messages recorded during power-up.	Section 2.2.1
	If power-up screens indicate problems with mass storage devices, use the troubleshooting flow charts to determine the problems.	Section 2.2.2
	Run RBD tests to verify problem.	Section 3.1
	Use the `show error` command to examine error information contained in serial control bus EEPROMs.	Section 3.1.4

Table 1–5 Diagnostic Flow for Boot Problems

Symptom	Action	Reference
System cannot find boot device.	Check system configuration for correct device parameters (node ID, device name, and so on) and environment variables (bootdef_dev, boot_file, boot_osflags).	Section 6.2.1, Section 6.3, and Section 6.4
Device does not boot.	Run device test to check that boot device is operating.	Section 3.2

Table 1–6 Diagnostic Flow for Errors Reported by the Operating System

Symptom	Action	Reference
System is hung or has crashed.	Examine the crash dump file.	Operating system documentation
	Use the `show error` command to examine error information contained in serial control bus EEPROMs (console environment error log).	Section 3.1.4
Operating system is up.	Examine the operating system error log files to isolate the problem.	Chapter 4
	If the problem occurs intermittently, run DEC VET or UETP to stress the system.	Section 3.3 and Section 3.4
	Examine other log files, such as SETHOST.LOG, OPCOM.LOG, and OPERATOR.LOG.	

1.2 Service Delivery Methodology

Before beginning any maintenance operation, you should be familiar with the following:

- The site agreement
- Your local and area geography support and escalation procedures
- Your Digital Services product delivery plan

Service delivery methods are part of the service support and escalation procedure. When appropriate, remote services should be part of the initial system installation. Methods of service delivery include the following:

- Local support
- Remote call screening
- Remote diagnosis (using modem support)

Recommended System Installation

The recommended system installation includes:

1. Hardware installation and acceptance testing. Acceptance testing includes running ROM-based diagnostics.
2. Software installation and acceptance testing. For example, using OpenVMS Factory Installed Software (FIS), and then acceptance testing with DEC VET or UETP.
3. Installation of the remote service tools and equipment to allow a Digital Service Center to dial in to the system. Refer to your remote service delivery strategy.

If you do not follow your service delivery methodology, you risk incurring excessive service expenses for any product.

1.3 Product Service Tools and Utilities

This section lists the array of service tools and utilities available for acceptance testing, diagnosis, and serviceability and provides recommendations for their use.

Error Handling/Logging

OpenVMS and DEC OSF/1 operating systems provide recovery from errors, fault handling, and event logging. The OpenVMS Error Report Formatter (ERF) provides bit-to-text translation of the event logs for interpretation. DEC OSF/1 uses UERF to capture the same kinds of information.

RECOMMENDED USE: Analysis of error logs is the primary method of diagnosis and fault isolation. If the system is up, or the customer allows the service representative to bring the system up, look at this information first. Refer to Chapter 4 for information on using error logs to isolate faults.

ROM-Based Diagnostics (RBDs)

ROM-based diagnostics have significant advantages:

- There is no load time.
- The boot path is more reliable.
- Diagnosis is done in console mode.

RECOMMENDED USE: The ROM-based diagnostic facility is the primary means of console environment testing and diagnosis of the CPU, memory, Ethernet, Futurebus+, and SCSI and DSSI subsystems. Use ROM-based diagnostics in the acceptance test procedures when you install a system, add a memory module, or replace the following: CPU module, memory module, backplane, I/O module, Futurebus+ device, or storage device. Refer to Section 3.1 for information on running ROM-based diagnostics.

Loopback Tests

Internal and external loopback tests are used to isolate a failure by testing segments of a particular control or data path. The loopback tests are a subset of the ROM-based diagnostics.

RECOMMENDED USE: Use loopback tests to isolate problems with the auxiliary console port and Ethernet controllers. Refer to Section 3.1.12 for instructions on performing loopback tests.

Firmware Console Commands

Console commands are used to set and examine environment variables and device parameters. For example, the `show memory`, `show configuration`, and `show device` commands are used to examine the configuration; the `set` (bootdef_dev, auto_action, and boot_osflags) commands are used to set environment variables; and the `cdp` command is used to configure DSSI parameters.

RECOMMENDED USE: Use console commands to set and examine environment variables and device parameters. Refer to Section 6.2 for information on firmware commands and utilities.

Option LEDs During Power-Up

The power supply LEDs display pass/fail test results for the power supply subsystem; the operator control panel (OCP) LEDs display pass/fail self-test results for CPU, memory, I/O, and Futurebus+ modules. Storage devices and Futurebus+ modules have their own LEDs as well.

RECOMMENDED USE: Monitor LEDs during power-up to see if the devices pass their self-tests. Refer to Chapter 2 for information on LEDs and power-up tests.

Operating System Exercisers (DEC VET or UETP)

The Digital Verifier and Exerciser Tool (DEC VET) is supported by the OpenVMS and DEC OSF/1 operating systems. DEC VET performs exerciser-oriented maintenance testing of both hardware and operating system. UETP is included with OpenVMS and is designed to test whether the OpenVMS operating system is installed correctly.

RECOMMENDED USE: Use DEC VET or UETP as part of acceptance testing to ensure that the CPU, memory, disk, tape, file system, and network are interacting properly. Also use DEC VET or UETP to stress test the user's environment and configuration by simulating system operation under heavy loads to diagnose intermittent system failures.

Crash Dumps

For fatal errors, such as fatal bugchecks, OpenVMS and DEC OSF/1 operating systems will save the contents of memory to a crash dump file.

RECOMMENDED USE: The support representative should analyze crash dump files. To save a crash dump file for analysis, you need to know proper system settings. Refer to the *OpenVMS AXP Alpha System Dump Analyzer Utility Manual* or the *Guide to Kernel Debugging* (AA–PS2TA–TE) for instructions.

Other Log Files

Several types of log files, such as operator log, console event log, sethost log, and accounting file (accounting.dat) are useful in troubleshooting.

RECOMMENDED USE: Use the sethost log and other log files to capture/examine the console output and compare with event logs and crash dumps in order to see what the system was doing at the time of the error.

1.4 Information Services

As a Digital service representative, you may access several information resources, including advanced database applications, online training courses, and remote diagnostic tools. A brief description of some of these resources follows.

Technical Information Management Architecture (TIMA)

TIMA is an online database that delivers technical and reference information to service representatives. A key benefit of TIMA is the pooling of worldwide knowledge and expertise.

DEC 4000 AXP Model 600 Series Information Set

The DEC 4000 AXP Model 600 Series Information Set consists of service documentation that contains information on installing and using, servicing and upgrading, and understanding the system. The guide you are reading is part of the set. The hardcopy kit number is EK–KN430–DK. The set is also available on TIMA. Refer to your *DEC 4000 Model 600 Information Map* (EK–KN430–IN) for detailed information.

Training

Computer Based Training (CBT) and lecture lab courses are available from the Digital training center:

- DEC 4000 System Installation and Troubleshooting (CBT course, EY–I090E–CO)
- Alpha Architecture Concepts (CBT course, EY–K725E–MT—magnetic tape; EY–K725E–TK—TK50 tape)
- Futurebus+ Concepts (EY–F479E–CO)

Digital Services Product Delivery Plan (Hardware or Software)

The Product Delivery Plan documents Digital Services' delivery commitments. The plan is the communications vehicle used among the various groups responsible for ensuring consistency between Digital Services' delivery strategies and engineering product strategies.

Blitzes

Technical updates are "blitzed" to the field using online mail and TIMA.

Storage and Retrieval System (STARS)

STARS is a worldwide database for storing and retrieving technical information. The STARS databases, which contain more than 150,000 entries, are updated daily.

Using STARS, you can quickly retrieve the most up-to-date technical information via DSNlink or DSIN.

1.5 Field Feedback

Providing the proper feedback to the corporation is essential in closing the loop on any service call. Consider the following when completing a service call:

- Fill out repair tags accurately and with as much symptom information as possible so that repair centers can fix a problem.
- Provide accurate call closeout information for Labor Activity Reporting System (LARS) or Call-Handling and Management Planning (CHAMP).
- Keep an up-to-date site maintenance log, whether hardcopy or electronic, to provide a record of the performed maintenance.

2

Power-On Diagnostics and System LEDs

This chapter provides information on how to interpret system LEDs and the power-up console screens. In addition, a description of the power-up and bootstrap sequence is provided as a resource to aid in troubleshooting.

- Section 2.1 describes how to interpret system LEDs.
- Section 2.2 describes how to interpret the power-up screens.
- Section 2.3 describes the power-up sequence.
- Section 2.3.3 describes power-on self-tests.
- Section 2.4 describes the boot sequence.

2.1 Interpreting System LEDs

DEC 4000 AXP systems have several diagnostic LEDs that indicate whether modules and subsystems have passed self-tests. The power system controller constantly monitors the power supply subsystem and can indicate several types of failures. The system LEDs are used primarily to troubleshoot power problems and problems getting to the console program.

This section describes the function of each of the following types of system LEDs, and what action to take when a failure is indicated.

- Power supply LEDs
- Operator control panel (OCP) LEDs
- I/O panel LEDs
- Futurebus+ option LEDs
- Storage device LEDs

2.1.1 Power Supply LEDs

The power supply LEDs (Figure 2–1) are used to indicate the status of the components that make up the power supply subsystem. The following types of failures will cause the power system controller to shut down the system:

- Power system controller (PSC) failure
- Fan failure
- Overtemperature condition
- Power regulator failures (indicated by the DC3 or DC5 failure LEDs)
- Front end unit (FEU) failure

Note

The AC circuit breaker will also shut down the system. If a power surge occurs, the breaker will trip, causing the switch to return to the off position (0). If the circuit breaker trips, wait 30 seconds before setting the switch to the on position (1).

Refer to Table 2–1 for information on interpreting the LEDs and determining what actions to take when a failure is indicated.

Figure 2–2 shows the local disk converter (LDC) and fan locations as they correspond to the fault ID display.

Figure 2–1 Power Supply LEDs

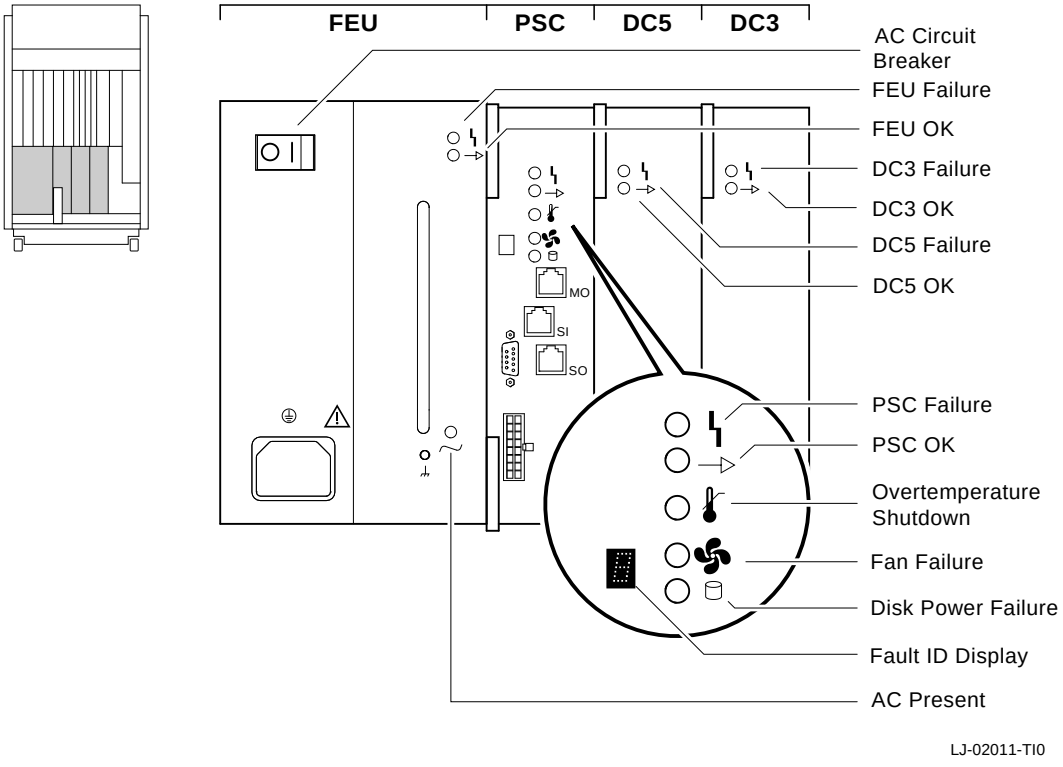


Table 2–1 Interpreting Power Supply LEDs

Indicator	Meaning	Action on Error
Front End Unit (FEU)		
AC Present	When lit, indicates AC power is present at the AC input connector (regardless of circuit breaker position).	If AC power is not present, check the power source and power cord. If the system will not power up and the AC LED is the only lit LED, check if the system AC circuit breaker has tripped. Replace the front end unit (Chapter 5) if the system circuit breaker is broken.
FEU OK	When lit, indicates DC output voltages for the FEU are above the specified minimum.	
FEU Failure	When lit, indicates DC output voltages for the FEU are less than the specified minimum.	Replace front end unit (Chapter 5).

(continued on next page)

Table 2–1 (Cont.) Interpreting Power Supply LEDs

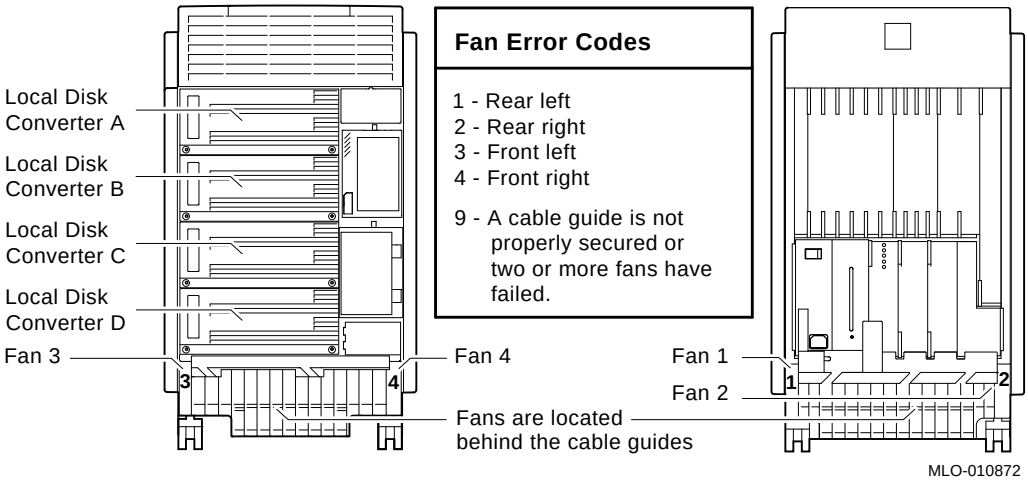
Indicator	Meaning	Action on Error
Power System Controller (PSC)		
PSC OK	When blinking, indicates the PSC is performing power-up self-tests. When steady, indicates the PSC is functioning normally.	
PSC Failure	When lit, indicates the PSC has detected a fault in itself.	Replace power system controller (Chapter 5).
Disk Power Failure	When lit, indicates a disk power problem for the storage compartment specified in the hexadecimal fault ID display. The most likely failing unit is the local disk converter, but a shorting cable or drive could also be at fault.	To isolate the local disk converter, disconnect the drives on the specified bus and then power up the system. If the Disk Power Failure LED lights with the drives disconnected, replace the failing local disk converter (Chapter 5). Refer to Figure 2–2 to locate the local disk converter specified by the fault ID display. A is the top compartment, D is the bottom compartment.
Fan Failure	When lit, indicates a fan has failed or a cable guide is not properly secured. The failure is identified by a number displayed in the hexadecimal fault ID display.	Refer to Figure 2–2 to locate the failure specified by the fault ID display. Replace the failing fan (Chapter 5).
Overtemperature Shutdown	When lit, indicates the PSC has shut down the system due to excessive internal temperature.	Set the AC circuit breaker to off (0) and wait one minute before turning on the system. Make sure the air intake is unobstructed and that the room temperature does not exceed maximum requirement as described in the *DEC 4000 Site Preparation Checklist*.

(continued on next page)

Table 2–1 (Cont.) Interpreting Power Supply LEDs

Indicator	Meaning	Action on Error
DC–DC Converter (DC3)		
DC3 OK	When lit, indicates that all the DC3 output voltages are within specified tolerances.	
DC3 Failure	When lit, indicates that one of the output voltages is outside specified tolerances.	Replace the DC3 converter (Chapter 5).
DC–DC Converter (DC5)		
DC5 OK	When lit, indicates the DC5 output voltage is within specified tolerances.	
DC5 Failure	When lit, indicates the DC5 output voltage is outside specified tolerances.	Replace the DC5 converter (Chapter 5).

Figure 2–2 LDC and Fan Unit Locations and Error Codes



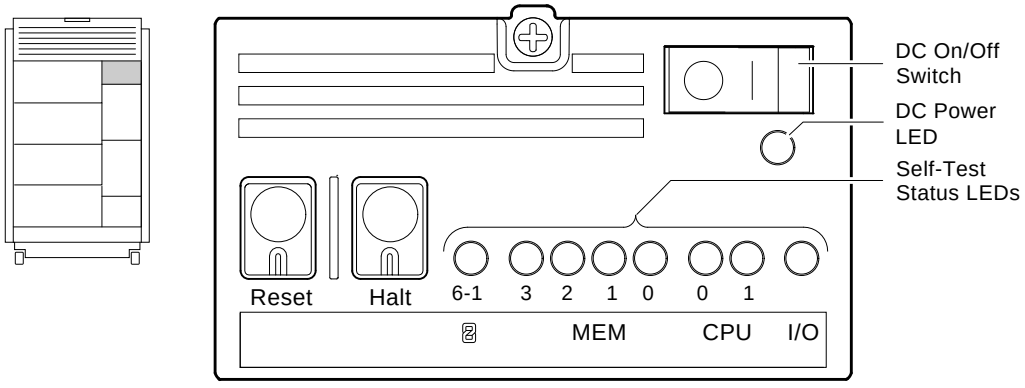
2.1.2 Operator Control Panel LEDs

The OCP LEDs (Figure 2–3) are used to indicate the progress and result of self-tests for Futurebus+, memory, CPU, and I/O modules. These LEDs are the primary diagnostic tool for troubleshooting problems getting to the console program.

Note

A failure in the CPU, memory module, or I/O module can cause both the I/O and CPU LEDs or I/O and memory LEDs to indicate self-test failures even if only one of the modules is failing. If two LEDs are lit, the I/O module is the more likely source of the failure.

Figure 2–3 OCP LEDs



LJ-02008-TI0

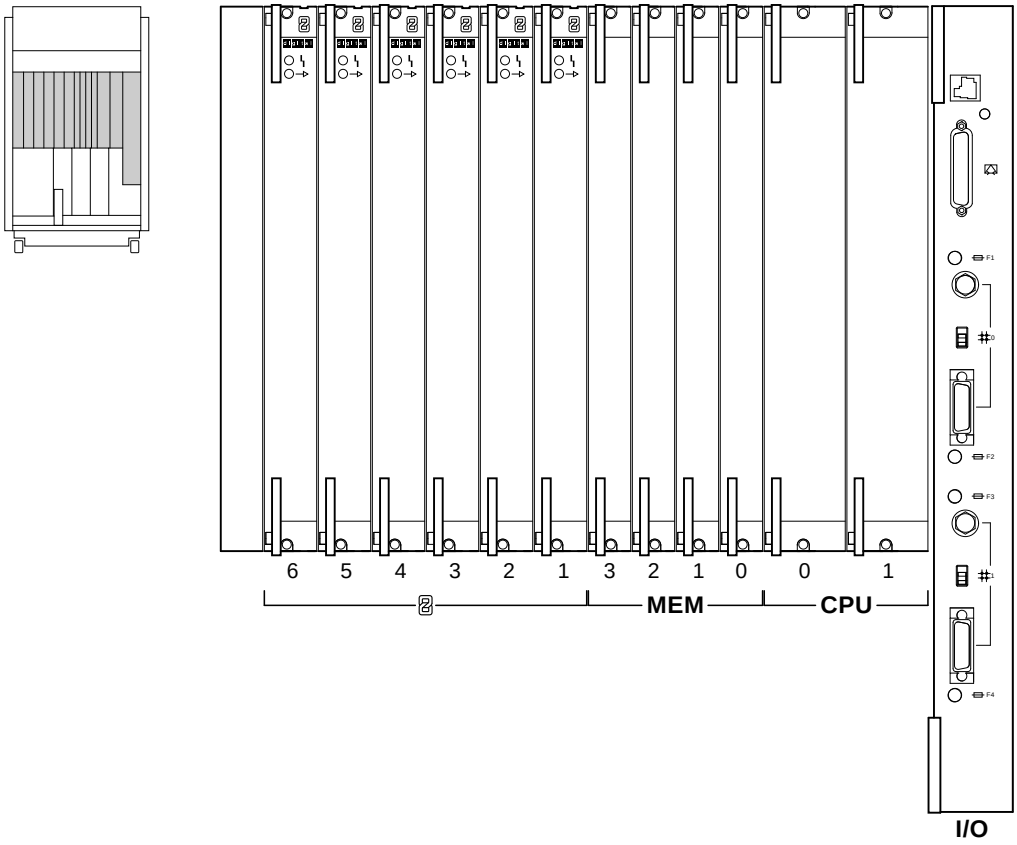
Refer to Table 2–2 for information on interpreting the OCP LEDs and determining what actions to take when a failure is indicated.

Figure 2–4 shows the module locations as they correspond to the LEDs.

Table 2–2 Interpreting OCP LEDs

Indicator	Meaning	Action on Error
Futurebus+ 6–1	Remains lit if a Futurebus+ option has failed power-on diagnostics.	Examine LEDs on the Futurebus+ options to determine which option to replace.
MEM 3, 2, 1, 0	Remains lit if a memory module has failed power-on diagnostics. If no good memory is found, all four memory LEDs may remain lit even if there are less than four memory modules present.	Replace the failed module (Chapter 5).
CPU 0, 1	Remains lit if a CPU module has failed power-on diagnostics.	Replace the failed module (Chapter 5).
I/O	Remains lit if the I/O module has failed power-on diagnostics.	Replace the I/O module (Chapter 5).
DC Power	When lit indicates the proper DC power is present. When unlit, indicates no DC power is present.	If no DC power is indicated, set the DC on/off switch to on (1) and examine the power supply LEDs.

Figure 2–4 Module Locations Corresponding to OCP LEDs



2.1.3 I/O Panel LEDs

The I/O panel LEDs (Figure 2–5) are used to indicate the status of ThinWire and thickwire (standard) Ethernet fuses.

Refer to Table 2–3 for information on interpreting the LEDs and determining what actions to take when a failure is indicated.

Figure 2–5 I/O Panel LEDs

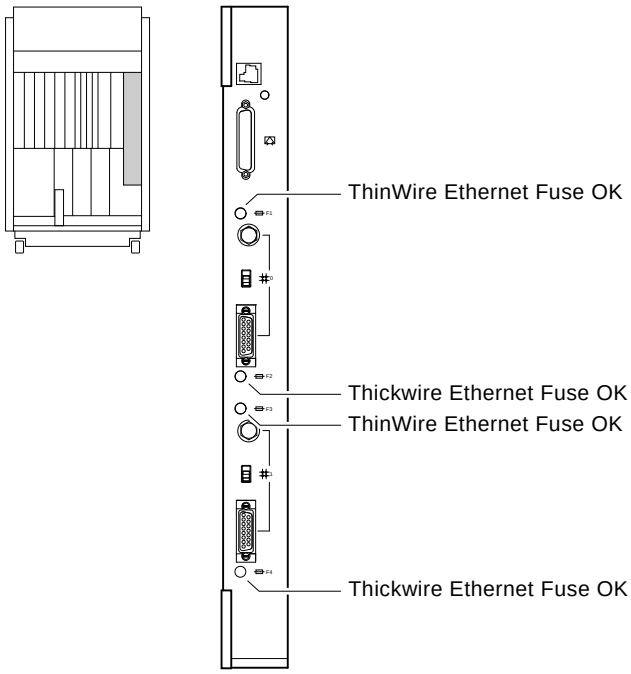


Table 2–3 Interpreting I/O Panel LEDs

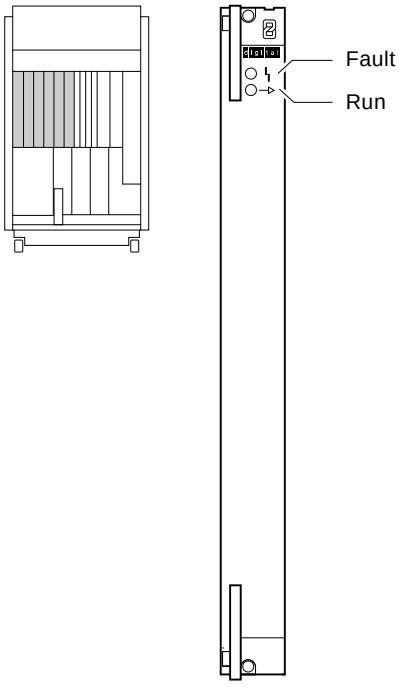
Indicator	Meaning	Action on Error
ThinWire Ethernet Fuse OK	When lit, indicates ThinWire fuse is good; unlit indicates fuse has blown.	Replace fuse (refer to Chapter 5).
Thickwire Ethernet Fuse OK	When lit, indicates thickwire fuse is good; unlit indicates fuse has blown.	Replace fuse (refer to Chapter 5).

2.1.4 Futurebus+ Option LEDs

The Futurebus+ option LEDs (Figure 2–6) are used to indicate the progress and result of self-tests for a specific Futurebus+ option.

Refer to Table 2–4 for information on interpreting the LEDs and determining what actions to take when a failure is indicated.

Figure 2–6 Futurebus+ Option LEDs



LJ-02010-TI0

Table 2–4 Interpreting Futurebus+ Option LEDs

Indicator	Meaning	Action on Error
Fault	The Fault indicator lights during self-tests. If it remains lit, the module has failed self tests.	Replace module.
Run	The Run indicator blinks during self-tests and remains lit if the module passes self-tests.	

2.1.5 Storage Device LEDs

Storage device LEDs are used to indicate the status of the device. The LEDs for fixed-media storage devices are shown in Figures 2–7 and Figure 2–8. Refer to the *DEC 4000 Model 600 Series Owner's Guide* for information on LEDs for the removable-media devices.

Refer to Table 2–5 for information on interpreting the LEDs and determining what actions to take when a failure is indicated.

Figure 2–7 Fixed-Media Mass Storage LEDs (SCSI)

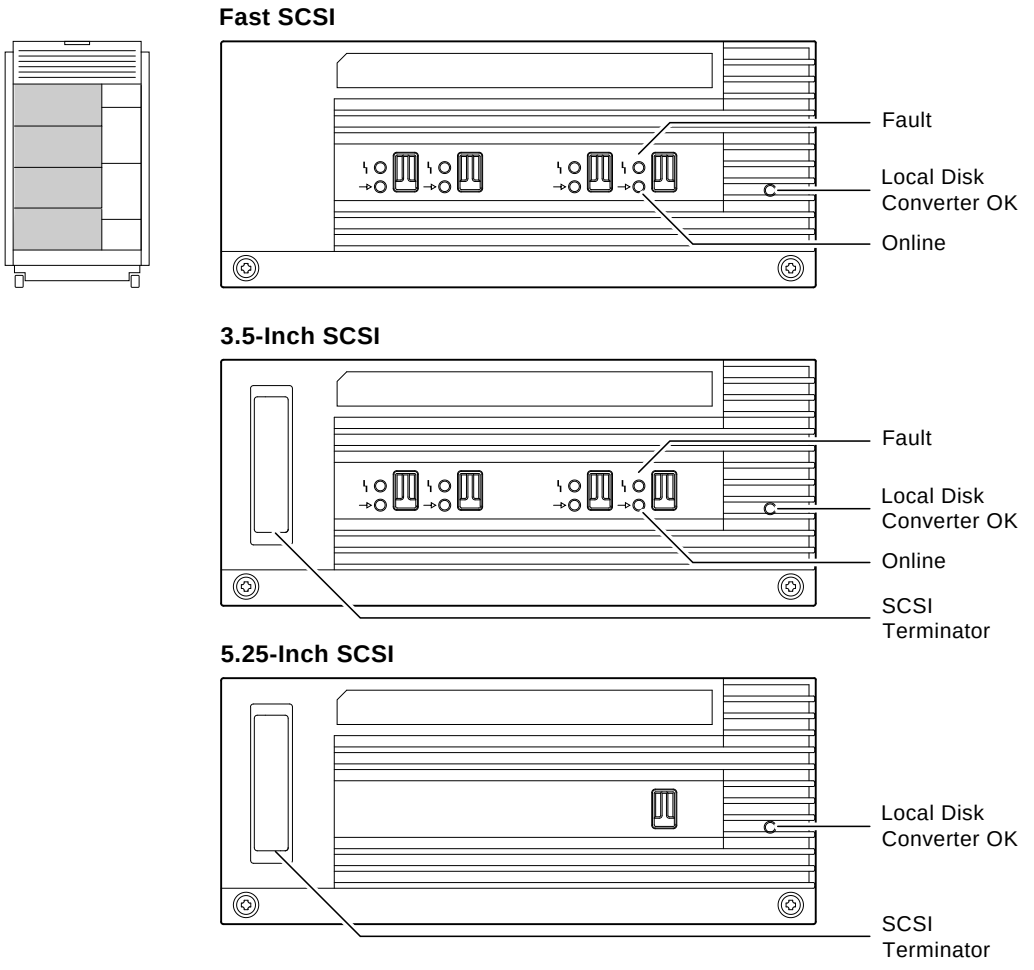


Figure 2–8 Fixed-Media Mass Storage LEDs (DSSI)

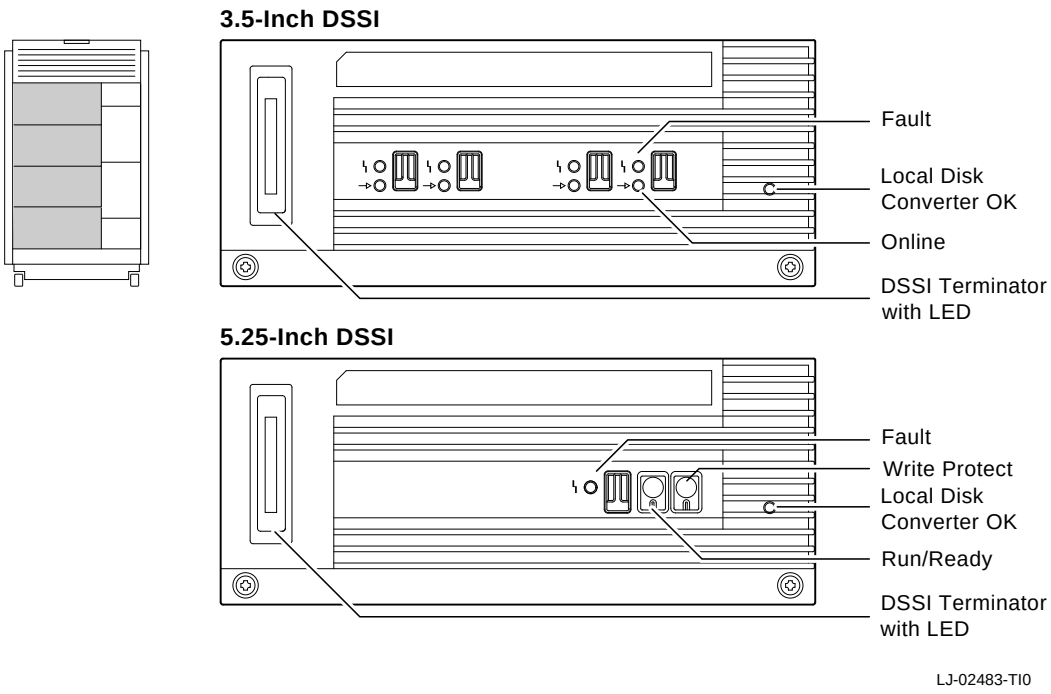


Table 2–5 Interpreting Fixed-Media Mass Storage LEDs

Indicator	Meaning	Action on Error
Fault	When lit, indicates an error condition in the device. The Fault indicator may light temporarily during self-tests.	Run device RBD tests and internal device tests to determine the nature of the error, and replace device.
Online	DSSI: When lit, indicates the device is on line and available for use. Under normal operation, flashes as seek operations are performed. SCSI: Flashes as seek operations are performed; indicates drive activity.	

(continued on next page)

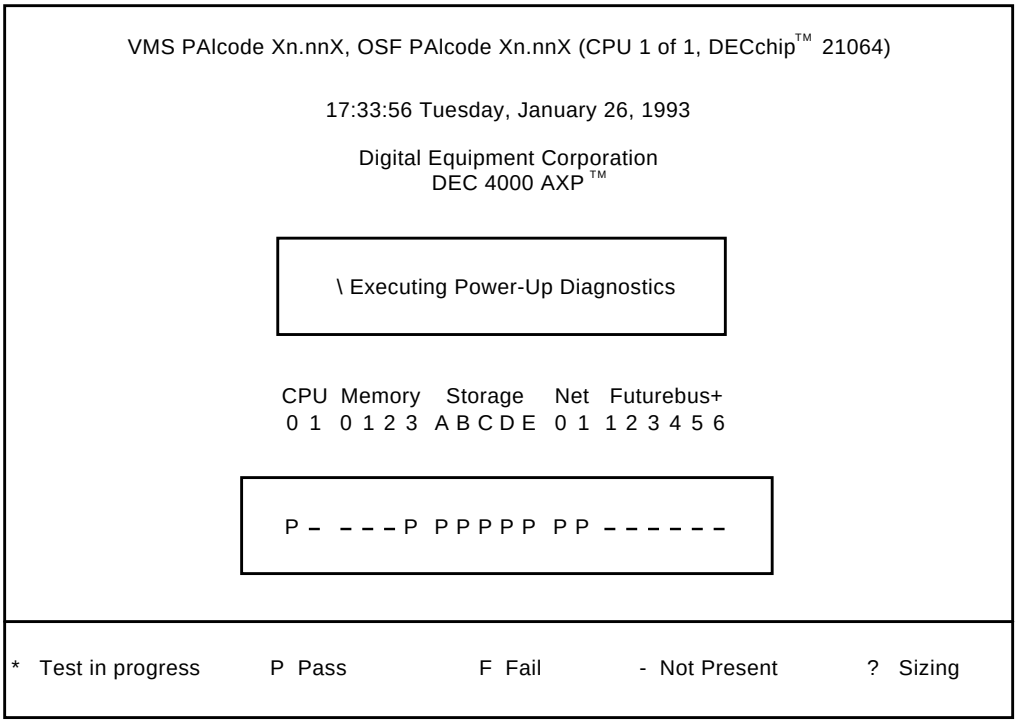
Table 2–5 (Cont.) Interpreting Fixed-Media Mass Storage LEDs

Indicator	Meaning	Action on Error
DSSI Terminator	When lit, indicates DSSI termination power is present.	If the DSSI terminator LED does not light, check the DSSI bus connections for that bus. If bus connections seem secure, the local disk converter module or DC5 converter may need to be replaced (Section 5.2): • Local disk converters (located in the fixed-media storage compartments) supply termination power for fixed-media storage devices. • The DC5 converter (part of the power supply subsystem) supplies termination power for storageless fixed-media compartments.
Local Disk Converter OK	When lit, indicates local disk converter for the specified storage compartment has power (this LED is located on the local disk power supply module behind the front panel of the storage compartment).	Confirm that the system power supply is working properly (by checking power supply LEDs). Replace the local disk converter module (Section 5.2).

2.2 Power-Up Screens

During power-up self-tests a screen similar to the one shown in Figure 2–9 is displayed on the console terminal. The screen shows the status and result of the self-tests.

Figure 2–9 Power-Up Self-Test Screen



LJ-02266-TI0

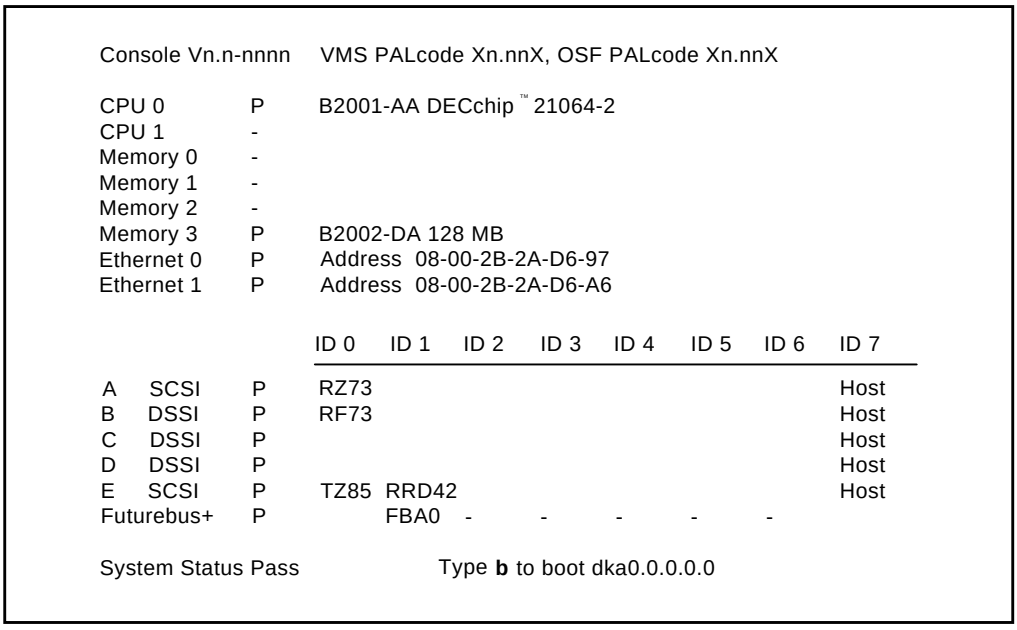
Note

A power-on self-test failure indicated under Storage A–E may represent a failure of an embedded storage adapter (A–E) or failure of a drive on the specified bus. Check the console event log for additional information (Section 2.2.1).

Power-on self-tests failures indicated for all six Futurebus+ slots indicate a failure of the Futurebus+ bridge on the I/O module. Replace the I/O module in the event that all six Futurebus+ slots show failures.

When the power-up diagnostics are completed, a second screen similar to the one shown in Figure 2–10 is displayed. This screen provides configuration information for the system.

Figure 2–10 Sample Power-Up Configuration Screen



>>>

LJ-02267-TI0

2.2.1 Console Event Log

DEC 4000 AXP systems maintain a console event log consisting of status messages received during power-on self-tests. If there are problems during power-up, standard error messages may be embedded in the console event log. To display a console event log, use the `cat el` command.

Use the `set screen_mode off` command if you want to display the console event log during power-up, rather than the two power-up screens.

The following example shows an abbreviated console event log that contains two standard error messages: The first (a hard error) indicates a failure with storage bus B. This failure could be caused by a bad LDC, improperly seated storage drawer, or a disconnected power cable within the storage drawer. The second (a soft error) indicates a SCSI continuity card is missing from the removable-media storage compartment.

```
>>> cat el
Starting console.
halt code = 1
PC = 0
initialized idle PCB
initializing semaphores
   .
   .
   .
test Storage Bus B
ncr1, loopback connector attached  OR
SCSI bus failure, could not acquire bus; Control Lines:ff Data lines:ff
ncr1 SCSI bus failure

*** Hard Error - Error #800 -

Diagnostic Name        ID             Device  Pass  Test  Hard/Soft   7-OCT-1970
powerup          00000004               ncr1     0     0     1    0     10:48:58
Storage Bus B failure

*** End of Error ***

enable ncr2 ACK
test Storage Bus C
port p_c0.7.0.2.0 initialized, scripts are at 1d07e0
SCSI device found on pkc.0.0.2.0
loading SCSI driver for port p_c0.7.0.2.0
   .
   .
   .
*** Soft Error - Error #1 - Lower SCSI Continuity Card Missing (connector J7)

Diagnostic Name        ID             Device  Pass  Test  Hard/Soft   7-OCT-1992
io_test          00000067       scsi_low_con     1     1     0    1     11:25:53

*** End of Error ***
device mud9.5.0.3.0 (TF85) found on pud0.5.0.3.0
>>>
```

2.2.2 Mass Storage Problems Indicated at Power-Up

Mass storage failures at power-up are usually indicated in one of two ways:

- The power-up screens report a storage adapter port failure (indicated by an "F").
- One or more drives are missing from the configuration screen display (or too many drives are displayed).

Figures 2–11 and 2–12 provide a flowchart for troubleshooting fixed-media mass storage problems indicated at power-up. Use the flowchart to diagnose the likely cause of the problem. Table 2–6 lists the symptoms and corrective action for each of the possible problems.

Figure 2–11 Flowchart for Troubleshooting Fixed-Media Problems

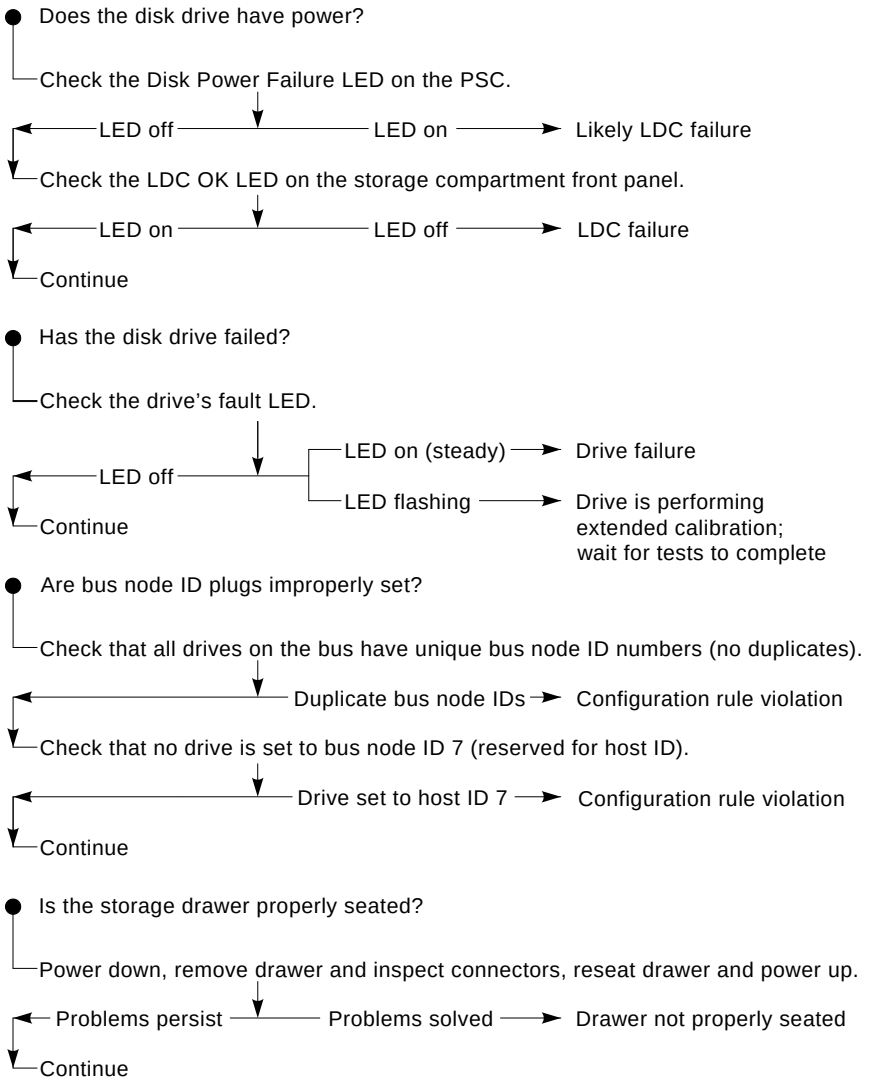


Figure 2–12 Flowchart for Troubleshooting Fixed-Media Problems (Continued)

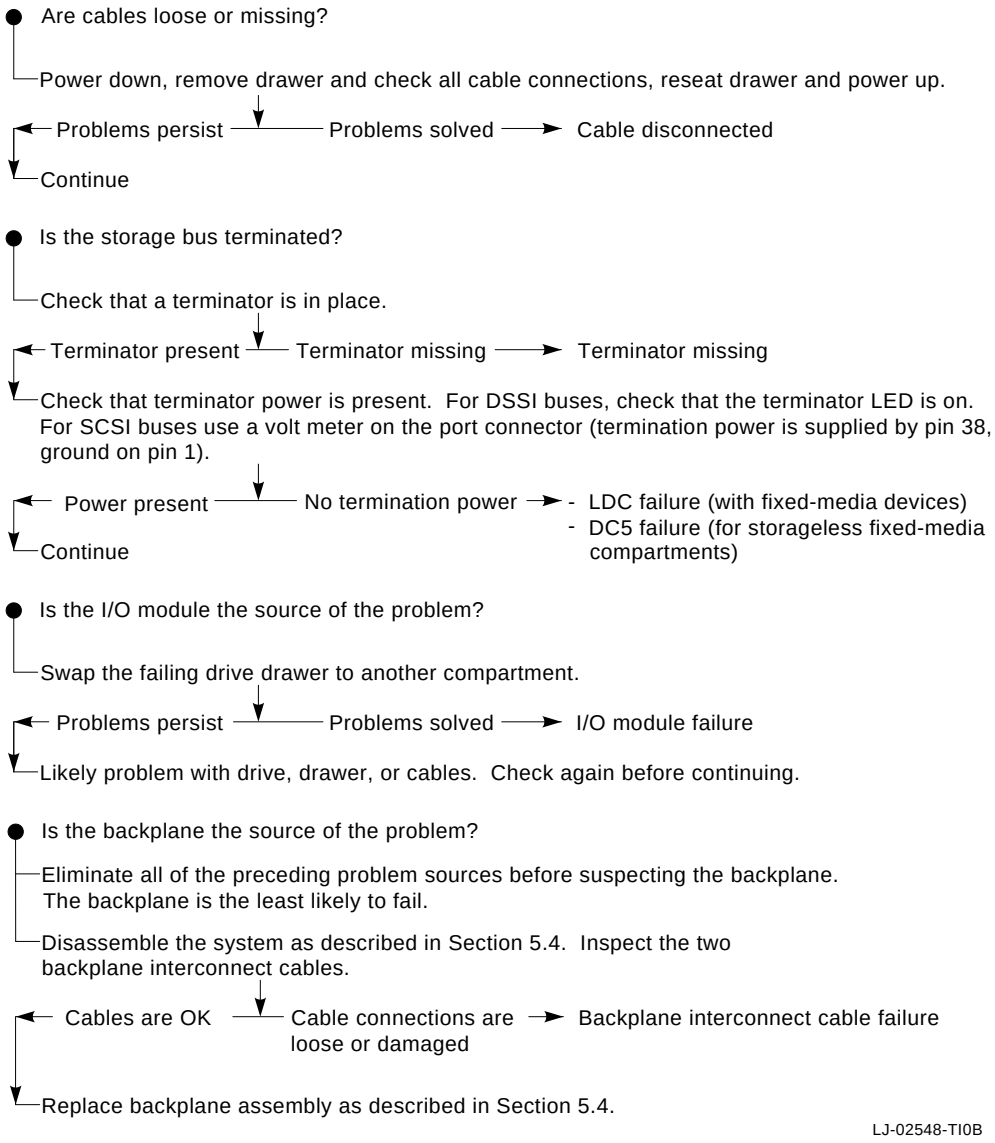


Table 2–6 Fixed-Media Mass Storage Problems

Problem	Symptom	Corrective Action
LDC failure	Disk power failure LED on PSC is on. LDC OK LED on storage compartment front panel is off. Power-up screen reports a failing storage adapter port.	Replace LDC.
Drive failure	Fault LED for drive is on (steady).	Replace drive.
Duplicate bus node ID plugs (or a missing plug)	Drives with duplicate bus node ID plugs are missing from the configuration screen display. A drive with no bus node ID plug defaults to zero.	Correct bus node ID plugs.
Bus node ID set to 7 (reserved for host ID)	Valid drives are missing from the configuration screen display. One drive may appear seven times on the configuration screen display.	Correct bus node ID plugs.
Storage drawer not properly seated	Disk power failure LED on PSC is on. LDC OK LED on storage compartment front panel is off. Power-up screen reports a failing storage adapter port.	Remove drawer and check its connectors. Reseat drawer.

(continued on next page)

Table 2–6 (Cont.) Fixed-Media Mass Storage Problems

Problem	Symptom	Corrective Action
Missing or loose cables	Cable: storage device to ID panel—Bus node ID defaults to zero; online LEDs do not come on.	Remove storage drawer and inspect cable connections.
	Flex circuit: LDC to storage interface module—Disk power failure LED on PSC is on; LDC OK LED on storage compartment front panel is off; and power-up screen reports a failing storage adapter port.	
	Cable: LDC to storage interface module—Power-up screen reports a failing storage adapter port; drive LEDs do not come on at power-up.	
	Cable: LDC to storage device—Drive does not show up in configuration screen display.	
Terminator missing	Read/write errors in console event log; storage adapter port may fail	Attach terminator to connector port.
No termination power	DSSI terminator LED is off, or no termination voltage measured at SCSI connector (pin 38, ground pin 1); Read/write errors; storage adapter port may fail.	Replace LDC (termination power source for fixed-media storage compartments).
		Replace DC5 converter (termination power source for storageless fixed-media storage compartments).
I/O module failure	The storage drawer exhibits no problems when moved to another compartment.	Replace I/O module.
Backplane failure	Replacing the I/O module does not solve problem. The port continues to fail and the problem is not with the storage drawer.	Disassemble system and inspect backplane interconnect cables. If the cables and cable connections do not appear to be the problem, replace the backplane.

Figures 2–13 and 2–14 provide a flowchart for troubleshooting removable-media storage problems indicated at power-up. Use the flowchart to diagnose the likely cause of the problem. Table 2–7 lists the symptoms and corrective action for each of the possible problems.

Figure 2–13 Flowchart for Troubleshooting Removable-Media Problems

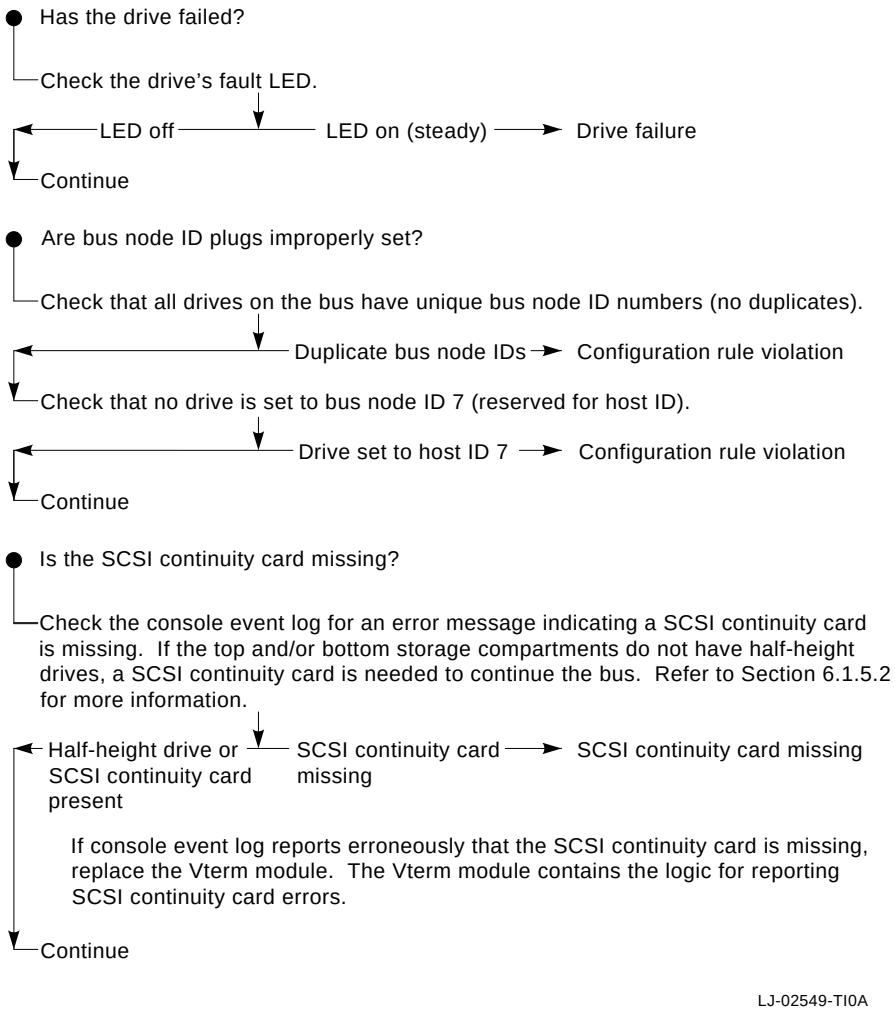


Figure 2–14 Flowchart for Troubleshooting Removable-Media Problems (Continued)

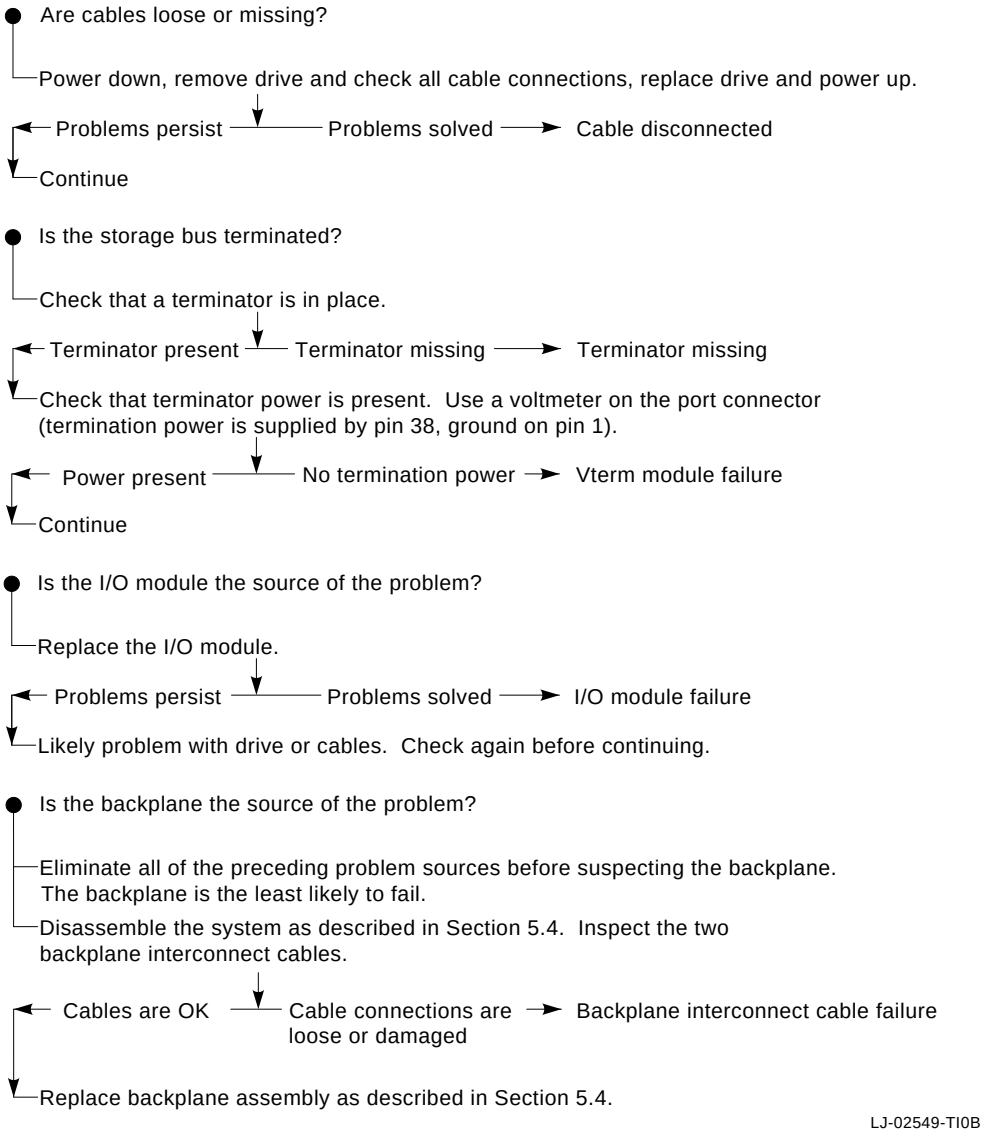


Table 2–7 Removable-Media Mass Storage Problems

Problem	Symptom	Corrective Action
Drive failure	Fault LED for drive is on (steady).	Replace drive.
Duplicate bus node ID plugs (or a missing plug)	Drives with duplicate bus node ID plugs are missing from the configuration screen display. A drive with no bus node ID plug defaults to zero.	Correct bus node ID plugs.
Bus node ID set to 7 (reserved for host ID)	Valid drives are missing from the configuration screen display. One drive may appear seven times on the configuration screen display.	Correct bus node ID plugs.
SCSI continuity card missing	Power-up screen reports a failing storage adapter port; console event log contains soft error message reporting a SCSI continuity card is missing; drives on Bus E are not displayed on configuration screen; possible read/write errors.	Attach SCSI continuity card (Section 6.1.5.2). If console erroneously reports SCSI continuity card as missing, replace the Vterm module. The Vterm module contains the logic for reporting SCSI continuity card errors.
Missing or loose cables	Cable: storage device to ID panel—Bus node ID defaults to zero; online LED does not come on. Cable: Power—Drive does not show up in configuration screen display.	Remove device and inspect cable connections.
Terminator missing	Read/write errors in console event log; storage adapter port may fail	Attach terminator to connector port.
Vterm module failure	No termination voltage measured at Bus E SCSI connector (pin 38, ground pin 1); Read/write errors; storage adapter port may fail; or console erroneously reports SCSI continuity card as missing.	Replace Vterm module (termination power source for removable-media storage compartment).

(continued on next page)

Table 2–7 (Cont.) Removable-Media Mass Storage Problems

Problem	Symptom	Corrective Action
I/O module failure	Problems persist after eliminating the above problem sources.	Replace I/O module.
Backplane failure	Replacing the I/O module does not solve problem—the port continues to fail and the problem is not with the device or cables.	Disassemble system and inspect backplane interconnect cables. If the cables and cable connections do not appear to be the problem, replace the backplane.

2.2.3 Robust Mode Power-Up

Robust mode allows you to power up without initiating drivers or running power-up diagnostics.

Robust mode permits you to get to the console program when one of the following is the cause of a problem getting to the console program under normal power-up:

- An error in the nonvolatile nvram file
- An incorrect environment variable setting
- A driver error

Note

The console program has limited functionality in robust mode.

Once in console mode, you can:

- Edit the nvram file (using the `edit` command)
- Assign a correct value to an environment variable (using the `show` and `set` commands)
- Start individual classes or sets of drivers, called phases (using the `init -driver` # command. The pound sign (#) is the phase number 2, 3, 4, or 5, and each phase is started individually in increasing order.

Note

The nonvolatile file, nvram, is shipped from the factory with no contents. The customer can use the `edit` command to create a customized script or command file that is executed as the last step of every power-up.

To set the system to robust mode, set the baud rate select switch located behind the OCP to 0, as shown in Section 6.5. The robust mode setting uses a 9600 console baud rate.

2.3 Power-Up Sequence

During the DEC 4000 AXP power-up sequence, the power supplies are stabilized and tested and the system is initialized and tested via the firmware power-on self-tests.

The power-up sequence includes the following:

- Power supply power-up:
  - Includes AC power-up and power supply self-test.
  - Includes DC power-up and power supply self-tests.
- Two sets of power-on diagnostics:
  - Serial ROM diagnostics
  - Console firmware-based diagnostics

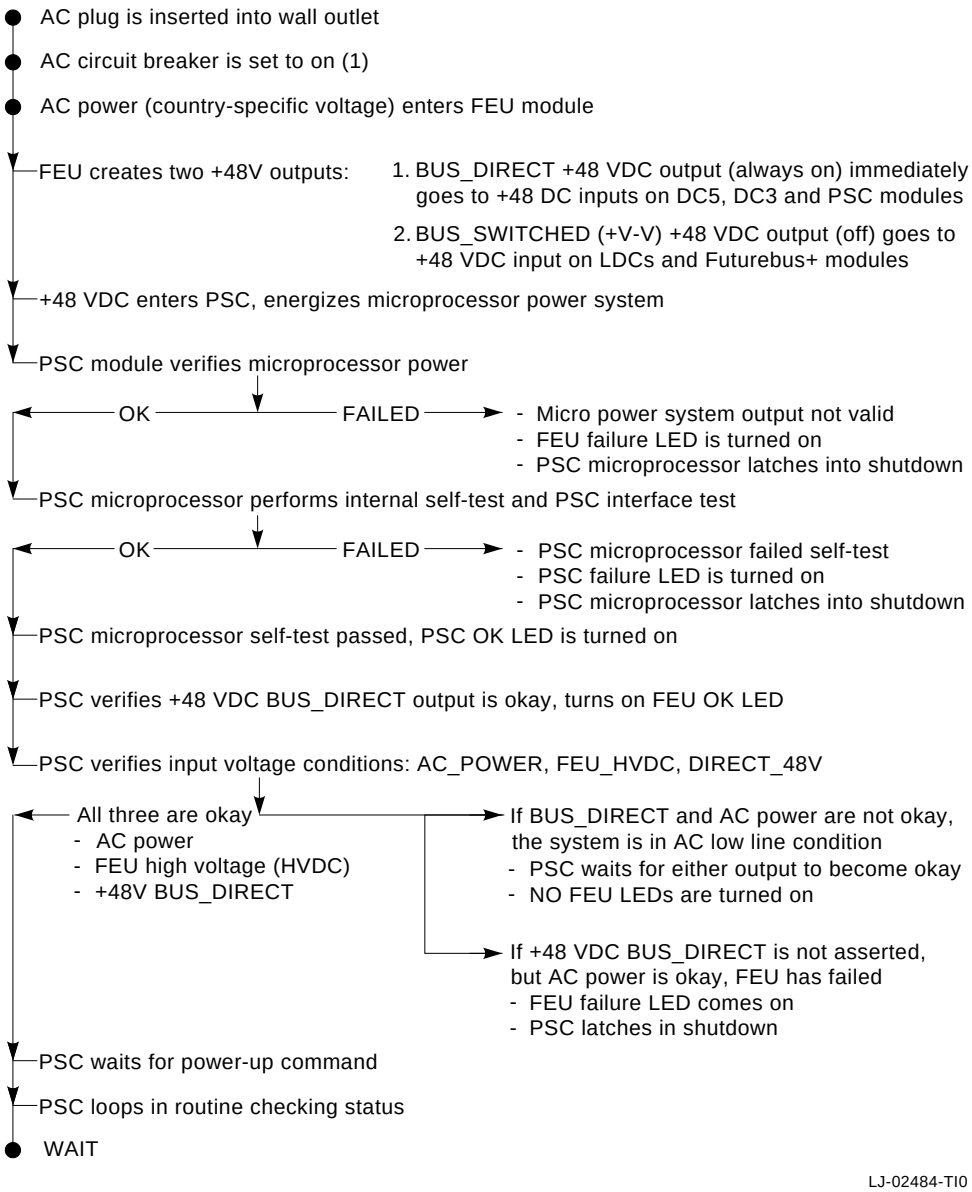
2.3.1 AC Power-Up Sequence

With no AC power applied, no energy is supplied to the entire enclosure. AC power is applied to the system with the AC circuit breaker on the front end unit (FEU) of the power supply (see Figure 2–1) . With just AC power applied, the AC present LED is the only LED illuminated on the power supply.

Figure 2–15 provides a description of the AC power-up sequence.

Failures during AC power-up are indicated by the power supply subsystem LEDs. Additional error information is displayed on the PSC Fault ID display. Refer to Appendix B for PSC fault display information.

Figure 2–15 AC Power-Up Sequence



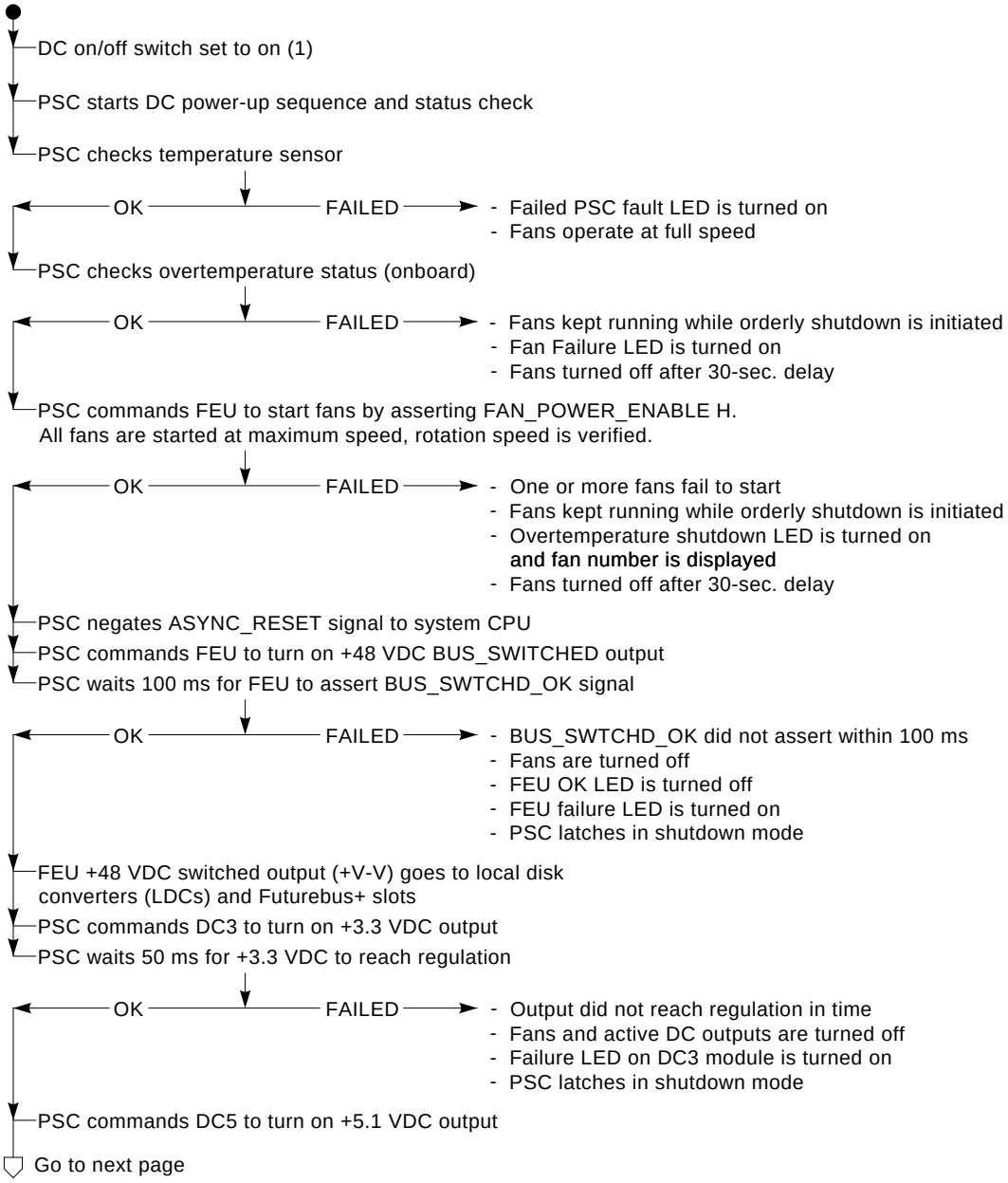
2.3.2 DC Power-Up Sequence

DC power is applied to the system with the DC on/off switch on the operator control panel.

Figures 2–16 and 2–17 provide a description of the DC power-up sequence.

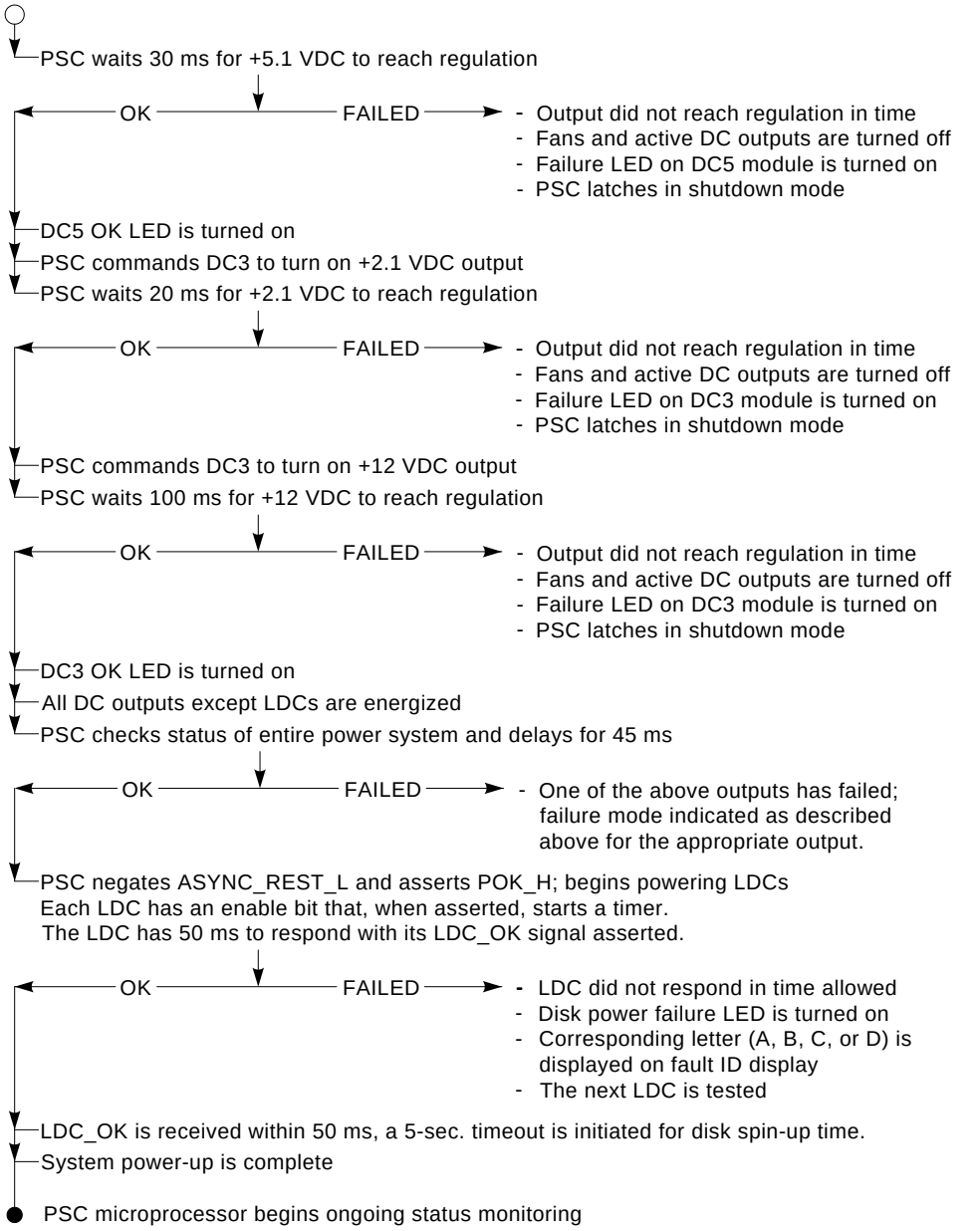
Failures during DC power-up are indicated by the power supply subsystem LEDs. Additional error information is displayed on the PSC Fault ID display. Refer to Appendix B for PSC fault display information.

Figure 2–16 DC Power-Up Sequence



LJ-02485-TI0A

Figure 2–17 DC Power-Up Sequence (Continued)



LJ-02485-TI0B

2.3.3 Firmware Power-Up Diagnostics

After successful completion of AC and DC power-up sequences, the processor performs its power-up diagnostics. These tests verify system operation, load the system console, and test the kernel system, including all boot path devices. These tests are performed as two distinct sets of diagnostics:

1. Serial ROM diagnostics—These tests are loaded from the serial ROM located on the CPU module into the CPU's instruction cache (I-cache). They check the basic functionality of the system and load the console code from the FEPROM on the I/O module into system memory.

   Failures during these tests are indicated by LEDs on the operator control panel.

2. Console firmware-based diagnostics—These tests are executed by the console code. They test the kernel system, including all boot path devices.

   Failures during these tests are reported to the console terminal (via the power-up screen or console event log).

2.3.3.1 Serial ROM Diagnostics

The serial ROM diagnostics are loaded into the CPU's I-cache from the serial ROM on the CPU module. They test the system in the following order:

1. Test the CPU and backup cache located on the CPU module.
2. Test the CPU module's system bus interface.
3. Check the access to the I/O module.
4. Locate the largest memory module in the system and test the first 4 MB of memory on the module. Only the first 4 MB of memory are tested. If there is more than one memory module of the same size, the one closest to the CPU is tested first.

   If the memory test fails, the next largest memory module in the system is tested. Testing continues until a good memory module is found. If a good memory module is not found, the corresponding LEDs on the OCP are illuminated and the power-up diagnostics are terminated.

5. After finding the first memory module with a good first 4 MB of memory, the console program is loaded into memory from the FEPROM on the I/O module. At this time control is passed to the console code and the console firmware-based diagnostics are run.

2.3.3.2 Console Firmware-Based Diagnostics

Console firmware-based tests are executed once control is passed to the console code in memory. They check the system in the following order:

1. Perform a complete check of system memory. If a system has more than one memory module, the modules are checked in parallel.
2. Set memory interleave to maximize interleave factor across as many memory modules as possible (one, two, or four-way interleaving). During this time the console firmware is moved into backup cache on the primary CPU module. After memory interleave is set, the console firmware is moved back into memory.

   Steps 3–7 may be completed in parallel.
3. Start the I/O drivers for mass storage devices and tapes. At this time a complete functional check of the machine is made. After the I/O drivers are started, the console program continuously polls the bus for devices (approximately every 20 or 30 seconds).
4. Size, configure, and test the Futurebus+ options.
5. Exercise memory.
6. Check that the SCSI continuity card or a storage device is installed in the removable-media storage bus (Bus E, connectors J6 and J7).
7. Run exercisers on the disk drives currently seen by the system.

   Note

   This step does not currently ensure that all disks in the system will be tested or that any device drivers will be completely tested. To ensure complete testing of disk devices, use the `test` command.

8. Enter console mode or boot the operating system. This action is determined by the auto_action environment variable.

2.4 Boot Sequence

Bootstrapping is the process of loading a program image into memory and transferring control to the loaded program. The system firmware uses the bootstrap procedure defined by the Alpha AXP architecture and described in the *Alpha System Reference Manual*. On a DEC 4000 AXP system, bootstrap can be attempted only by the primary processor or boot processor. The firmware uses

device and optional filename information specified either on the command line or in appropriate environment variables.

There are only three conditions under which the boot processor attempts to bootstrap the operating system:

1. The `boot` command is typed on the console terminal.
2. The system is reset or powered up and AUTO_ACTION is set to boot (and the halt switch is not set to halt).
3. An operating system restart is attempted and fails.

The firmware's function in a bootstrap is to load a program into memory and begin its execution. This program may be a primary bootstrap program, such as Alpha Primary Boot (APB), Ultrixboot, or any other applicable program specified by the user or residing in the boot block, MOP server, or TCP/IP server.

2.4.1 Cold Bootstrapping in a Uniprocessor Environment

This section describes a cold bootstrap in a uniprocessor environment. A system bootstrap will be a cold bootstrap when any of the follow occur:

- Power is first applied to the system
- A console `initialize` command is issued and the auto_action environment variable is set to "Boot."
- The boot_reset environment variable is set to "On."
- A cold bootstrap is requested by system software.

The console must perform the following steps in the cold bootstrap sequence:

1. Perform a system initialization
2. Size memory
3. Test sufficient memory for bootstrapping
4. Load PALcode
5. Build a valid Hardware Restart Parameter Block (HWRPB)
6. Build a valid Memory Data Descriptor Table in the HWRPB
7. Initialize bootstrap page tables and map initial regions
8. Locate and load the system software primary bootstrap image
9. Initialize processor state on all processors
10. Transfer control to the system software primary bootstrap image

The steps leading to the transfer of control to system software may be performed in any order. The final state seen by system software is defined, but the implementation-specific sequence of these steps is not. Prior to beginning a bootstrap, the console must clear any internally pended restarts to any processor.

2.4.2 Loading of System Software

The console uses the boot_dev environment variable to determine the bootstrap device and the path to that device. These environment variables contain lists of bootstrap devices and paths; each list element specifies the complete path to a given bootstrap device. If multiple elements are specified, the console attempts to load a bootstrap image from each in turn.

The console uses the bootdef_dev, boot_dev, and booted_dev environment variables as follows:

1. At console initialization, the console sets the bootdef_dev and boot_dev environment variables to be equivalent. The format of these environment variables is determined by the console implementation and is independent of the console presentation layer; the value may be interpreted and modified by system software.
2. When a bootstrap results from a `boot` command that specifies a bootstrap device list, the console uses the list specified with the command. The console modifies boot_dev to contain the specified device list. Note that this may require conversion from the presentation layer format to the registered format.
3. When a bootstrap is the result of a `boot` command that does not specify a bootstrap device list, the console uses the bootstrap device list contained in the bootdef_dev environment variable. The console copies the value of bootdef_dev to boot_dev.
4. When a bootstrap is not the result of a `boot` command, the console uses the bootstrap device list contained in the boot_dev environment variable. The console does not modify the contents of boot_dev.
5. The console attempts to load a bootstrap image from each element of the bootstrap device list. If the list is exhausted prior to successfully transferring control to system software, the bootstrap attempt fails and the subsequent console action is determined by auto_action.
6. The console indicates the actual bootstrap path and device used in the booted_dev environment variable. The console sets booted_dev after loading the primary bootstrap image and prior to transferring control to system software. The booted_dev format follows that of a boot_dev list element.

7. If the bootstrap device list is empty, bootdef_dev or boot_dev are null, and the action is implementation-specific. The console may remain in console I/O mode or attempt to locate a bootstrap device in an implementation-specific manner.

The boot_file and boot_osflags environment variables are used as default values for the bootstrap filename and option flags. The console indicates the actual bootstrap image filename (if any) and option flags for the current bootstrap attempt in the booted_file and booted_osflags and environment variables. The boot_file default bootstrap image filename is used whenever the bootstrap requires a filename and either none was specified on the `boot` command or the bootstrap was initiated by the console as the result of a major state transition. The console never interprets the bootstrap option flags, but simply passes them between the console presentation layer and system software.

2.4.3 Warm Bootstrapping in a Uniprocessor Environment

The actions of the console on a warm bootstrap are a subset of those for a cold bootstrap. A system bootstrap will be a warm bootstrap whenever the boot_reset environment variable is set to "Off" (46 $4E4F_{16}$) and console internal state permits.

The console program performs the following steps in the warm bootstrap sequence.

1. Locates and validates the Hardware Reset Parameter Block (HWRPB)
2. Locates and loads the system software primary bootstrap image
3. Initializes processor state on all processors
4. Initializes bootstrap page tables and maps initial regions
5. Transfers control to the system software primary bootstrap image

At warm bootstrap, the console does not load PALcode, does not modify the Memory Data Descriptor Table, and does not reinitialize any environment variables. If the console cannot locate and validate the previously initialized HWRPB, the console must initiate a cold bootstrap. Prior to beginning a bootstrap, the console must clear any internally pended restarts to any processor.

2.4.4 Multiprocessor Bootstrapping

Multiprocessor bootstrapping differs from uniprocessor bootstrapping primarily in synchronization between processors. In a shared memory system, processors cannot independently load and start system software; bootstrapping is controlled by the primary processor.

DEC 4000 AXP systems always select CPU0 as the primary processor. The secondary processor polls a mailbox for a start address.

2.4.5 Boot Devices

The supported boot devices shown in Table 2–8 are determined by the console's device drivers.

Table 2–8 Supported Boot Devices

Adapter	Bus	Device	Name
I/O module	Ethernet	TGEC	EZAn
I/O module	DSSI/SCSI	Disk	DUan/DKan
I/O module	DSSI/SCSI	Tape	MUan/MKan

3

Running System Diagnostics

This chapter provides information on how to run system diagnostics.

- Section 3.1 describes how to run ROM-based diagnostics, including error reporting utilities, and loopback tests.
- Section 3.2 describes how to run DSSI internal device tests.
- Section 3.3 describes the DEC VET verifier and exerciser software.
- Section 3.4 describes how to run UETP environmental test package software.
- Section 3.5 describes acceptence testing and initialization procedures.

3.1 Running ROM-Based Diagnostics

DEC 4000 AXP ROM-based diagnostics (RBDs), which are part of the console firmware that is loaded from the FEPROM on the I/O module, offer many powerful diagnostic utilities, including the ability to examine error logs from the console environment and run system- or device-specific exercisers.

Unlike previous systems, DEC 4000 AXP RBDs rely on exerciser modules, rather than functional tests to isolate errors. The exercisers are designed to run concurrently, providing a maximum bus interaction between the console drivers and the target devices.

The multitasking ability of the console firmware allows you to run diagnostics in the background (using the background operator "&" at the end of the command). You run RBDs by using console commands.

RBDs can be separated into four types of utilities:

1. System or device diagnostic test/exercisers using the `test` command (Section 3.1.1).

   The `test` command is the primary diagnostic for acceptance testing and console environment diagnosis.

2. Three related commands are used to list system bus FRUs, report the status of RBDs in progress, and report errors:
    - The `show fru` command (Section 3.1.2) reports system bus FRUs, module part numbers, hardware and software revision numbers, and summary error information.
    - The `show_status` command (Section 3.1.3) reports the error count and status of RBD test/exercisers currently in progress.
    - The `show error` command (Section 3.1.4) reports errors captured by test-directed diagnostics (TDD), via the RBDs, and by symptom-directed diagnostics (SDD), via the operating system.
3. Several commands allow you to perform extended testing and exercising of specific system components. These commands are used for troubleshooting and are not needed for routine acceptance testing:
    - The `memexer` command (Section 3.1.5) exercises memory by running a specified number of memory tests. The tests are run in the background.
    - The `memexer_mp` command (Section 3.1.6) tests memory in a multiprocessor system by running a specified number of memory exerciser sets. The tests are run in the background.
    - The `exer_read` command (Section 3.1.7) tests a disk by performing random reads on the device.
    - The `exer_write` command (Section 3.1.8) tests a disk by performing random writes to the specified device.
    - The `fbus_diag` command (Section 3.1.9) tests the Futurebus+ modules.
    - The `show_mop_counters` command (Section 3.1.10) is used to read the MOP counters.
    - The `clear_mop_counters` command (Section 3.1.11) is used to reset the MOP counters.
4. Loopback tests for testing console and Ethernet ports (Section 3.1.12)

In addition to the four utilities listed above, there are two diagnostic-related commands. The `kill` and `kill_diags` commands (Section 3.1.13) are used to terminate diagnostics.

3.1.1 test

The `test` command runs firmware diagnostics for the entire system, specified subsystems, or specific devices. These firmware diagnostics are run in the background. When the tests are successfully completed, the message "tests done" is displayed. If any of the tests fail, a failure message is displayed.

If you do not specify an argument with the `test` command, all tests except those for tape drives are performed.

Note

By default, no write tests are performed on disk; and read and write tests are performed for tape drives. You need a scratch tape to test tape drives.

Early systems may not support RBD testing for tape drives.

All tests run concurrently for a minimum of 30 seconds. Tests complete when all component tests have completed at least one pass. Test passes are repeated for any component that completes its test before other components.

The run time of a test is proportional to the amount of memory to be tested and the number of disk and tape drives to be tested. Running `test all` on a system with fully configured 512-MB memory takes approximately 10 minutes to complete.

Synopsis:

test ([all] [cpu] [disk] [tape] [dssi] [scsi] [fbus] [memory] [ethernet] [device_list])

Arguments:

[all]	Firmware diagnostics will test/exercise all the devices present in the system configuration: CPU, disk, tape, DSSI subsystem, SCSI subsystem, Futurebus+ subsystem, memory, Ethernet, and I/O devices.
[cpu]	Firmware diagnostics will test backup cache and memory coherency.
[disk]	Firmware diagnostics will perform read-only tests of all disk drives present in the system. One pass consists of seeking to a random block on the disk and reading a packet of 2048 bytes and repeating until 512 packets are read.
[tape]	Firmware diagnostics will perform read and write tests of all the tape devices present in the system. Testing the tape drives requires that a scratch tape be loaded in the tape drive.
[dssi]	Firmware diagnostics will test the DSSI subsystem, including read-only tests of all DSSI disks, and read-write tests for tape drives.

[scsi]	Firmware diagnostics will test the SCSI subsystem, including read-only tests of all SCSI disks and read-write tests for SCSI tape drives.
[fbus]	Firmware diagnostics will instruct all Futurebus+ modules to perform extended category default self-tests.
[memory]	Firmware diagnostics will test memory modules present in the system.
[ethernet]	Firmware diagnostics will test the Ethernet logic.
[device_list]	Use the device_list argument to specify disk, tape, or Futurebus+ devices to be tested. As with all the RBDs, uses the exer script to perform read-only tests on the specified disk devices, and read-write tests for tape drives. Legal devices are disk, tape, and Futurebus+ device names.

Examples:

```
>>> test
tests done
>>>
```

```
>>> test

*** Soft Error - Error #1 - Lower SCSI Continuity Card Missing

Diagnostic Name        ID             Device  Pass  Test  Hard/Soft
31-JUL-1992
io_test          0000032d       scsi_low_con     1     1     0    1
14:23:18

*** End of Error ***
>>>
```

3.1.2 show fru

The show fru command reports FRU and error information for the following FRUs based on the serial control bus EEPROM data:

- CPU modules
- Memory modules
- I/O modules
- Futurebus+ modules

For each of the above FRUs, the slot position, option, part, revision, and serial numbers, as well as any reported symptom-directed diagnostics (SDD) and test-directed diagnostics (TDD) event logs are displayed.

Synopsis:

show fru ([target [target . . .]])

Arguments:

[target] CPU{0,1}, mem{0,1,2,3}, io, fbus, and fban.

Examples:

```
>>> show fru
  ❶    ❷      ❸              ❹        ❺            ❻
                             Rev                   Events Logged
 Slot   Option  Part#         Hw Sw   Serial#       SDD     TDD
   1    IO      B2101-AA      D3 2    AY21739158    00      00
   2
   3    CPU0    B2001-AA      D1 0    AY21328712    00      00
   4
   5
   6
   7    MEM3    B2002-BA      B1 0    GA21700025    00      00

Futurebus+ Nodes
          ❼       Rev                                    ❽
 Slot   Option  Part#    Hw   Fw    Serial#          Description
   1
   2
   3    fbc0    B2102-AA B02  X1.53 ML22000053    Fbus+ Profile_B Exerciser
   4
   5
   6
>>>
```

❶ Slot number for FRU (slots 1–7 right to left)

Slot 1: I/O module
Slot 2, 3: CPU modules
Slot 4–7: Memory modules

❷ Option name (I/O, CPU#, or MEM#)

❸ Part number of option

❹ Revision numbers (hardware and firmware)

❺ Serial number

❻ Events logged:

SDD: Number of symptom-directed diagnostic events logged by the operating system, or in the case of memory, by the operating system and firmware diagnostics.
TDD: Number of test-directed diagnostic events logged by the firmware diagnostics.

❼ Futurebus+ option name, fb*an*, where:

fb indicates Futurebus+ option
a indicates corresponding Futurebus+ slot a–f (1–6)
n indicates the Futurebus+ node number, 0 or 1

❽ Description of Futurebus+ module

3.1.3 show_status

The show_status command reports one line of information per executing diagnostic. The information includes ID, diagnostic program, device under test, error counts, passes completed, bytes written and read.

Many of the diagnostics run in the background and provide information only if an error occurs. Use the show_status command to display the progress of diagnostics.

The following command string is useful for periodically displaying diagnostic status information for diagnostics running in the background:

>>> while true;show_status;sleep n;done

Where *n* is the number of seconds between show_status displays.

Synopsis:

show_status

Examples:

```
>>> show_status
     ❶        ❷           ❸       ❹   ❺        ❻             ❼
     ID      Program       Device   Pass Hard/Soft Bytes Written    Bytes Read
-------- ------------ ------------ ------ --------- ------------- -------------
00000001         idle       system      0    0    0             0             0
000000ea      memtest       memory      2    0    0      67108864      67108864
000000f1     exer_kid dub0.0.0.1.0      1    0    0             0             0
000000f2     exer_kid duc0.6.0.2.0      1    0    0             0             0
000000f3     exer_kid dud0.7.0.3.0      1    0    0             0             0
000000f4     exer_kid dka0.0.0.0.0      1    0    0             0             0
>>>
```

❶ Process ID

❷ Program module name

❸ Device under test

❹ Diagnostic pass count

❺ Error count (hard and soft): Soft errors are not usually fatal; hard errors halt the system or prevent completion of the diagnostics.

❻ Bytes successfully written by diagnostic

❼ Bytes successfully read by diagnostic

3.1.4 show error

The `show error` command reports error information based on the serial control bus EEPROM data. Both the operating system and the ROM-based diagnostics log errors to the serial control bus EEPROMs. This functionality provides the ability to generate an error log from the console environment.

A closely related command, `show fru` (Section 3.1.2), reports FRU and error information for FRUs.

Synopsis:

show error ([target [target . . .]])

Arguments:

[target] CPU{0,1}, mem{0,1,2,3}, and io.

Examples:

```
>>> show error mem3
Test Directed Errors

No Entries Found

Symptom Directed Entries
MEM3 Module EEROM Event Log
 ❶      ❷      ❸       ❹              ❺           ❻
Entry   Offest  RAM #   Bit Mask       Multi-Chip      Event Type
  0     383     9       0001                0           10
  1     402     10      0001                1           10
  2     402     11      0001                1           10
  3     402     2       0001                1           10
  4     402     3       0001                1           10
  5     404     0       0001                1           10
  6     404     1       0001                1           10
  7     408     12      0001                0           10

Entry   Error Mask      Device #        Event Type
  15    f01             71                  0
>>>
```

❶ Event log entry number

❷ Offset address of fault in RAM

❸ RAM number—indicates the RAM location on the board

❹ Four-bit bit field value, indicates bit in DRAM

Using the offset, RAM number, and bitmask, you can determine the location of the specific cell in memory.

❺ Multi-chip (0=no, 1=yes)—indicates that a group of entries are the result of a single error.

❻ Event type:

11—DRAM hard-failure
01—Correctable read data (CRD) error
10—Uncorrectable error
00—Other (non-DRAM error)

3.1.5 memexer

The memexer command tests memory by running a specified number of memory exercisers. The exercisers are run in the background and nothing is displayed unless an error occurs. Each exerciser tests all available memory in 2-MB blocks for each pass.

To terminate the memory tests, use the kill command to terminate an individual diagnostic or the kill_diags command to terminate all diagnostics. Use the show_status display to determine the process ID when killing an individual diagnostic test.

Synopsis:

memexer [number]

Arguments:

[number]	Number of memory exercisers to start. The default is 1. The number of exercisers, as well as the length of time for testing, depends on the context of the testing. Generally, running 3–5 exercisers for 15 minutes to 1 hour is sufficient for troubleshooting most memory problems.

Examples:

```
>>> memexer 4
>>> show_status
      ID      Program       Device   Pass Hard/Soft Bytes Written    Bytes Read
-------- ------------ ------------ ------ --------- ------------- -------------
00000001         idle       system      0    0    0             0             0
000000c7      memtest       memory      3    0    0     635651584      62565154
000000cc      memtest       memory      2    0    0     635651584      62565154
000000d0      memtest       memory      2    0    0     635651584      62565154
000000d1      memtest       memory      3    0    0     635651584      62565154
>>> kill_diags
>>>
```

3.1.6 memexer_mp

The memexer_mp command tests memory cache coherency in a multiprocessor system by running a specified number of memory exerciser sets. A set is a memory test that runs on each processor checking alternate longwords. The exercisers are run in the background and nothing is displayed unless an error occurs.

To terminate the memory tests, use the kill command to terminate an individual diagnostic or the kill_diags command to terminate all diagnostics. Use the show_status display to determine the process ID when killing an individual diagnostic test.

Synopsis:

memexer_mp [number]

Arguments:

[number]	Number of memory exerciser sets to start. The default is 1. The number of exercisers, as well as the length of time for testing, depends on the context of the testing. Generally, running 2 or 3 exercisers for 5 minutes is sufficient.

Examples:

```
>>> memexer_mp 2
>>> kill_diags
>>>
```

3.1.7 exer_read

The exer_read command tests a disk by performing random reads of 2048 bytes on one or more devices. The exercisers are run in the background and nothing is displayed unless an error occurs.

The tests continue until one of the following conditions occurs:

1. All blocks on the device have been read for a passcount of d_passes (default is 1).
2. The exer_read process has been terminated via the kill or kill_diags commands, or Ctrl/C.
3. The specified time has elapsed.

To terminate the read tests, enter Ctrl/C, or use the kill command to terminate an individual diagnostic or the kill_diags command to terminate all diagnostics. Use the show_status display to determine the process ID when killing an individual diagnostic test.

Synopsis:

exer_read [-sec seconds] [device_name device_name . . .]

Arguments:

[device_name] One or more device names to be tested. The default is du*.* dk*.* to test all DSSI and SCSI disks that are on line.

Options:

[-sec seconds] Number of seconds to run exercisers. If you do not enter the number of seconds, the tests will run until d_passes have completed (d_passes default is 1).

If you want to test the entire disk, run at least one pass across the disk. If you do not need to test the entire disk, run the test for 5 or 10 minutes.

Examples:

```
>>> exer_read

failed to send command to pkc0.1.0.2.0
failed to send Read to dkc100.1.0.2.0


*** Hard Error - Error #5 -

Diagnostic Name        ID             Device  Pass  Test  Hard/Soft
31-JUL-1992
exer_kid         00000175       dkc100.1.0.2     0     0     1    0
14:54:18
Error in read of 0 bytes at location 014DD400 from device dkc100.1.0.2.0

*** End of Error ***
>>>
```

3.1.8 exer_write

The exer_write command tests a disk by performing random writes on one or more devices. The exercisers are run in the background and nothing is displayed unless an error occurs.

The exer_write tests cause the device to seek to a random block and read a 2048-byte packet of data, write that same data back to the same location on the device, read the data again, and compare it to the data originally read.

The tests continue until one of the following conditions occurs:

1. All blocks on the device have been read for a passcount of d_passes (default is 1).
2. The exer_read process has been terminated via the kill or kill_diags commands, or Ctrl/C.
3. The specified time has elapsed.

To terminate the read tests, enter Ctrl/C, or use the kill command to terminate an individual diagnostic or the kill_diags command to terminate all diagnostics. Use the show_status display to determine the process ID when killing an individual diagnostic test.

Caution

Running the exer_write diagnostic may distroy data on the specified disk.

Synopsis:

exer_write [-sec seconds] [device_name device_name...]

Arguments:

[device_name] One or more device names to be tested. The default is du*.* dk*.* to test all DSSI and SCSI disks that are on line.

Options:

[-sec seconds] Number of seconds to run exercisers. If you do not enter the number of seconds, the tests will run until d_passes have completed (d_passes default is 1).

If you want to test the entire disk, run at least one pass across the disk. If you do not need to test the entire disk, run the test for 5 or 10 minutes.

Examples:

```
>>> exer_write dka0

EXECUTING THIS COMMAND WILL DESTROY DISK DATA
    OR DATA ON THE SPECIFIED DEVICES
Do you really want to continue? [Y/(N)]: y
failed to send command to pkc0.1.0.2.0
failed to send Read to dkc100.1.0.2.0

*** Hard Error - Error #5 -

Diagnostic Name        ID             Device  Pass  Test  Hard/Soft
31-JUL-1992
exer_kid         0000012e       dka0.0.0.0     0     0     1    0
15:21:22
Error in read of 0 bytes at location 017B3400 from device dka0.0.0.0.0

*** End of Error ***

failed to send command to pka0.0.0.0.0
failed to send Read to dka0.0.0.0.0
>>>
```

3.1.9 fbus_diag

The `fbus_diag` command is used to start execution of a diagnostic test script onboard a specific Futurebus+ device.

The `fbus_diag` comand uses the Futurebus+ standard test CSR interface to initiate commands on specific Futurebus+ devices, waits for tests to complete, and then reports the results to the console. If an error is reported by the Futurebus+ node, the diagnostic issues a dump buffer command to gain any available extended information that will also be reported to the console.

Refer to documentation for the specific Futurebus+ option for the recommended test procedures and form of the `fbus_diag` command to initiate module-resident diagnostics. For more information, consult the *Futurebus+ Handbook*.

Test categories that require a buffer pointer in the argument CSR will have a default buffer provided by this diagnostic if the user does not specify a buffer address.

Process options and command line arguments are used to specify the specific test or test script to be executed as well as the target Futurebus+ node for this command.

Synopsis:

fbus_diag [-rb] [-p pass_count] [-st test_number] [-cat test_group node [test_arg]

Arguments:

node	Specifies the device name of the Futurebus+ device to execute the test. Use the command `show device fb` to display the Futurebus+ device names.
[test_arg]	Specifies an argument to be passed to the Futurebus+ node in the test argument CSR. If this parameter is not specified and the category is either extended or system, the routine allocates a buffer and passes the buffer address through the test argument CSR.

Options:

[-rb]	Randomly allocates from memzone on each pass with a block size of 4096.
[-p]	(*pass_count*) Specifies the number of times to run the test. If 0, the test runs continuously. This overrides the value of the pass_count environment variable. In the absence of this option, pass_count is used. The default for pass_count is 1.
[-st]	(*test_number*) Specifies the test number to be run. The default is 0, which runs the default tests in the category.

[-cat] (*test_group*) Specifies the test category to be executed. The possible categories are as follows:

- Init: Initialization tests
- Extended: Extended tests (default category)
- System: System tests
- Manual: Manual tests
- *x*: Bit mask of the desired test categories

[-opt] (*test_option*) Specify the Test Start CSR Option field bits to be set. The possible option bits are as follows:

- Loop_error: Loop on test if an error is detected
- Loop_test: Loop on this test
- Cont_error: Continue if an error is detected
- x: Bit mask of the desired option bits

The default value for this qualifier is based on the current values in the global enviroment variables as follows:

- Loop_test: 1 if D_PASSES = = 0 ; 0 otherwise
- Loop_error: 1 if D_HARDERR = = "Loop" ; 0 otherwise
- Cont_error: 1 if D_HARDERR = = "Continue" ; 0 otherwise

3.1.10 show_mop_counter

The show_mop_counter command displays the MOP counters for the specified Ethernet port.

Synopsis:

show_mop_counter [port_name]

Arguments:

[port_name]	Specifies the Ethernet port for which to display MOP counters: eza0 for Ethernet port 0; ezb0 for Ethernet port 1.

Examples:

```
>>> show_mop_counter eza0

eza0 MOP Counters

DEVICE SPECIFIC:
 TI: 211 RI: 34834 RU: 1 ME: 0 TW: 0 RW: 0 BO: 0
 HF: 0 UF: 0 TN: 0 LE: 0 TO: 0 RWT: 33535 RHF: 33536  TC: 56

PORT INFO:
tx full: 0 tx index in: 2 tx index out: 2
rx index in: 3

MOP BLOCK:
 Network list size: 0

MOP COUNTERS:
Time since zeroed (Secs): 4588

TX:
 Bytes: 117068 Frames: 210
 Deferred: 1 One collision: 32 Multi collisions: 15
TX Failures:
 Excessive collisions: 0 Carrier check: 0 Short circuit: 0
 Open circuit: 0 Long frame: 0 Remote defer: 0
 Collision detect: 0

RX:
 Bytes: 116564 Frames: 194
 Multicast bytes: 16730668 Multicast frames: 36953
RX Failures:
 Block check: 0 Framing error: 0 Long frame: 0
 Unknown destination: 36953 Data overrun: 0 No system buffer: 18
 No user buffers: 0
>>>
```

3.1.11 clear_mop_counter

The `clear_mop_counter` command initializes the MOP counters for the specified Ethernet port.

Synopsis:

show_mop_counter [port_name]

Arguments:

[port_name]	Specifies the Ethernet port for which to initialize MOP counters: eza0 for Ethernet port 0; ezb0 for Ethernet port 1.

Examples:

```
>>> clear_mop_counter eza0
>>>
```

3.1.12 Loopback Tests

Internal and external loopback tests can be used to isolate a failure by testing segments of a particular control or data path. The loopback tests are a subset of the RBDs.

3.1.12.1 Testing the Auxiliary Console Port (exer)

Using a loopback connector (29–24795–00) and a form of the exer command, you can test the auxiliary serial port. Before running the loopback test, you must set the tt_allow_login environment variable to 1; after the test is completed, you must set tt_allow_login to 0.

Use the following commands to send a fixed data pattern through the auxiliary serial port:

```
>>> set tt_allow_login 1
>>> exer -bs 1 -a "wRc" -p 0 tta1 &
>>> kill_diags
>>> set tt_allow_login 0
>>>
```

In the above command, the portion in quotes (the write, read, and compare instruction) is case sensitive. The background operator &, at the end of the command, causes the loopback tests to run in the background. Nothing is displayed unless an error occurs.

To terminate the console loopback test, use the kill command to terminate the individual diagnostic or the kill_diags command to terminate all diagnostics. Use the show_status display to determine the process ID when killing an individual diagnostic test.

3.1.12.2 Testing the Ethernet Ports (netexer)

The netexer command performs an Ethernet port-to-port MOP loopback test between eza0 and ezb0. The network ports must be connected and terminated.

The loopback tests are run in the background. Nothing is displayed unless an error occurs.

To terminate the console loopback test, use the kill command to terminate the individual diagnostic or the kill_diags command to terminate all diagnostics. Use the show_status display to determine the process ID when killing an individual diagnostic test.

3.1.13 kill and kill_diags

The `kill` and `kill_diags` commands terminates diagnostics that are currently executing .

- The `kill` command terminates a specified process.
- The `kill_diags` command terminates all diagnostics.

Synopsis:

kill_diags

kill [PID . . .]

Arguments:

[PID . . .]	The process ID of the diagnostic to terminate. Use the `show_status` command to determine the process ID.

3.1.14 Summary of Diagnostic and Related Commands

Table 3–1 provides a summary of the diagnostic and related commands.

Table 3–1 Summary of Diagnostic and Related Commands

Command	Function	Reference
Acceptance Testing		
test	Test the entire system, subsystem, or specific device.	Section 3.1.1
Error Reporting and Diagnostic Status		
show fru	Reports system bus and Futurebus+ FRUs, module identification numbers, and summary error information.	Section 3.1.2
show_status	Reports the status of currently executing test/exercisers.	Section 3.1.3
show error	Reports some errors captured by diagnostics and operating system.	Section 3.1.4

(continued on next page)

Table 3–1 (Cont.) Summary of Diagnostic and Related Commands

Command	Function	Reference
Extended Testing/Troubleshooting		
memexer	Exercises memory by running a specified number of memory tests. The tests are run in the background.	Section 3.1.5
memexer_mp	Tests memory in a multiprocessor system by running a specified number of memory exerciser sets. The tests are run in the background.	Section 3.1.6
exer_read	Tests a disk by performing random reads on the specified device.	Section 3.1.7
exer_write	Tests a disk by performing random writes to the specified device.	Section 3.1.8
fbus_diag	Initiates onboard tests for a specified Futurebus+ device.	Section 3.1.9
show_mop_ counter	Displays the MOP counters for the specified Ethernet port.	Section 3.1.10
clear_mop_ counter	Initializes the MOP counters for the specified Ethernet port.	Section 3.1.11
Loopback Testing		
exer	Conducts loopback tests for the specified console port.	Section 3.1.12.1
netexer	Conducts loopback tests for the Ethernet ports.	Section 3.1.12.2
Diagnostic-Related Commands		
kill	Terminates a specified process.	Section 3.1.13
kill_diags	Terminates all currently executing diagnostics.	Section 3.1.13

3.2 DSSI Device Internal Tests

A DSSI storage device may fail either during initial power-up or during normal operation. In both cases, the failure is indicated by the lighting of the red Fault LED on the drive's front panel.

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

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

- Controller errors are caused by the hardware associated with the controller function of the drive module. A controller error is fatal to the operation of the drive, since the controller cannot establish a logical connection to the host. The red Fault LED comes on. If this occurs, replace the drive module.
- Drive errors are caused by the hardware associated with the drive control function of the drive module. These errors are not fatal to the drive, since the drive can establish a logical connection and report the error to the host. Both LEDs go out for about 1 second, then the red Fault LED comes on. In this case, run either DRVTST, DRVEXR, or PARAMS via the `set host -dup` command, as described in the drive's service documentation, to determine the error code.

Three configuration errors are often the cause of drive errors:

- More than one node with the same bus node ID number
- Identical node names
- Identical MSCP unit numbers

The first error cannot be detected by software. Use the `show device` command (Section 6.2) to display the second and third types of errors. This command displays each device along with such information as bus node ID, unit number, and node name.

If the device is connected to the front panel of the storage compartment, you must install a bus node ID plug in the corresponding socket on the front panel. If the device is not connected to the front panel, it reads the bus node ID from the three-switch DIP switch on the side of the drive.

DSSI storage devices contain the following local programs:

DIRECT	A directory, in DUP-specified format, of available local programs
DRVTST	A comprehensive drive functionality verification test
DRVEXR	A utility that exercises the device
HISTRY	A utility that saves information retained by the drive, including the internal error log
ERASE	A utility that erases all user data from the disk
VERIFY	A utility that is used to determine the amount of "margin" remaining in on-disk structures
DKUTIL	A utility that displays disk structures and disk data
PARAMS	A utility that allows you to look at or change drive status, history, parameters, and the internal error log

Use the set host -dup command to access the local programs listed above. Example 3–1 provides an abbreviated example of running DRVTST for a device (Bus node 2 on Bus 0).

Caution

When running internal drive tests, always use the default (0 = No) in responding to the "Write/read anywhere on medium?" prompt. Answering Yes could destroy data.

Example 3–1 Running DRVTST

```
>>> set host -dup -task drvtst dub0

Starting DUP server...
Copyright (C) 1992  Digital Equipment Corporation
Write/read anywhere on medium? [1=Yes/(0=No)] Return
  5 minutes to complete.
GAMMA::MSCP$DUP  17-MAY-1992 12:51:20 DRVTST  CPU=  0 00:00:09.29 PI=160
GAMMA::MSCP$DUP  17-MAY-1992 12:51:40 DRVTST  CPU=  0 00:00:18.75 PI=332
GAMMA::MSCP$DUP  17-MAY-1992 12:52:00 DRVTST  CPU=  0 00:00:28.40 PI=503
   .
   .
   .
GAMMA::MSCP$DUP  17-MAY-1992 12:55:42 DRVTST  CPU=  0 00:02:13.41 PI=2388
Test passed.

Stopping DUP server...
>>>
```

Example 3–2 provides an abbreviated example of running DRVEXR for an RF-series disk (Bus node 2 on Bus 0).

Example 3–2 Running DRVEXR

```
>>> set host -dup -task drvexr dub0

Starting DUP server...
Copyright (C) 1992  Digital Equipment Corporation
Write/read anywhere on medium? [1=Yes/(0=No)] Return
Test time in minutes? [(10)-100] Return
Number of sectors to transfer at a time? [0 - 50] 5
Compare after each transfer? [1=Yes/(0=No)]:  Return
Test the DBN area? [2=DBN only/(1=DBN and LBN)/0=LBN only]:  Return
 10 minutes to complete.
GAMMA::MSCP$DUP  17-MAY-1992 13:02:40 DRVEXR  CPU=  0 00:00:25.37 PI=1168
GAMMA::MSCP$DUP  17-MAY-1992 13:03:00 DRVEXR  CPU=  0 00:00:29.53 PI=2503
GAMMA::MSCP$DUP  17-MAY-1992 13:03:20 DRVEXR  CPU=  0 00:00:33.89 PI=3835
   .
   .
   .
GAMMA::MSCP$DUP  17-MAY-1992 13:12:24 DRVEXR  CPU=  0 00:02:24.19 PI=40028
   13332 operations completed.
   33240 LBN blocks (512 bytes) read.
       0 LBN blocks (512 bytes) written.
   33420 DBN blocks (512 bytes) read.
       0 DBN blocks (512 bytes) written.
       0 bytes in error (soft).
       0 uncorrectable ECC errors.
Complete.

Stopping DUP server...
>>>
```

Refer to the *RF-Series Integrated Storage Element Service Guide* for instructions on running these programs.

3.3 DEC VET

Digital's DEC Verifier and Exerciser Tool (DEC VET) software is a multipurpose system maintenance tool that performs exerciser-oriented maintenance testing. DEC VET runs on both OpenVMS AXP and DEC OSF/1 operating systems. DEC VET consists of a manager and exercisers that test devices. The DEC VET manager controls these exercisers.

DEC VET exercisers test system hardware and the operating system.

DEC VET supports various exerciser configurations, ranging from a single device exerciser to full system loading—that is, simultaneous exercising of multiple devices.

Refer to the *DEC Verifier and Exerciser Tool User's Guide* (AA–PTTMA–TE) for instructions on running DEC VET.

3.4 Running UETP

The User Environment Test Package (UETP) tool is an OpenVMS AXP software package designed to test whether the OpenVMS AXP operating system is installed correctly. UETP software puts the system through a series of tests that simulate a typical user environment, by making demands on the system that are similar to demands that might occur in everyday use.

Run UETP after system installation when OpenVMS AXP is running; or when you need to run stress tests to pinpoint intermittent errors.

UETP is not a diagnostic program; it does not attempt to test every feature exhaustively. When UETP runs to completion without encountering unnrecoverable errors, the system being tested is ready for use.

UETP exercises devices and functions that are common to all VMS and OpenVMS AXP systems, with the exception of optional features, such as high-level language compilers. The system components tested include the following:

- Most standard peripheral devices
- The system's multiuser capability
- DECnet for OpenVMS AXP software

3.4.1 Summary of UETP Operating Instructions

This section summarizes the procedure for running all phases of UETP with default values.

1. Log in to the SYSTEST account as follows:

```
Username: SYSTEST
Password:
```

Caution

Because the SYSTEST and SYSTEST_CLIG accounts have privileges, unauthorized use of these accounts might compromise the security of your system.

2. Make sure no user programs are running and no user volumes are mounted.

Caution

By design, UETP assumes and requests the exclusive use of system resources. If you ignore this restriction, UETP may interfere with applications that depend on these resources.

3. After you log in, check all devices to be sure that the following conditions exist:
   - All devices you want to test are powered up and are on line to the system.
   - Scratch disks are mounted and initialized.
   - Disks contain a directory named [SYSTEST] with OWNER_ UIC=[1,7]. (You can create this directory with the DCL command CREATE/DIRECTORY.)
   - Scratch magnetic tape reels are physically mounted on each drive you want tested and are initialized with the label UETP (using the DCL command INITIALIZE). Make sure magnetic tape reels contain at least 600 feet of tape.
   - Scratch tape cartridges have been inserted in each drive you want to test and are initialized with the label UETP.
   - Line printers and hardcopy terminals have plenty of paper.
   - Terminal characteristics and baud rate are set correctly (see the user's guide for your terminal).
4. To start UETP, enter the following command and press Return:

```
$ @UETP
```

   UETP responds with the following question:

```
Run "ALL" UETP phases or a "SUBSET" [ALL]?
```

   Press Return to choose the default response enclosed in brackets. UETP responds with three more questions in the following sequence:

```
How many passes of UETP do you wish to run [1]?
How many simulated user loads do you want [n]?
Do you want Long or Short report format [Long]?
```

   Use the default values when acceptance testing with UETP. For stress testing, enter your own values.

Press Return after each prompt. After you answer the last question, UETP initiates its entire sequence of tests, which run to completion without further input. The final message should look like the following:

```
************************************************
*                                              *
   END OF UETP PASS 1 AT 20-JUL-1992 16:30:09.38
*                                              *
************************************************
```

5. After UETP runs, check the log files for errors. If testing completes successfully, the OpenVMS AXP operating system is working properly.

Note

After a run of UETP, you should run the Error Log Utility to check for hardware problems that can occur during a run of UETP. For information on running the Error Log Utility, refer to the *VMS Error Log Utility Manual*.

If UETP does not complete successfully, refer to Section 3.4.11.

3.4.2 System Disk Requirements

Before running UETP, be sure that the system disk has at least 1200 blocks available. Systems running more than 20 load test processes may require a minimum of 2000 available blocks. If you run multiple passes of UETP, log files will accumulate in the default directory and further reduce the amount of disk space available for subsequent passes.

If disk quotas are enabled on the system disk, you should disable them before you run UETP.

3.4.3 Preparing Additional Disks

To prepare each disk drive in the system for UETP testing, use the following procedure:

1. Place a scratch disk in the drive and spin up the drive. If a scratch disk is not available, use any disk with a substantial amount of free space. UETP does not overwrite existing files on any volume. If your scratch disk contains files that you want to keep, do not initialize the disk; go to step 3.
2. If the disk does not contain files you want to save, initialize it. For example:

```
$ INITIALIZE DUA1: TEST1
```

This command initializes DUA1, and assigns the volume label TEST1 to the disk. All volumes must have unique labels.

3. Mount the disk. For example:

   ```
   $ MOUNT/SYSTEM DUA1: TEST1
   ```

   This command mounts the volume labeled TEST1 on DUA1. The /SYSTEM qualifier indicates that you are making the volume available to all users on the system.

4. UETP uses the [SYSTEST] directory when testing the disk. If the volume does not contain the directory [SYSTEST], you must create it. For example:

   ```
   $ CREATE/DIRECTORY/OWNER_UIC=[1,7] DUA1:[SYSTEST]
   ```

   This command creates a [SYSTEST] directory on DUA1 and assigns a user identification code (UIC) of [1,7]. The directory must have a UIC of [1,7] to run UETP.

   If the disk you have mounted contains a root directory structure, you can create the [SYSTEST] directory in the [SYS0.] tree.

3.4.4 Preparing Magnetic Tape Drives

Set up magnetic tape drives that you want to test by doing the following:

1. Place a scratch magnetic tape with at least 600 feet of magnetic tape in the tape drive. Make sure that the write-enable ring is in place.
2. Position the magnetic tape at the beginning-of-tape (BOT) and put the drive on line.
3. Initialize each scratch magnetic tape with the label UETP. For example, if you have physically mounted a scratch magnetic tape on MTA1, enter the following command and press Return:

   ```
   $ INITIALIZE MTA1: UETP
   ```

   Magnetic tapes must be labeled UETP to be tested. As a safety feature, UETP does not test tapes that have been mounted with the MOUNT command.

3.4.5 Preparing Tape Cartridge Drives

Set up tape cartridge drives that you want to test by doing the following:

1. Insert a scratch tape cartridge in the tape cartridge drive.
2. Initialize the tape cartridge. For example:

   ```
   $ INITIALIZE MKE0: UETP
   ```

Tape cartridges must be labeled UETP to be tested. As a safety feature, UETP does not test tape cartridges that have been mounted with the MOUNT command.

3.4.5.1 TLZ06 Tape Drives

During the initialization phase, UETP sets a time limit of 6 minutes for a TLZ06 unit to complete the UETTAPE00 test. If the device does not complete the UETTAPE00 test within the alloted time, UETP displays a message similar to the following:

```
-UETP-E-TEXT, UETTAPE00.EXE testing controller MKA was stopped ($DELPRC) at 16:23:23.07
       because the time out period (UETP$INIT_TIMEOUT) expired or
       because it seemed hung or because UETINIT01 was aborted.
```

To increase the timeout value, type a command similar to the following before running UETP:

```
$  DEFINE/GROUP UETP$INIT_TIMEOUT "0000 00:08:00.00"
```

This example defines the initialization timeout value as 8 minutes.

3.4.6 Preparing RRD42 Compact Disc Drives

To run UETP on an RRD42 compact disc drive, you must first load the test disc that you received with your compact disc drive unit.

3.4.7 Preparing Terminals and Line Printers

Terminals and line printers must be turned on to be tested by UETP. They must also be on line. Check that line printers and hardcopy terminals have enough paper. The amount of paper required depends on the number of UETP passes that you plan to execute. Each pass requires two pages for each line printer and hardcopy terminal.

Check that all terminals are set to the correct baud rate and are assigned appropriate characteristics (see the user's guide for your terminal).

Spooled devices and devices allocated to queues fail the initialization phase of UETP and are not tested.

3.4.8 Preparing Ethernet Adapters

Make sure that no other processes are sharing the Ethernet adapter device when you run UETP.

Note

UETP will not test your Ethernet adapter if DECnet for OpenVMS AXP or another application has the device allocated.

Because either DECnet for OpenVMS AXP or the LAT terminal server might also try to use the Ethernet adapter (a shareable device), you must shut down DECnet for OpenVMS AXP and the LAT terminal server before you run the device test phase, if you want to test the Ethernet adapter.

3.4.9 DECnet for OpenVMS AXP Phase

The DECnet for OpenVMS AXP phase of UETP uses more system resources than other tests. You can, however, minimize disruptions to other users by running the test on the "least busy" node.

By default, the file UETDNET00.COM specifies the node from which the DECnet for OpenVMS AXP test will be run. To run the DECnet for OpenVMS AXP test on a different node, enter the following command before you invoke UETP:

```
$ DEFINE/GROUP UETP$NODE_ADDRESS node_address
```

This command equates the group logical name UETP$NODE_ADDRESS to the node address of the node in your area on which you want to run the DECnet for OpenVMS AXP phase of UETP.

For example:

```
$ DEFINE/GROUP UETP$NODE_ADDRESS 9.999
```

Note

When you use the logical name UETP$NODE_ADDRESS, UETP tests only the first active circuit found by NCP. Otherwise, UETP tests all active testable circuits.

When you run UETP, a router node attempts to establish a connection between your node and the node defined by UETP$NODE_ADDRESS. Occasionally, the connection between your node and the router node might be busy or nonexistent. When this happens, the system displays the following error messages:

```
%NCP-F-CONNEC, Unable to connect to listener
-SYSTEM-F-REMRSRC, resources at the remote node were insufficient

%NCP-F-CONNEC, Unable to connect to listener
-SYSTEM-F-NOSUCHNODE, remote node is unknown
```

3.4.10 Termination of UETP

At the end of a UETP pass, the master command procedure UETP.COM displays the time at which the pass ended. In addition, UETP.COM determines whether UETP needs to be restarted.

At the end of an entire UETP run, UETP.COM deletes temporary files and does other cleanup activities.

Pressing Ctrl/Y or Ctrl/C lets you terminate a UETP run before it completes normally. Normal completion of a UETP run, however, includes the deletion of miscellaneous files that have been created by UETP for the purpose of testing. The use of Ctrl/Y or Ctrl/C might interrupt or prevent these cleanup procedures.

3.4.11 Interpreting UETP VMS Failures

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

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

- VMS Error Log Utility—Run the Error Log Utility to obtain a detailed report of hardware and system errors. Error log reports provide information about the state of the hardware device and I/O request at the time of each error. For information about running the Error Log Utility, refer to the *VMS Error Log Utility Manual* and Chapter 4 of this manual.
- Diagnostic facilities—Use the diagnostic facilities to test exhaustively a device or medium to isolate the source of the error.

3.4.12 Interpreting UETP Output

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

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

3.4.12.1 UETP Log Files

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

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

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

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

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

3.4.12.2 Possible UETP Errors

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

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

- Wrong quotas, privileges, or account
- UETINIT01 failure
- Ethernet device allocated or in use by another application

- Insufficient disk space
- Incorrect cluster setup
- Problems during the load test
- DECnet for OpenVMS AXP error
- Lack of default access for the FAL object
- Errors logged but not displayed
- No PCB or swap slots
- Hangs
- Bugchecks and machine checks

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

3.5 Acceptance Testing and Initialization

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

CPU modules
Memory modules
I/O module
Backplane
Storage devices
Futurebus+ options

1. Run the RBD acceptance tests using the `test` command.
2. Bring up the operating system.
3. Run DEC VET or UETP to test that the operating system is correctly installed. Refer to Section 3.3 for information on DEC VET. Refer to Section 3.4 for instructions on running UETP.

4
Error Log Analysis

This chapter provides information on how to interpret error logs reported by the operating system.

- Section 4.1 describes machine check/interrupts and how these errors are detected and reported.
- Section 4.2 describes the entry format used by the ERF/UERF error formatters.
- Section 4.3 describes how to translate the error log information using the OpenVMS AXP and DEC OSF/1 error formatters.
- Section 4.4 describes how to interpret the system error log to isolate the failing FRU.

4.1 Fault Detection and Reporting

Table 4–1 provides a summary of the fault detection and correction components of DEC 4000 AXP systems.

Generally, PALcode handles exceptions as follows:

- The PALcode determines the cause of the exception.
- If possible, it corrects the problem and passes control to the operating system for reporting before returning the system to normal operation.
- If a problem is not correctable, or if error/event logging is required, control is passed through the system control block (SCB) to the appropriate exception handler.

Table 4–1 DEC 4000 AXP Fault Detection and Correction

Component	Fault Detection/Correction Capability
KN430 Processor Module	
DECchip 21064 microprocessor	Error Detection and Correction (EDC) logic. For all data entering the 21064 microprocessor, single bits are checked and corrected; for all data exiting the 21064 microprocessor, the appropriate check bits are generated. A single-bit error on any of the four longwords being read can be corrected (per cycle).
Backup cache (B-cache)	EDC check bits on the data store; and parity on the tag store and control store.
MS430 Memory Modules	
Memory module	EDC logic protects data by detecting and correcting up to 2 bits per DRAM chip per gate array. The four bits of data per DRAM are spread across two gate arrays (one for even longwords, the other for odd longwords).
KFA40 I/O Module	
I/O module	DSSI/SCSI buses: Data parity is checked and generated. Lbus data transfers to Ethernet and SCSI/DSSI controllers: Data parity is checked and generated. Futurebus+ data transfers: Parity is checked and passed on.
System Bus	
System bus	Longword parity on command, address, and data.

4.1.1 Machine Check/Interrupts

The exceptions that result from hardware system errors are called machine check/interrupts. They occur when a system error is detected during the processing of a data request. There are three types of machine check/interrupts related to system events:

1. Processor machine check
2. System machine check
3. Processor corrected machine check

The causes for each of the machine check/interrupts are as follows. The system control block (SCB) vector through which PALcode transfers control to the operating system is shown in parentheses.

Processor Machine Check (SCB: 670)

Processor machine check errors are fatal system errors and immediately crash the system.

- The DECchip 21064 microprocessor detected one or more of the following uncorrectable data errors:
  - Uncorrectable B-cache data error
  - Uncorrectable memory data error (CU_ERR asserted)
  - Uncorrectable data from other CPU's B-cache (CU_ERR asserted)
- A B-cache tag or tag control parity error occurred
- Hard error status was asserted in response to:
  - A read data parity error
  - System bus timeouts (NOACK error bit asserted)—The bus responder detected a write data or command address error and did not acknowledge the bus cycle.

System Machine Check (SCB: 660)

A system machine check is a system detected error, external to the DECchip 21064 microprocessor and possibly not related to the activities of the microprocessor. It occurs when C_ERROR is asserted on the system bus.

Fatal errors:

- The I/O module detected a system bus error while serving as system bus commander:
  - System bus timeouts (NOACK error bit asserted)—The bus responder detected a write data or command address error and did not acknowledge the bus cycle
  - Uncorrectable data (CU-ERR asserted) from responder
- Any system bus device detected a command/address parity error
- A bus responder detected a write data parity error
- Memory or I/O system bus gate array detected an internal error (SYNC error)

Nonfatal errors:

- A memory module correctable error occurred
- Correctable B-cache errors were detected while the B-cache was providing data to the system bus (errors from other CPU)
- Duplicate tag store parity errors occurred

Processor Corrected Machine Check (SCB: 630)

Processor corrected machine checks are caused by B-cache errors that are detected and corrected by the DECchip 21064 microprocessor. These errors are nonfatal and result in an error log entry.

4.1.2 System Bus Transaction Cycle

In order to interpret error logs for system bus errors, you need a basic understanding of the system bus transaction cycle and the function of the commander, responder, and bystanders.

For any particular bus transaction cycle there is one commander (either CPU or I/O) that initiates bus transactions and one responder (memory, CPU, or I/O) that accepts or supplies data in response to a command/address from the system bus commander. A bystander is a system bus node (CPU, I/O, or memory) that is not addressed by a current system bus commander.

There are four system bus transaction types: read, write, exchange, and nut.

- Read and write transactions consist of a command/address cycle followed by two data cycles.
- Exchange transactions are used to replace the cache block when a cache block resource conflict occurs. They consist of a command/address cycle followed by four data cycles: two writes and two reads.
- Nut transactions consist of a command/address cycle and two dummy data cycles for which no data is transferred.

For more information, refer to the *DEC 4000 Model 600 Series Technical Manual.*

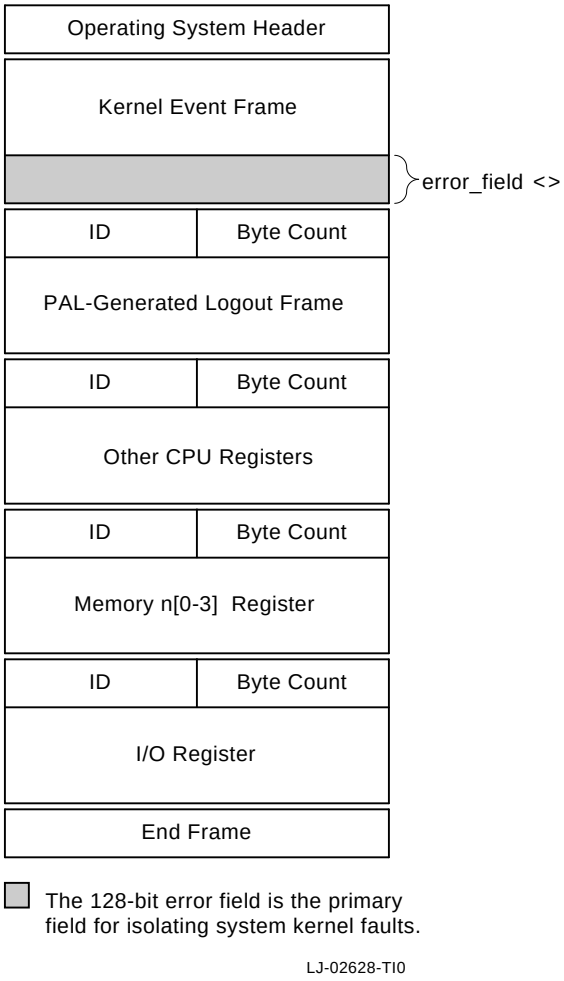
4.2 Error Logging and Event Log Entry Format

The OpenVMS AXP and DEC OSF/1 error handlers can generate several entry types. All error entries, with the exception of correctable memory errors, are logged immediately. Entries can be of variable length based on the number of registers within the entry.

Each entry consists of an operating system header, kernel event frame, several device frames, and an end frame. Most entries have a PAL-generated logout frame, and may contain registers for a second CPU, memory (0–3), and I/O.

Figure 4–1 shows the general error log format used by the ERF/UERF error formatters.

Figure 4–1 ERF/UERF Error Log Format



By examining the error field of the kernel event frame, you can isolate the failing system kernel FRU for system faults reported by the operating system. One or more bits are set in the low and high quadword of the error field as the result of the system error handling process. During the error handling process, errors are first handled by the appropriate PALcode error routine and then by the associated operating system error handler.

Section 4.4 describes how to interpret the error field to isolate to the FRU that is the source of the failure. The next generation of fault management and error notification tools will key off of these error field bits.

Note

For error logs indicating problems with a storage device, use the `test` command to verify the problem with the specifed device.

4.3 Event Record Translation

The ERF and UERF error formatters translate the entry into the format described in Section 4.2. OpenVMS AXP uses the ERF error formatter; DEC OSF/1 uses the UERF error formatter.

Both ERF and UERF provide bit-to-text translation for the kernel event frame.

Section 4.3.1 summarizes the commands used to translate the error log information for the OpenVMS AXP operating system. Section 4.3.2 summarizes the commands used to translate the error log for the DEC OSF/1 operating system.

4.3.1 OpenVMS AXP Translation

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

Format:

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

Example:

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

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

ERF bit-to-text translation highlights all error flags that are set, and other significant state. These are displayed in capital letters in the third column of the error log (see ❸ in Example 4–1). Otherwise, nothing is shown in the translation column.

Section 4.4.9 provides a sample ERF-generated error log.

4.3.2 DEC OSF/1 Translation

Error log information is written to /var/adm/binary.errlog. Use the following command to save the error log information by copying it to another file:

```
$ cp /var/adm/binary.errlog /tmp/errors_upto_today
```

To clear the error log file, use the following command:

```
$ cp /dev/null /var/adm/binary.errlog
```

To produce a bit-to-text translation of the error log file, use the following command:

```
$ uerf -f /tmp/errors_upto_today -R
```

To view all all error logs in reverse chronological order, use the following command:

```
$ uerf -R
```

For filtering of error logs, see the reference page for UERF on the system you are currently using:

```
$ man uerf
```

Section 4.4.10 provides a sample UERF-generated error log.

4.4 Interpreting System Faults Using ERF and UERF

Use the following steps to determine the failing FRU when a system error is reported via an error log.

1. Examine the error field of the kernel event frame.

   If a system error has been reported, one or more bits may be set for the low and high quadword and their corresponding bit-to-text definition will be listed.

2. Using Table 4–2, find the entry that matches the set bit and bit-to-text to determine the most probable source of the fault listed in the third column.

3. If the table entry lists a note number along with the most probable failing module, refer to that note following Table 4–2.

There are eight possible notes, Note 1–Note 8. Each note provides a synopsis of the problem and additional information to consider for analysis.

Section 4.4.9 provides a sample ERF-generated error log. Section 4.4.10 provides a sample UERF-generated error log.

Table 4–2 Error Field Bit Definitions for Error Log Interpretation

Error Field Bits	U/ERF Bit-to-Text Definition	Module/Notes
Quadword 0, CPU0-Detected		
W0-Byte-0, CPU Machine Check Related Errors		
<0> C3_0_CA_NOACK	CPU_0 Bus Command No-Ack	CPU_0, Note 1
<1> C3_0_WD_NOACK	CPU_0 Bus Write Date No-Ack	CPU_0, Note 2
<2> C3_0_RD_PAR	CPU_0 Bus Read Parity Error	CPU_0, Note 3
<3> EV_0_C_UNCORR	CPU_0 Cache Uncorrectable	CPU_0, Note 4
<4> EV_0_TC_PAR	CPU_0 Cache Tag Control Parity Error	CPU_0
<5> EV_0_T_PAR	CPU_0 Cache Tag Parity Error	CPU_0
<6> C3_0_EV	CPU_0 EV to system bus interface data error	CPU_0
W0-Byte-1, CPU Interrupt and Machine Check Related Errors		
<0> C3_0_C_UNCORR	CPU_0 Cache Uncorrectable (system bus interface detected)	CPU_0, Note 4
<1> C3_0_TC_PAR	CPU_0 Cache tag Control Parity Error	CPU_0
<2> C3_0_T_PAR	CPU_0 Cache tag Parity Error	CPU_0
<3> C3_0_C_CORR	CPU_0 Cache Correctable (system bus interface detected)	CPU_0
<4> EV_0_C_CORR	CPU_0 Cache Correctable (21064 detected)	CPU_0

(continued on next page)

Table 4–2 (Cont.) Error Field Bit Definitions for Error Log Interpretation

Error Field Bits	U/ERF Bit-to-Text Definition	Module/Notes
Quadword 1, CPU1-Detected		
W1-Byte-0, CPU Machine Check Related Errors		
<0> C3_1_CA_NOACK	CPU_1 Bus Command No-Ack	CPU_1, Note 1
<1> C3_1_WD_NOACK	CPU_1 Bus Write Date No-Ack	CPU_1, Note 2
<2> C3_1_RD_PAR	CPU_1 Bus Read Parity Error	CPU_1, Note 3
<3> EV_1_C_UNCORR	CPU_1 Cache Uncorrectable (CPU detected)	CPU_1, Note 4
<4> EV_1_TC_PAR	CPU_1 Cache tag Control Parity Error	CPU_1
<5> EV_1_T_PAR	CPU_1 Cache tag Parity Error	CPU_1
<6> C3_1_EV	CPU_1 CPU to system bus interface data error	CPU_1
W1-Byte-1, CPU Interrupt and Machine Check Related Errors		
<0> C3_1_C_UNCORR	CPU_1 Cache Uncorrectable (system bus interface detected)	CPU_1, Note 4
<1> C3_1_TC_PAR	CPU_1 Cache tag Control Parity Error	CPU_1
<2> C3_1_T_PAR	CPU_1 Cache tag Parity Error	CPU_1
<3> C3_1_C_CORR	CPU_1 Cache Correctable (system bus interface detected)	CPU_1
<4> EV_1_C_CORR	CPU_1 Cache Correctable (CPU detected)	CPU_1
Miscellaneous Flags		
W2-Byte-0, CPU-Specific (in context of CPU that is reporting the error)		
<0> EV_SYN_1F	CPU reported syndrome 0x1f	Note 4
<1> C3_SYN_1F	System bus interface reported syndrome 0x1f	Note 4
<2> DT_PAR	Duplicate Tag Store Parity Error	This CPU
<3> EV_HARD_ERROR	CPU cycle aborted with HARD ERROR	

(continued on next page)

Table 4–2 (Cont.) Error Field Bit Definitions for Error Log Interpretation

Error Field Bits	U/ERF Bit-to-Text Definition	Module/Notes
W2-Byte-1, Event Correlation Flags		
<0> C3_MEM_R_ERROR	CPU error caused by memory	Note 4
<1> IO_MEM_R_ERROR	I/O error caused by memory	
<2> C3_OCPU_ADD_ MATCH	CPU error caused by other CPU	Note 4
<3> MIXED_ERRORS	Mixed errors (no correlation)	
I/O As Commander (bus errors that the I/O module can detect while the I/O module is commander)		
W3-Byte-0, External Cause		
<0> IO_CA_NOACK	I/O detected Bus Command/Add No-Ack	I/O, Note 1
<1> IO_WD_NOACK	I/O detected Bus Write Date No-Ack	I/O, Note 2
<2> IO_RD_PAR	I/O detected Bus Read Parity Error	I/O, Note 3
<3> IO_CB_UNCORR	Data delivered to I/O is corrupted	Note 5
W3-Byte-1, Internal Cause		
<0> IO_LB_DMA_PAR	I/O - L-Bus DMA Parity Error	I/O
<1> IO_FB_DMA_PAR	I/O - F-Bus DMA Parity Error	I/O, Note 6
<2> IO_FB_MB_PAR	I/O - F-Bus Mailbox Access Par Error	I/O, Note 7
<3> IO_BUSSYNC	I/O - Chip-SysBus Sync Error	I/O
<4> IO_SCSTALL	I/O - Chip Sync Error	I/O

(continued on next page)

Table 4–2 (Cont.) Error Field Bit Definitions for Error Log Interpretation

Error Field Bits	U/ERF Bit-to-Text Definition	Module/Notes
Quadword 1 Responder Errors		
W0-Byte-0, Command/Address Parity Error Detected		
<0> C3_0_CA_PAR	CPU_0 Bus Command/Add Parity Error	CPU_0, Note 1
<1> C3_1_CA_PAR	CPU_1 Bus Command/Add Parity Error	CPU_1, Note 1
<2> MEM0_CA_PAR	MEM_0 Bus Command/Add Parity Error	MEM_0, Note 1
<3> MEM1_CA_PAR	MEM_1 Bus Command/Add Parity Error	MEM_1, Note 1
<4> MEM2_CA_PAR	MEM_2 Bus Command/Add Parity Error	MEM_2, Note 1
<5> MEM3_CA_PAR	MEM_3 Bus Command/Add Parity Error	MEM_3, Note 1
<6> IO_CA_PAR	I/O Bus Command/Add Parity Error	I/O, Note 1
W0-Byte-0, System Bus Interface Write Data Parity Errors		
<0> C3_0_WD_PAR	CPU_0 Bus Write Data Parity Error	CPU_0, Note 2
<1> C3_1_WD_PAR	CPU_1 Bus Write Data Parity Error	CPU_1, Note 2
<2> MEM0_WD_PAR	MEM_0 Bus Write Data Parity Error	MEM_0, Note 2
<3> MEM1_WD_PAR	MEM_1 Bus Write Data Parity Error	MEM_1, Note 2
<4> MEM2_WD_PAR	MEM_2 Bus Write Data Parity Error	MEM_2, Note 2
<5> MEM3_WD_PAR	MEM_3 Bus Write Data Parity Error	MEM_3
<6> IO_WD_PAR	I/O Bus Write Data Parity Error	I/O
W1-Byte-0, Memory Uncorrectable Errors		
<0> MEM0_UNCORR	MEM_0 Uncorrectable Error	MEM_0
<1> MEM1_UNCORR	MEM_1 Uncorrectable Error	MEM_1
<2> MEM2_UNCORR	MEM_2 Uncorrectable Error	MEM_2
<3> MEM3_UNCORR	MEM_3 Uncorrectable Error	MEM_3

(continued on next page)

Table 4–2 (Cont.) Error Field Bit Definitions for Error Log Interpretation

Error Field Bits	U/ERF Bit-to-Text Definition	Module/Notes
W1-Byte-1, Memory Correctable Errors		
<0> MEM0_CORR	MEM_0 Correctable Error	MEM_0, Note 8
<1> MEM1_CORR	MEM_1 Correctable Error	MEM_1, Note 8
<2> MEM2_CORR	MEM_2 Correctable Error	MEM_2, Note 8
<3> MEM3_CORR	MEM_3 Correctable Error	MEM_3, Note 8
W2-Byte-0, Sync Errors (the two gate arrays are not working together)		
<0> MEM0_SYNC_Error	MEM_0 Chip Sync Error	MEM_0
<1> MEM1_SYNC_Error	MEM_1 Chip Sync Error	MEM_1
<2> MEM2_SYNC_Error	MEM_2 Chip Sync Error	MEM_2
<3> MEM3_SYNC_Error	MEM_3 Chip Sync Error	MEM_3

4.4.1 Note 1: System Bus Address Cycle Failures

Synopsis:

System bus address cycle failures can be reported by the bus commander, responders, or both:

- By commander: _CA_NOACK—Bus Command Address No-Ack

  Commander did not receive an acknowledgment command/address. Probable causes are:

  – A programming error, software fault (addressed nonexistent address)

  – A bus buffer failure on the bus commander

- By responders: _CA_PAR—Bus Command/Address Parity Error

  Responder detected a parity error during the Command/Address cycle. The bus was corrupted by commander module (I/O or CPU), backplane, or responder module (I/O, memory, or CPU).

Analysis:

Note

All bus nodes check command/address parity during the command/address cycle.

- _CA_NOACK errors without respective command/address parity errors are most likely caused by problems in the bus commander, such as programming errors, address generation, and the like. You should consider the context of the error; for example, a software fault may cause the system to crash each time you run a particular piece of software.
- _CA_NOACK errors with all responders reporting command/address parity errors are most likely caused by a bus commander failure or bus failure.
- _CA_PAR errors, without respective command/address NOACKs are most likely the result of a failing buffer within the device reporting the isolated CA_PAR error.

4.4.2 Note 2: System Bus Write-Data Cycle Failures

Synopsis:

System Bus Write Data failures can be reported by the bus commander, responders, or both.

- By commander: _WD_NOACK—Write-Data No-Ack

  Commander did not receive an acknowledgment to write-data cycle. A bus buffer failure on the bus commander is the probable cause.
- By responders: _WD_PAR—Write-Data Parity Error

  Responder detected a parity error during the write-data cycle. The bus was corrupted by commander module (I/O or CPU), backplane, or responder module (I/O, memory, or CPU).

Analysis:

Note

Only the addressed bus responder checks write-data parity.

- _WD_NOACK (write-data NOACK) errors without respective WD_PAR (write-data parity) errors are most likely caused by problems in the bus commander. However, there is a small probability that the responder could be at fault.

Examine the commander's command trap register to identify the respective responder.

- _WD_NOACK errors with the responder reporting _WD_PAR errors could indicate a failure with either device.
- _WD_PAR errors without respective _WD_NOACK would require two failures to occur:

  1. Bad data received by responder
  2. A valid response was received when one should not have been sent.

  The failing module could be either partner in the transfer.

4.4.3 Note 3: System Bus Read Parity Error

Synopsis:

System bus read-data failures are reported only by the bus commander.

- By commander: _RD_PAR error—Read-data parity error.

  The bus commander (device reporting _RD_PAR) detected a parity error on data received from the system bus.

Analysis:

Note

Only the bus commander checks write-data parity on bus reads.

- The failure could be caused by either the bus commander or responder. The failing data's address is captured in the commander's bus trap register.
- A system bus read parity error can result as a side effect of a command/address NOACK.

4.4.4 Note 4: Backup Cache Uncorrectable Error

Synopsis:

Data from the backup cache is either delivered to the DECchip 21064 microprocessor or the system bus interface chip is corrupted.

Analysis:

The failing module is the CPU reporting the failure, except:

- If EV_SYN_1F (“CPU reported syndrome 0x1f”) or C3_SYN_1F (“C3 reported syndrome 0x1f”) bits are set in the error field, known bad data was supplied to the CPU from another source (either memory or the other CPU).
  - If C3_MEM_R_ERROR (“CPU error caused by memory”) bit is set, examine MEMn_UNCORR (“MEM_n Uncorrectable Error”) or MEMn_ SYNC_Error (“MEM_n Chip Sync Error”) to identify which memory was the source of the error.
  - If C3_OCPU_ADD_MATCH (“CPU error caused by other CPU”) is set, the other CPU caused the error.
- If other error bits associated with the CPU reporting the error are also set, there is a probability that the fault is associated with this CPU module.

4.4.5 Note 5: Data Delivered to I/O Is Known Bad

Synopsis:

IO_CB_UNCORR—I/O module received data identified as bad from system bus.

Analysis:

Check to see if the following bits are set for the error field:

MEMn_UNCORR (“MEM_n Uncorrectable Error”)
MEMn_SYNC_Error (“MEM_n Chip Sync Error”)
CPUn_XXXXXX errors (“CPU_n xxx... error”)

4.4.6 Note 6: Futurebus+ DMA Parity Error

Synopsis:

Either an address or data parity error occurred on the Futurebus+ while a DMA data transfer was executing from a Futurebus+ option to memory (detected by the I/O module).

Analysis:

The failing module could be either the I/O module or one of the Futurebus+ options. There is no way to isolate to the failing Futurebus+ module from the error log.

4.4.7 Note 7: Futurebus+ Mailbox Access Parity Error

Synopsis:

A data parity error occurred during reading of data from a Futurebus+ option via a mailbox operation.

Analysis:

The failing module could be either the I/O module or one of the Futurebus+ options. There is no way to isolate to the failing Futurebus+ module from the error log.

4.4.8 Note 8: Multi-Event Analysis of Command/Address Parity, Write-Data Parity, or Read-Data Parity Errors

Analysis:

Because command/address, read-data, and write-data share the backplane and bus transverse, problems with these components can be seen as failures in any of these cycles. It may be possible to identify the failing module by examining several failure entries and drawing a conclusion as to the failing module.

- Are the parity errors always associated with the same responder?

  If so, the fault is most likely with the responder.

- Are the read-parity errors always associated with the same commander?

  If so, the fault is most likely with the commander.

- Is one module never reporting or associated with an error?

  If so, this module could be corrupting the bus.

4.4.9 Sample System Error Report (ERF)

Example 4–1 provides an abbreviated ERF-generated error log for a processor corrected machine check, SCB 630 (❶).

The low quadword of the error field, ERR FIELD LOW (❷), has one bit set. The corresponding bit-to-text translation (❸) is provided in the third column.

The high quadword of the error field register, ERR FIELD HIGH (❹), has no bits set.

Example 4–1 ERF-Generated Error Log Entry Indicating CPU Corrected Error

```
 V M S               SYSTEM ERROR REPORT         COMPILED 17-NOV-1992 10:54:57
                                                                  PAGE   1.

 ****************************** ENTRY      1. ******************************
 ERROR SEQUENCE 1.                              LOGGED ON:  CPU_TYPE 00000002
 DATE/TIME 21-SEP-1992 12:00:24.83                           SYS_TYPE 00000002
 SYSTEM UPTIME: 0 DAYS 00:10:04
 SCS NODE: DSSI3                                              VMS T1.0-FT4

 CACHE ERROR KN430
 CACHE ERROR

 KERNEL EVENT HEADER
       FRAME REVISION      0000
       SCB VECTOR          0630 ❶
       1ST MOST PRB FRU      00
                                    FIELD NOT VALID
       2ND MOST PRB FRU      00
                                    FIELD NOT VALID
       SEVERITY            0000
                                    FIELD NOT VALID
       CPU ID              0000
       ERROR COUNT         0001
       THRESHOLD           0000
       FAIL CODE           0000
       ERR FIELD LOW   00000000 00001000 ❷
                                    CPU_0 CACHE CORR. (CPU DETECTED) ❸
       ERR FIELD HIGH  00000000 00000000 ❹

 MACHINE CHECK FRAME

       RETRY/BYTE CNT  80000000 00000230
        .
        .
        .
 MEMORY ERROR FRAME

       MEMORY ERROR 1  00040002 00040001
                                    Sync Error Even
                                    EDC Corr Error Even
                                    Cmd ID Odd Array = 00(X)
        .
        .
        .
 OTHER CPU FRAME

       CPU #               0000
                                    CPU Number = 0.
        .
        .
        .
ANALYZE/ERROR/OUT=ERIK.TXT MEM_FRAME.ZPD
```

4.4.10 Sample System Error Report (UERF)

Example 4–2 provides an abbreviated UERF-generated error log for a processor machine check, SCB 670 (❶).

The low quadword of the error field register, ERROR FLAG1 (❷), has two bits set. The corresponding bit-to-text translations may not be provided for some versions of DEC OSF/1. The high quadword of the error field register, ERROR FLAG2 (❸), has no bits set.

Note

The following analysis of the error field is helpful in finding the corresponding bit-to-text translation in Table 4–2.

ERROR FLAG1 corresponds to quadword 0; ERROR FLAG2 corresponds to quadword 1.

The error field bits are arranged in four-character words (0–3, right to left); for example,

```
ERROR FLAG1    x|0000|0008|0000|0005
                  3    2    1    0
```

Example 4–2 UERF-Generated Error Log Entry Indicating CPU Error

```
       uerf version 4.2-011 (118)

******************************** ENTRY    1. ********************************

----- EVENT INFORMATION -----

EVENT CLASS                             ERROR EVENT
OS EVENT TYPE                 100.      CPU EXCEPTION
SEQUENCE NUMBER                 1.
OPERATING SYSTEM                        DEC OSF/1
OCCURRED/LOGGED ON                      Sun Jul  4 08:04:10 1976
OCCURRED ON SYSTEM                      forge
SYSTEM ID               x0002000F       CPU TYPE: DEC
                                        CPU SUBTYPE: KN430

----- HEADER FRAME -----
```

(continued on next page)

Example 4–2 (Cont.) UERF-Generated Error Log Entry Indicating CPU Error

```
FRAME REVISION               x0001
SCB VECTOR                   x0670 ❶
FRU 1                        x0000     FIELD NOT VALID
FRU 2                        x0000     FIELD NOT VALID
SEVERITY                     x0001     SEVERITY FATAL
CPU ID                       x0000
ERROR COUNT                  x0001
THRESHOLD FOR FAIL C         x0000     FIELD NOT VALID
FAIL CODE                    x0000
ERROR FLAG1 ❷   x0000000800000005
ERROR FLAG2 ❸   x0000000000000000

----- LEP MACHINE CHECK STACK FRAME -----

PROCESSOR OFFSET          x000001B0
SYSTEM OFFSET             x00000120
PALTEMP0             x0000000000000001
PALTEMP1             xFFFFFC0280000000
    .
    .
    .
----- COBRA CPU SPECIFIC STACK FRAME -----

BCC_CSR0             x00000000400001C1     ENB ALLOCATE
                                           ENB COR ERR INTERRUPT
    .
    .
    .
----- MEMORY FRAME -----

MEMORY MODULE ID          x00000003
    .
    .
    .
----- I/O FRAME -----

IOCSR                x00000E0000000E00
    .
    .
    .
----- UNKNOWN FRAME -----

FRAME ID                  x00000009
      .
      .
      .
0100:   00000000  00000000  00000000  00000000       *................*
```

5

Repairing the System

This chapter describes the removal and replacement procedures for DEC 4000 AXP systems.

- Section 5.1 gives general guidelines for FRU removal and replacement.
- Section 5.2 covers FRUs accessed at the front of the system.
- Section 5.3 covers FRUs accessed at the rear of the system.
- Section 5.4 describes the backplane removal and replacement.
- Section 5.5 describes the types of repair data that should accompany returned FRUs.

5.1 General Guidelines for FRU Removal and Replacement

Use the illustrations in this chapter as the primary source of FRU removal information. Text is provided for procedures or precautions that require additional clarification.

Unless otherwise specified, you can install an FRU by reversing the steps in the removal procedure.

Refer to the *DEC 4000 AXP Model 600 Illustrated Parts Breakdown: Mass Storage Device* (EK–MS430–IP) and *DEC 4000 AXP Model 600 Illustrated Parts Breakdown: Series Enclosure* (EK–EN430–IP) if you need a more detailed illustration.

Caution

Only qualified service personnel should remove or install FRUs.

Turn off the DC on/off switch and AC circuit breaker, then unplug the system before you remove or install FRUs.

Static electricity can damage integrated circuits. Always use a grounded wrist strap (29–26246) and grounded work surface when working with the internal parts of a computer system.

The cable guide screws do not contact the chassis and should not be used for static grounding.

Warning

The following warning symbols appear on the system enclosure. Please review their definitions.

Hazardous voltages are present within the front end unit (AC power supply). Do not access unless properly trained. Before you access this unit, remove AC power by pressing the AC circuit breaker to the Off (0) position, and unplug the power cord. Wait several minutes to ensure that stored charge is no longer present. Do not plug in the AC power cord unless the front end unit enclosure, including all covers and guards, is fully assembled.

Unless you remove AC power by pressing the AC circuit breaker to the Off (0) position, 48 V may be live in certain areas within this unit. If 48 V is present, high currents exist. If you are working in a high-current area and are using conductive tools or wearing conductive jewelry, you can incur severe burns.

Before you replace any Futurebus+ module, remove power by pressing the AC circuit breaker to the Off (0) position. High currents exist on the card cage modules and can cause severe burns if you do not remove power. Failure to remove power can cause damage to the Futurebus+ modules, as well.

The BA640 enclosure does not support warm swap of Futurebus+ modules. You can use Futurebus+ modules that have a warm swap feature within the BA640 enclosure, but their warm swap feature will be inoperative.

Do not access while fans are moving. Press the AC circuit breaker to the Off (0) position to remove power and ensure that fans cannot become energized unexpectedly.

5.2 Front FRUs

The following sections contain the part numbers of the FRUs accessed at the front of the system. Text is provided for those procedures or precautions that require additional clarification.

Refer to Figure 5–2 for the location of front FRUs.

5.2.1 Operator Control Panel

Part Number	Name
70–28749–02	Bezel assembly OCP with PCB (operator control panel)

5.2.2 Vterm Module

Part Number	Name
54–21159–01	Vterm module (with soldered 10-conductor cable to OCP)

Removal and Replacement Tips

The Vterm module is located behind the OCP.

5.2.3 Fixed-Media Storage

Refer to Figures 5–3 through 5–7 for removal and replacement information. For more detailed cabling illustrations refer to the *DEC 4000 AXP Model 600 Illustrated Parts Breakdown: Mass Storage Device* (EK–MS430–IP).

5.2.3.1 3.5-Inch Fast-SCSI Disk Drives (RZ26, RZ27, RZ35)

Refer to Figure 5–3 and Figures 5–5 and 5–6.

Part Number	Name
BA6ZB–MY	Storage tray for up to four 3.5-inch fast SCSI disk drives
17–03572–01	Cable assembly, 50-conductor
17–03428–02	Cable, 12-conductor (storage devices to front panel)
17–03155–01	Flex circuit (local disk converter module to storage interface module)
17–03057–01	Harness assembly, 2-conductor (local disk converter module to storage interface module)
17-03080–02	Harness assembly, 4-conductor (local disk converter module to storage devices)
54–20868–01	Module, local disk converter

Part Number	Name
54–21135–01	Module, hard disk interface card
54–21191–01	RF35/RZ35 remote front panel
54–21835–01	Termination board, SCSI
RZXX–MY	3.5-inch drive with tray-specific cable

Removal and Replacement Tips

When adding or replacing 3.5-inch SCSI disk drives you must remove the drive's three resistor packs and two terminator power jumpers (Figure 5–5) before installing the drive to its storage tray. Failure to do so will result in problems with the SCSI bus.

Refer to Figure 5–6 to determine the proper placement of drives within the storage tray. The position of the drive corresponds to the bus node ID plugs as shown.

5.2.3.2 3.5-Inch SCSI Disk Drives

Refer to Figures 5–4, 5–5, and 5–6.

Part Number	Name
BA6ZE–MY	Storage tray for up to four 3.5-inch SCSI disk drives
70–28753–01	Cable assembly, includes 50-conductor cable 17–03074–01
17–03428–02	Cable, 12-conductor (storage devices to front panel)
17–03155–01	Flex circuit (local disk converter module to storage interface module)
17–03057–01	Harness assembly, 2-conductor (local disk converter module to storage interface module)
17-03080–02	Harness assembly, 4-conductor (local disk converter module to storage devices)
54–20868–01	Module, local disk converter
54–21191–01	RF35/RZ35 remote front panel
54–21135–01	Module, hard disk interface card
12–30552–01	Terminator, SCSI (H8574–A)
RZXX–MY	3.5-inch drive with tray-specific cable

Removal and Replacement Tips

When adding or replacing 3.5-inch SCSI disk drives, you must remove the drive's three resistor packs and two terminator power jumpers (Figure 5–5) before

installing the drive to its storage tray. Failure to do so will result in problems with the SCSI bus.

Refer to Figure 5–6 to determine the proper placement of drives within the storage tray. The position of the drive corresponds to the bus node ID plugs as shown.

5.2.3.3 5.25-Inch SCSI Disk Drive

Refer to Figure 5–7.

Part Number	Name
BA6ZE–MX	Storage tray for 5.25-inch SCSI disk drive
70–28753–02	Cable assembly, includes 50-conductor cable 17–03075–01
17–03155–01	Flex circuit (local disk converter module to storage interface module)
17–03057–01	Harness assembly, 2-conductor (local disk converter module to storage interface module)
17–03437–01	Harness assembly, 6-conductor (storage device to ID panel)
17–01329–02	Harness assembly, 4-conductor (local disk converter module to storage device)
54–20868–01	Module, local disk converter
54–21135–01	Module, hard disk interface card
54–20898–01	SCSI ID panel
12–30552–01	Terminator, SCSI
RZXX–MX	5.25-inch drive with tray-specific cable

5.2.3.4 SCSI Storageless Tray Assembly

Part Number	Name
70–29491–02	Storageless tray assembly, SCSI
54–21135–02	Fixed storage interface card
17–03075–01	Cable assembly, 50-conductor, interface card to bulkhead
12–30552–01	Terminator, SCSI

5.2.3.5 3.5-Inch DSSI Disk Drive

Refer to Figures 5–4 and 5–6.

Part Number	Name
BA6FE–MY	Storage tray for up to four 3.5-inch DSSI disk ISEs
70–28752–02	Cable assembly (includes 17–03408–01 cable, 50-conductor)
17–03057–01	Harness assembly, 2-conductor (local disk converter module to storage interface card)
17–03401–01	Harness assembly, 4-conductor (local disk converter module to storage device)
17–03428–02	Harness assembly, 12-conductor (storage device to front panel)
17–03155–01	Flex circuit (local disk converter module to storage interface module)
54–20868–01	Module, local disk converter
54–21135–01	Module, hard-disk interface card
54–21191–01	RF35/RZ35 remote front panel
12–29258–01	Terminator, DSSI
RFXX–MY	3.5-inch drive with tray-specific cable

Removal and Replacement Tips

Refer to Figure 5–6 to determine the proper placement of drives within the storage tray. The position of the drive corresponds to the bus node ID plugs as shown.

5.2.3.6 5.25-Inch DSSI Disk Drive

Refer to Figure 5–7.

Part Number	Name
BA6FE–MX	Storage tray for one 5.25-inch DSSI disk ISE
70–28752–01	Cable assembly (includes 17–03478–01 cable, 50-conductor)
17-03554–01	Cable, 10-conductor (ISE to DSSI remote front panel)
17–03057–01	Harness assembly, 2-conductor (local disk converter module to storage interface card)
17–03058–01	Harness assembly, 5-conductor (local disk converter module to storage device)

Part Number	Name
17–03155–01	Flex circuit (local disk converter module to storage interface card)
54–20868–01	Module, local disk converter
54–21135–01	Module, hard-disk interface card
54–20896–02	DSSI remote front panel
12–29258–01	Terminator, DSSI
RFXX–MX	5.25-inch drive with tray-specific cable

5.2.3.7 DSSI Storageless Tray Assembly

Part Number	Name
70–29491–01	Storageless tray assembly, DSSI
54–21135–01	Fixed-storage interface card
17–03078–01	Cable assembly, 50-conductor, interface card to bulkhead
12–29258–01	Terminator, DSSI

5.2.4 Removable-Media Storage (Tape and Compact Disc)

For information on removal and replacement of removable-media drives, refer to the *DEC 4000 AXP Model 600 Series Options Guide* (EK–KN430–OG).

5.2.4.1 SCSI Bulkhead Connector

Part Number	Name
70–29427–01	Cable/bracket assembly with 17–03182–01 cable

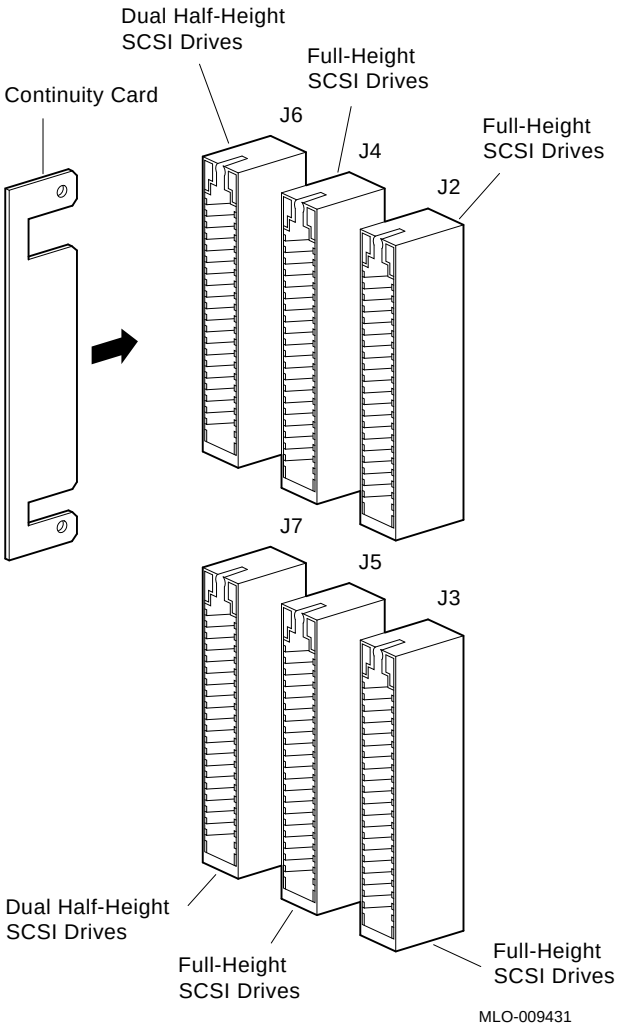
5.2.4.2 SCSI Continuity Card

Part Number	Name
54–21157–01	SCSI continuity card (for Bus E continuity)

Removal and Replacement Tips

Connectors J6 (upper left) and J7 (lower left) of the removable-media storage compartment require either a storage device (half-height) or SCSI continuity card. If a half-height device is installed, store the SCSI continuity card in connectors J4 or J5 (Figure 5–1).

Figure 5–1 SCSI Continuity Card Placement



5.2.5 Fans

Two fans (fan number 3 and 4) are accessed at the front of the system.

Part Number	Name
12–36202–01	Fan
17–03111–01	Fan power harness

Figure 5–2 Front FRUs

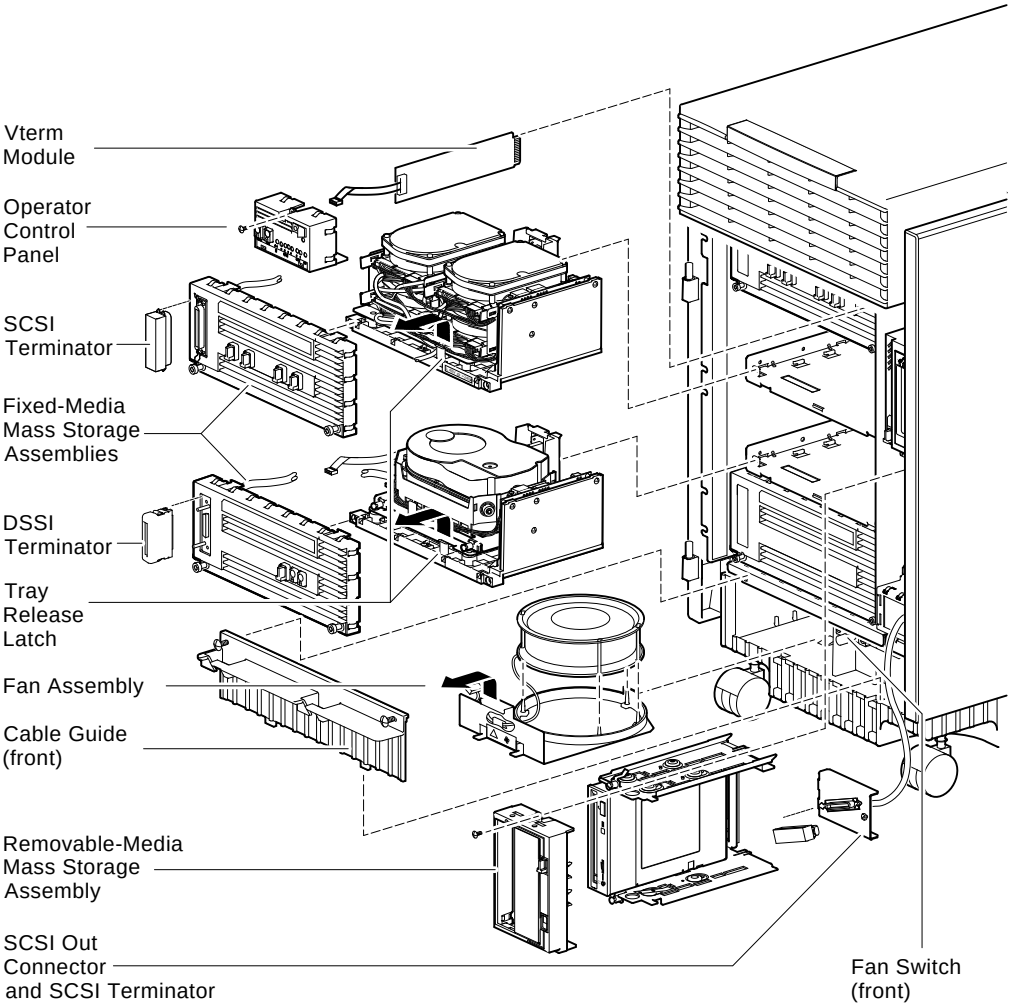


Figure 5–3 Storage Compartment with Four 3.5-inch Fast-SCSI Drives (RZ26, RZ27, RZ35)

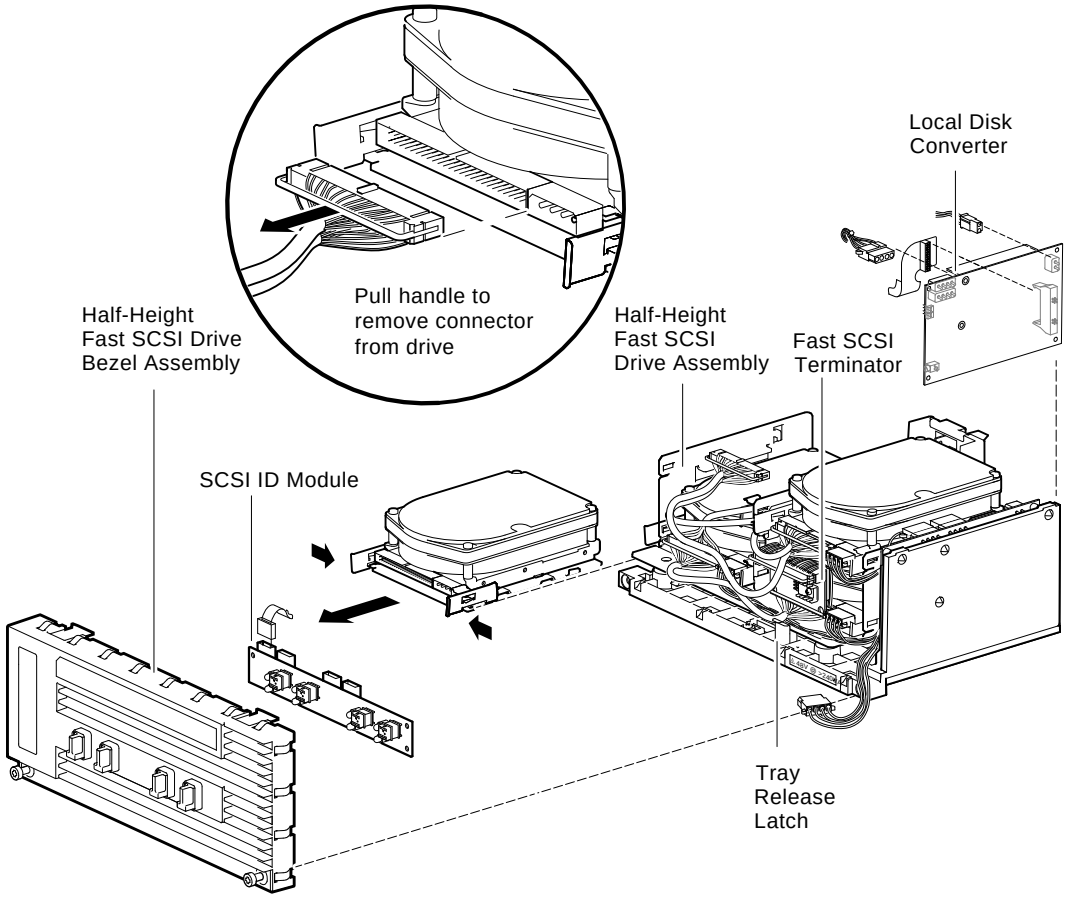


Figure 5–4 Storage Compartment with Four 3.5-inch SCSI/DSSI Drives

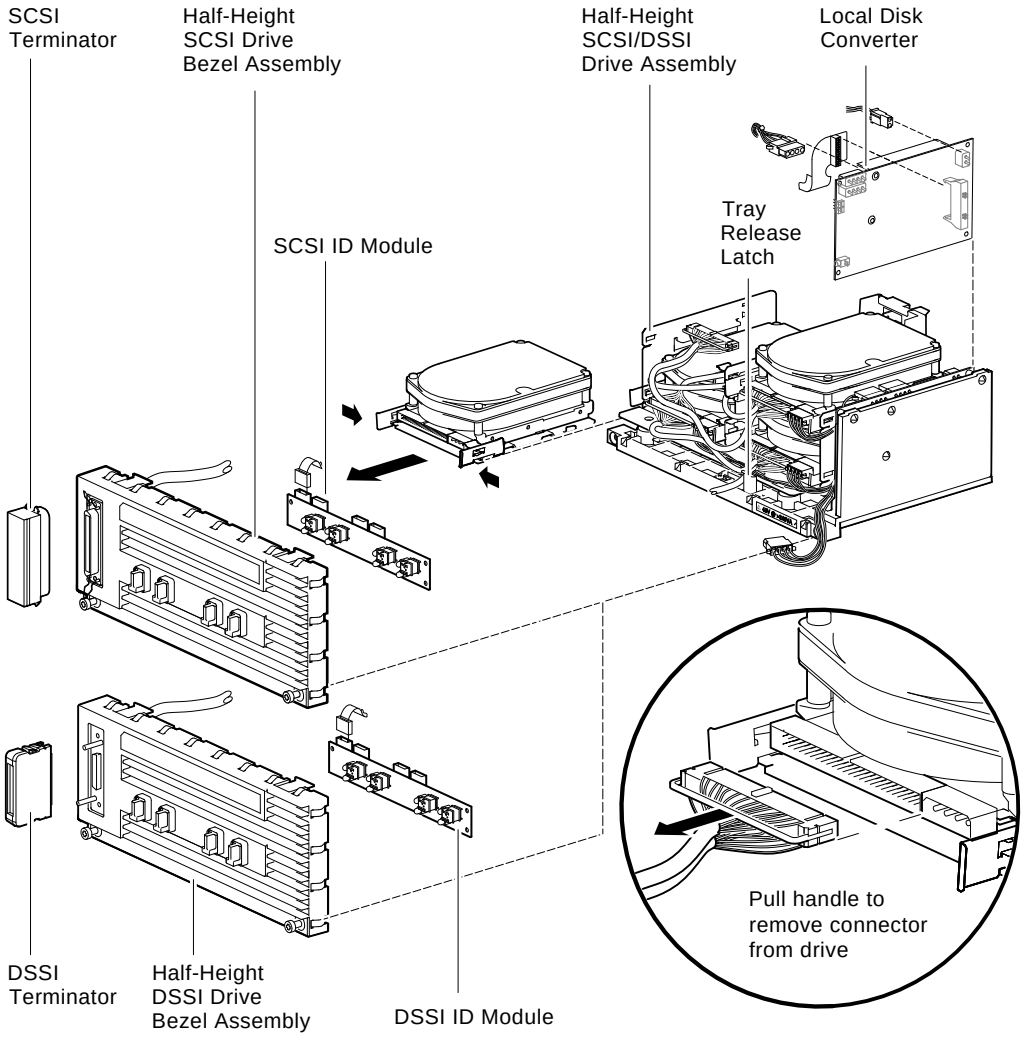
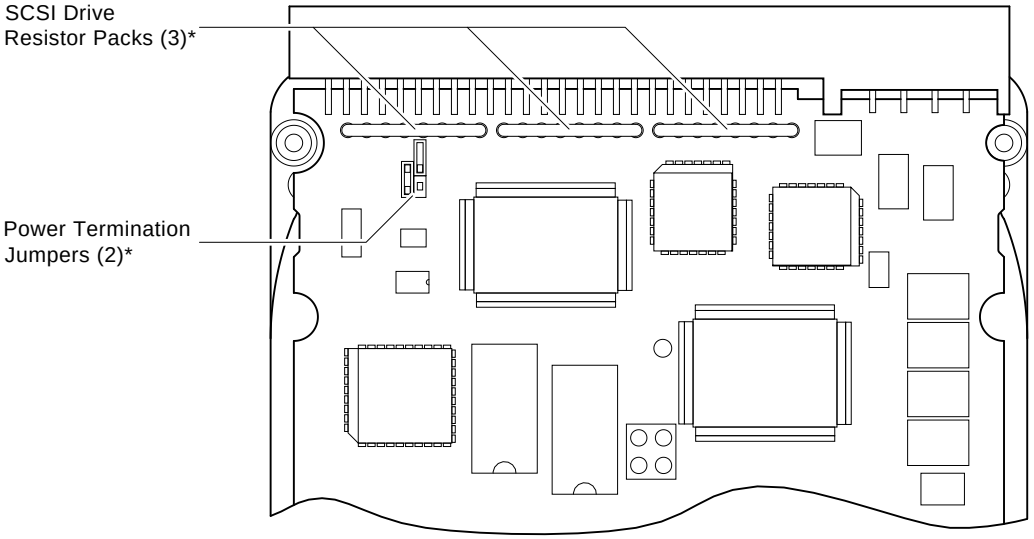


Figure 5–5 3.5-Inch SCSI Drive Resistor Packs and Power Termination Jumpers



* Must be removed before drive is installed to storage tray.

LJ-02268-TI0

Figure 5–6 Position of Drives in Relation to Bus Node ID Numbers

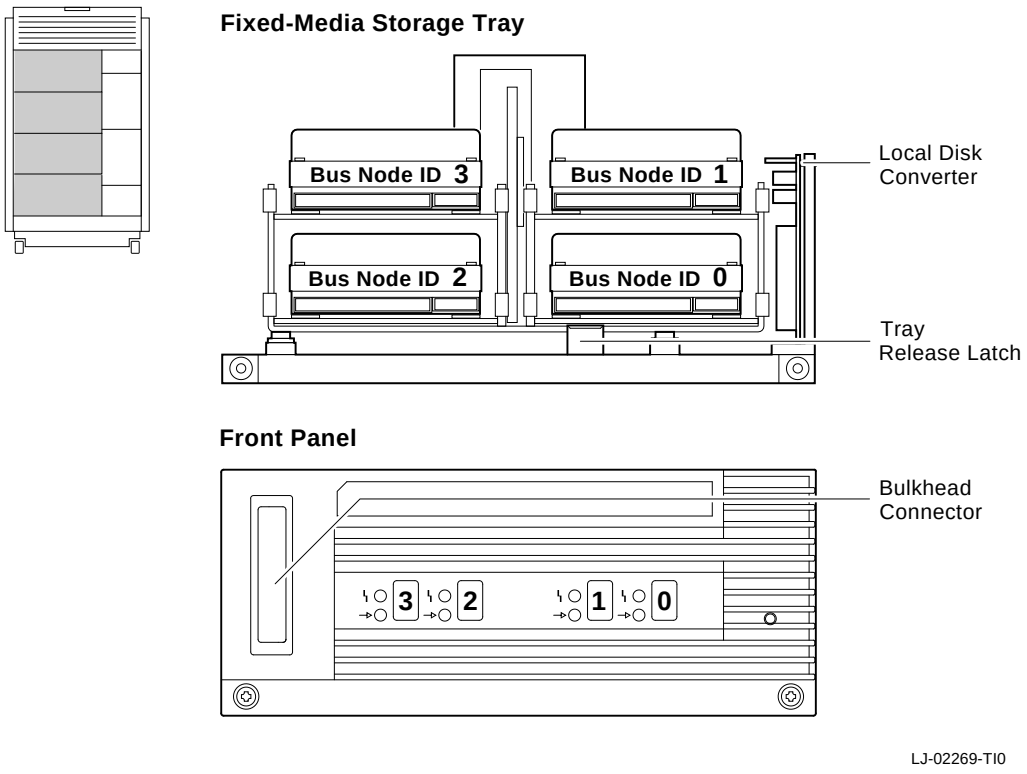
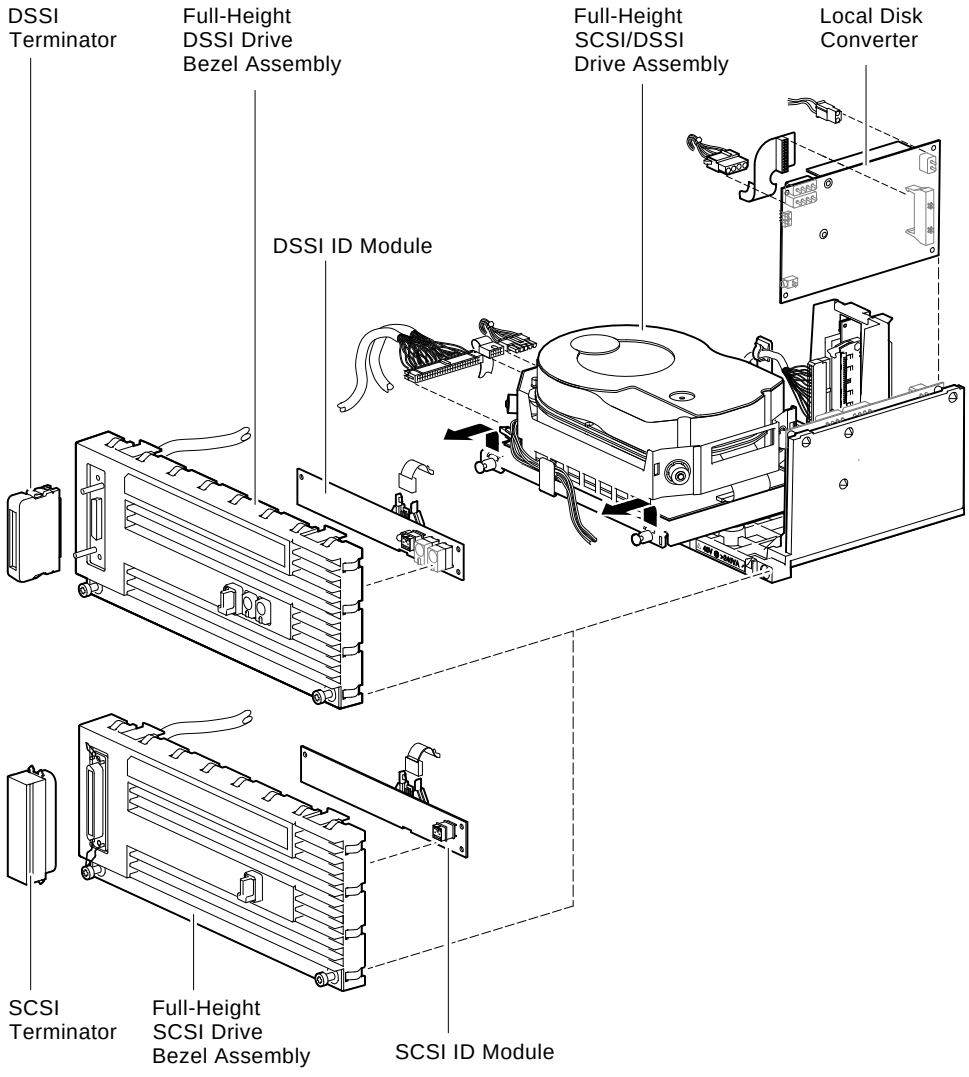


Figure 5–7 Storage Compartment with One 5.25-inch SCSI/DSSI Drive



5.3 Rear FRUs

The following sections contain the part numbers of the FRUs accessed at the rear of the system. Text is provided for additional procedures or precautions.

Refer to Figure 5–8 for the location of rear FRUs.

5.3.1 Modules (CPU, Memory, I/O, Futurebus+)

Part Number	Name
B2001–AA	KN430 processor module
B2002–BA	MS430–BA 32-MB memory module
B2002–CA	MS430–CA 64-MB memory module
B2002–DA	MS430–DA 128-MB memory module
B2101–AA	KFA40 I/O module

Removal and Replacement Tips

Note

The two small Phillips screws on each module are used to seat the modules. Loosen these screws before you remove the modules.

To replace the I/O module:

1. Record the customer's nonvolatile environment variable settings using the table in Appendix C. The `show` command lists all the environment variables.
2. Record the version of the console program. The `show config` command displays the console version.
3. Remove the I/O module and move the two socketed Ethernet address ROMs (labeled "Enet Adrs") to the new I/O module. Refer to Figure 5–9 to locate the Ethernet address ROMs.
4. Install the new I/O module and power up the system. If the system passes power-up tests, note the version of the console program. If the console version of the new I/O module is less than that of the module you removed, update the firmware using the CD–ROM shipped to the customer.
5. Complete acceptance testing using the `test` command.
6. Set the nonvolatile environment variables to the customer's original settings. Use the `set` command as shown in the examples below:

```
>>> set bootdef_dev eaz0
>>> set boot_osflags 0,1
>>>
```

5.3.2 Ethernet Fuses

Ethernet fuses are located on the I/O module. Refer to Figure 5–9 for the specific fuse location.

Part Number	Name
12–09159–00	0.5 A ThinWire Ethernet fuse (F1, F3)
12–10929–08	1.5 A thickwire Ethernet fuse (F2, F4)

5.3.3 Power Supply

Part Number	Name
H7884–AA	FEU, front end unit (20 A, replaces H7853–AA)
H7853–AA	FEU, front end unit (15 A) (early systems)
H7851–AA	PSC, power system controller
H7885–AA	DC5, DC–DC converter (150 A, 5 V, replaces H7179)
H7179–AA	DC5, DC–DC converter (90 A, 5 V) (early systems)
H7178–AA	DC3, DC–DC converter (3.3 V)
17–03342–01	Fan switch harness, 4-conductor with fan switch

5.3.4 Fans

Two fans (fan number 1 and 2) are accessed at the rear of the system.

Part Number	Name
12–36202–01	Fan
17–03111–01	Fan power harness

Figure 5–8 Rear FRUs

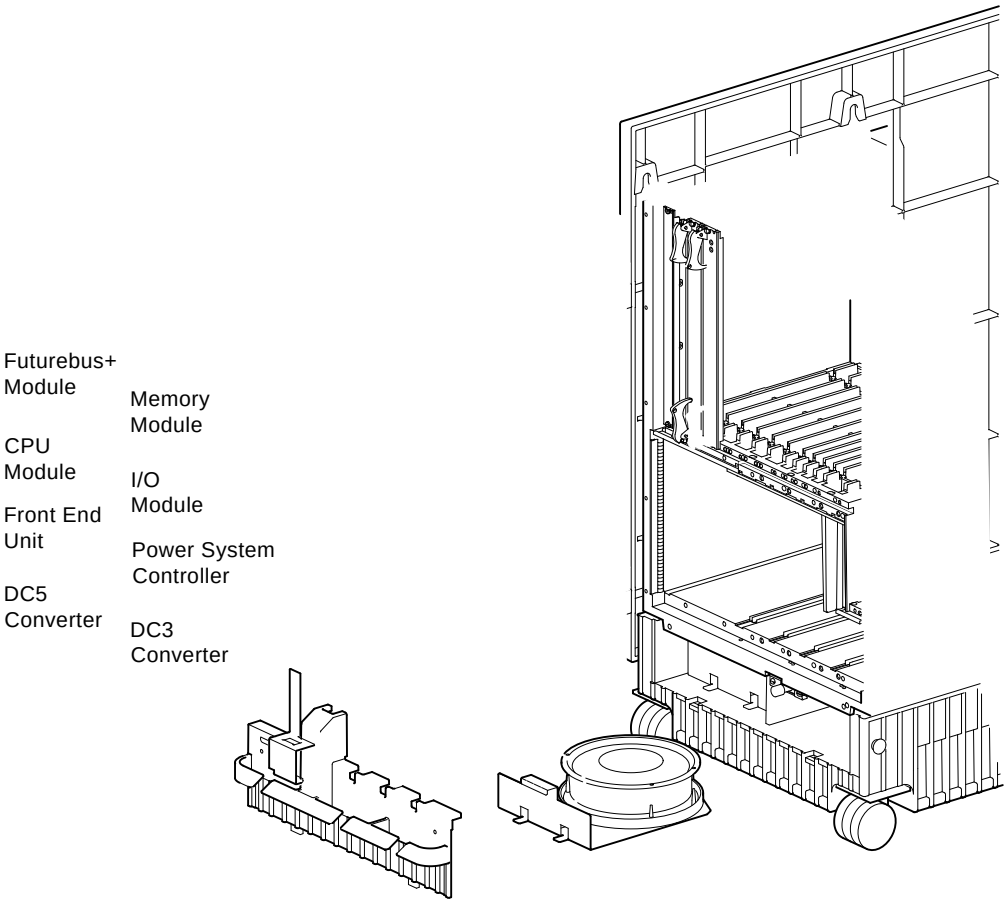
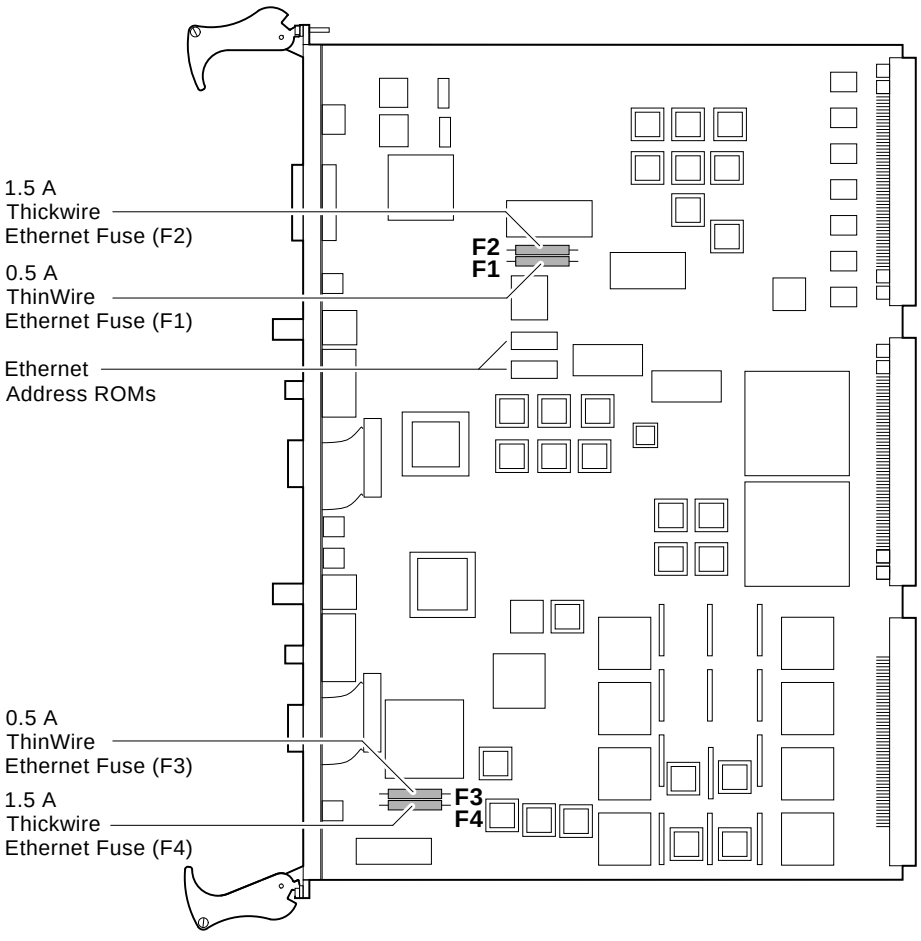


Figure 5–9 Ethernet Fuses and Ethernet Address ROMs



5.4 Backplane

Refer to Figures 5–10 and 5–11.

Part Number	Name
70–28747–01	Backplane assembly
17–03340–01	Cable assembly, 100-conductor backplane-to-backplane (2)
17–03341–01	Cable assembly, 40-conductor, backplane-to-backplane

Removal and Replacement Tips

To remove the backplane:

1. Unseat all modules (CPU, memory, I/O, and power supply modules) from the rear backplane.
2. Unseat and remove the Vterm module and all storage devices from the storage backplane.
3. Remove SCSI out connector and disconnect its cable from the storage backplane.
4. Remove outer shell (Figure 5–10).
5. Remove screws (Figure 5–11) and with the aid of an assistant, slide the front chassis forward enough to remove the backplane.

   Before removing the backplane, inspect the backplane cable assemblies. If the cables are damaged or improperly connected, replace the cables and not the backplane.

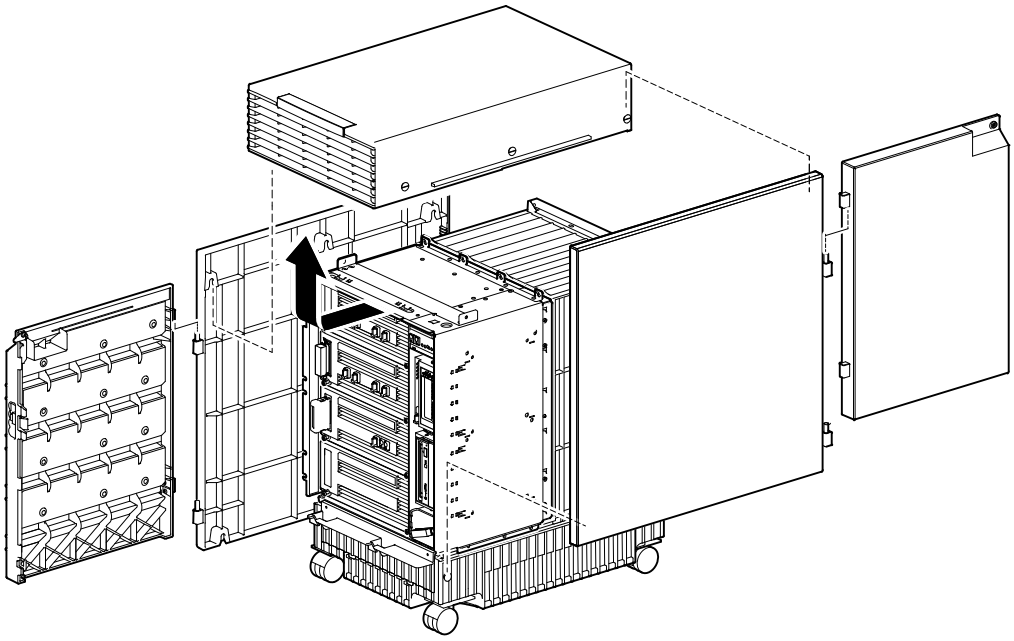
Warning

Lifting the front chassis requires two people.

To replace the backplane:

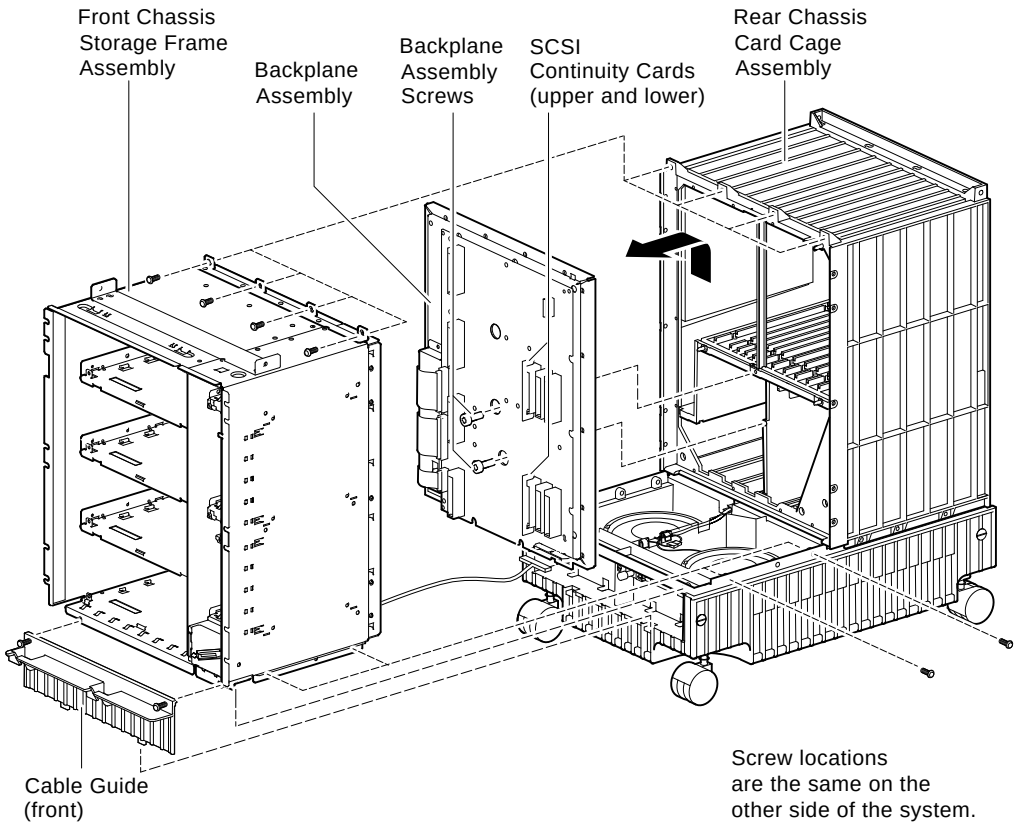
1. Secure the backplane with the two screws at the center.
2. Make sure the backplane is properly aligned by securing the front chassis to the rear chassis using the four screws at the top.
3. Replace remaining screws.

Figure 5–10 Removing Shell



LJ-01677-TI0

Figure 5–11 Removing Backplane



5.5 Repair Data for Returning FRUs

When you send back an FRU for repair, staple the error log to the fault tag or include as much of the error log information as possible.

- If one or more error flags are set in a particular entry, record the mnemonics of the registers, the hex data, and error flag translations on the repair tag.
- If an error address is valid, include the mnemonic, hex data, and translation on the repair tag as well.
- For memory and cache errors, describe the error and include corrected-bit/bit-in-error information, along with the register mnemonic and hex data.

6

System Configuration and Setup

This chapter provides a functional description of the system components, as well as system configuration and setup information.

- Section 6.1 provides a description of the major components and subsystems that make up the DEC 4000 system.
- Section 6.2 describes how to examine the system configuration using console commands.
- Section 6.3 describes how to set and examine environment variables.
- Section 6.4 describes how to set and examine DSSI parameters.
- Section 6.5 describes how to set console line baud rates.

6.1 Functional Description

The DEC 4000 AXP system is a department-level system that uses the custom VLSI CPU chip (DECchip 21064 microprocessor) based on the Alpha APX RISC architecture. The system is housed in a BA640 enclosure and includes the following components:

- Card cage that holds:
  - Up to 2 CPU modules
  - One I/O module
  - Up to 4 memory modules
  - Up to 6 Futurebus+ modules

- Four fixed-media storage compartments (each can hold up to 4 half-height drives or 1 full-height drive).
- A removable-media storage compartment (can hold 2 full-height or up to 4 half-height devices)
- Four fans
- Backplane assembly (includes system backplane: serial control bus, Futurebus+, and power bus; storage backplane: fixed-media and removable-media)
- Power subsystem
- Operator control panel

Figure 6–1 provides a block diagram of the system components. The major system components are:

- System bus (CPUs, memory, and I/O module)
- Serial control bus
- Futurebus+ and associated options

Figure 6–2 provides a diagram of the system backplane.

Figure 6–1 System Block Diagram

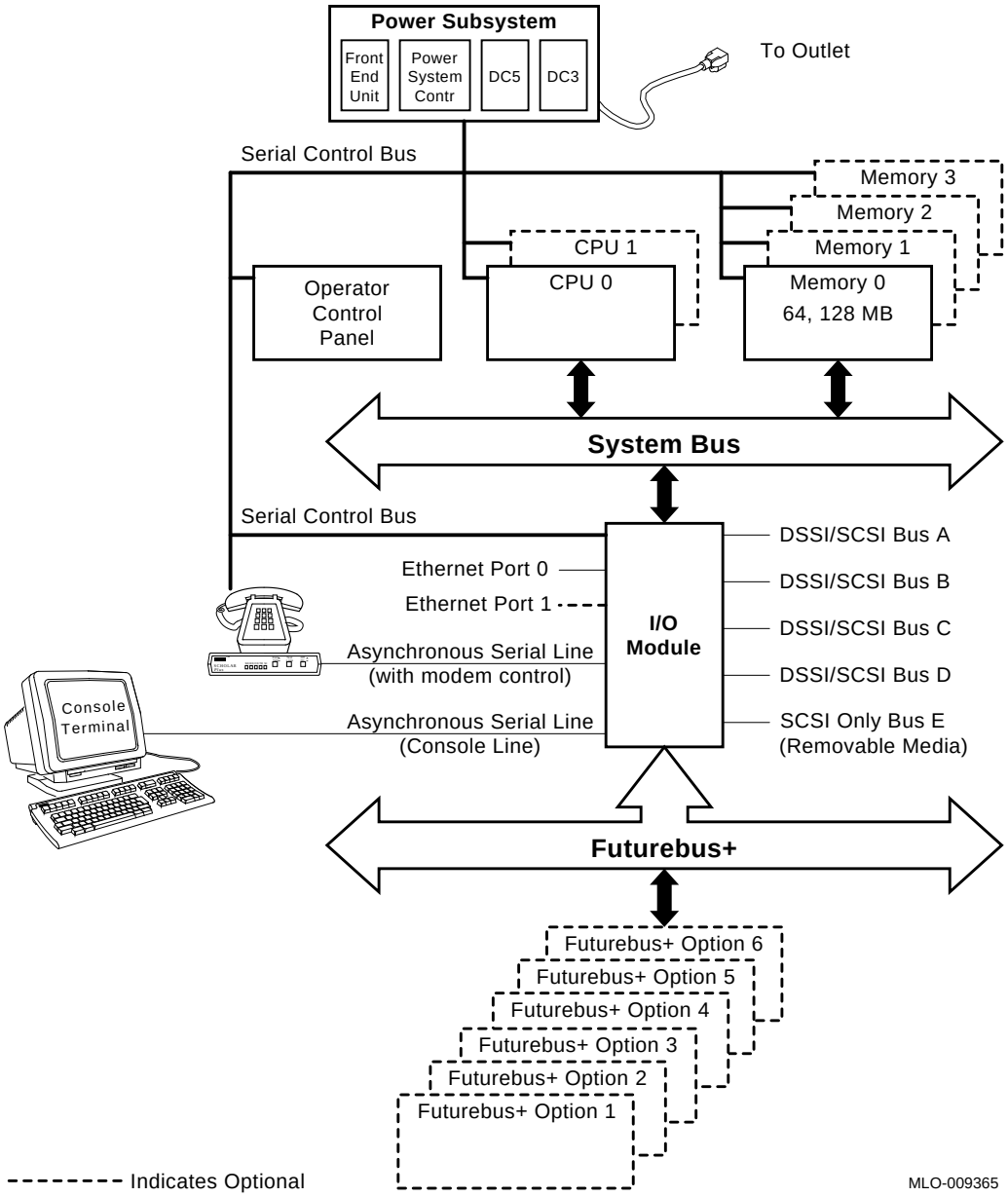
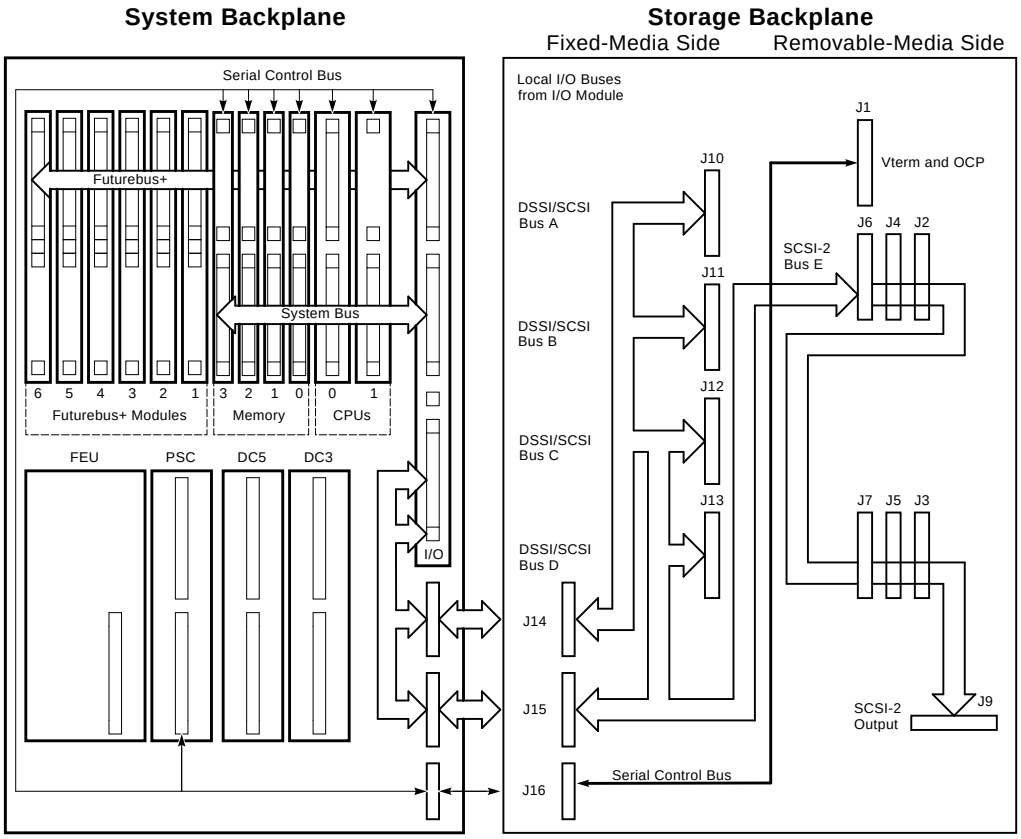


Figure 6–2 System Backplane



LJ-02062-TI0

Figures 6–3 and 6–4 show the front and rear of the BA640 enclosure.

Figure 6–3 BA640 Enclosure (Front)

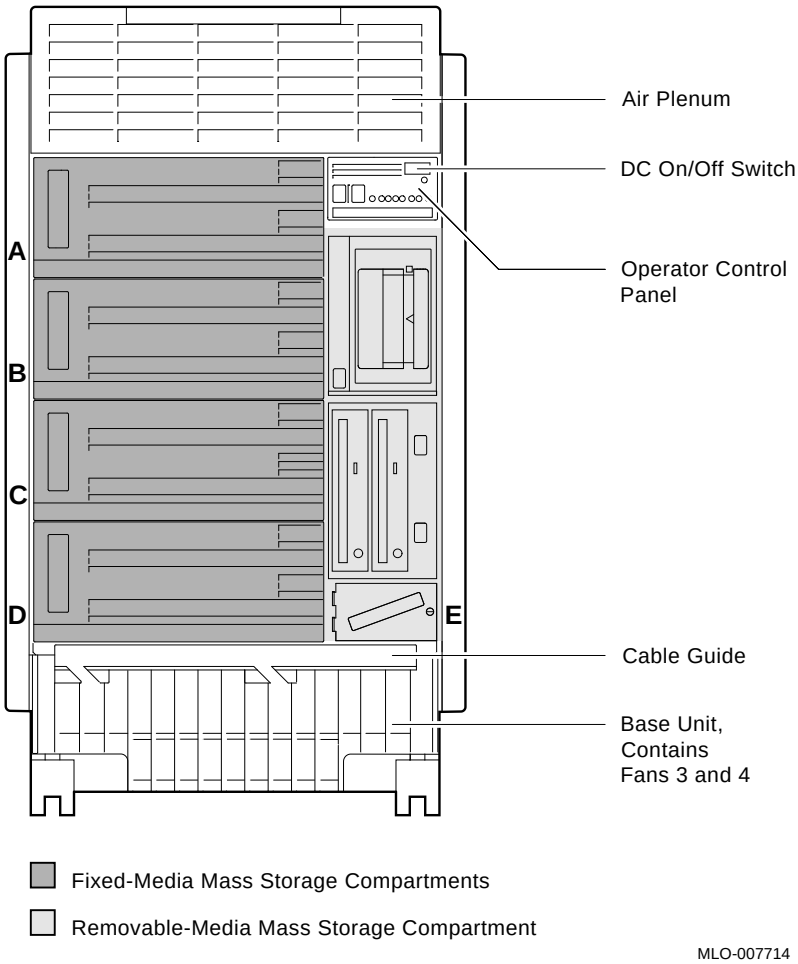
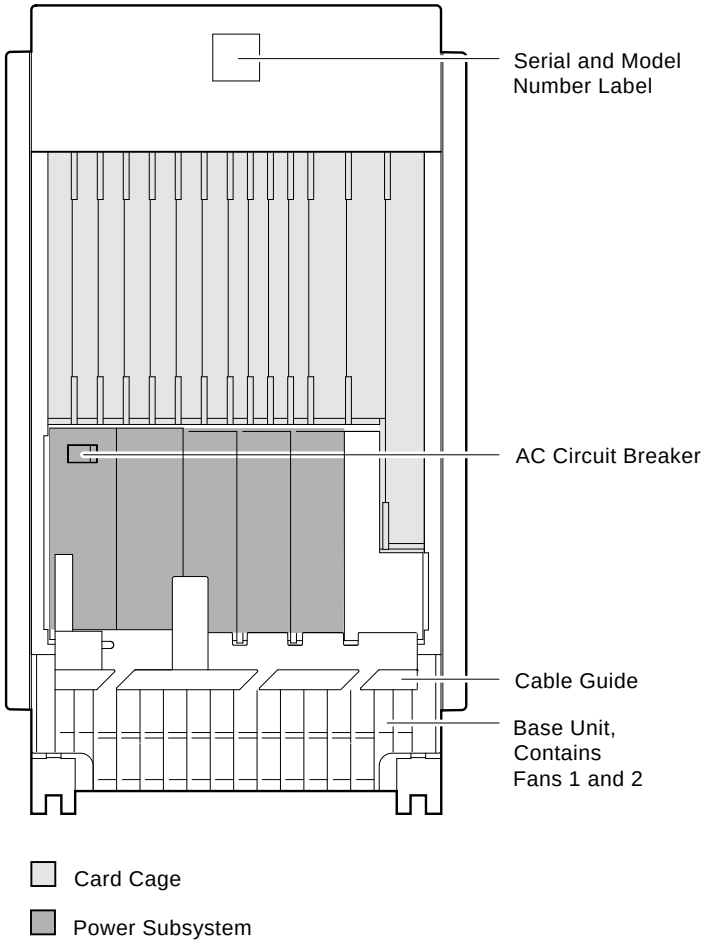


Figure 6–4 BA640 Enclosure (Rear)



6.1.1 System Bus

The system bus interconnects the CPUs, memory modules, and I/O module. The I/O module provides access to basic I/O functions (network, storage devices, and console program). The I/O module also is the adapter to the I/O expansion bus, Futurebus+.

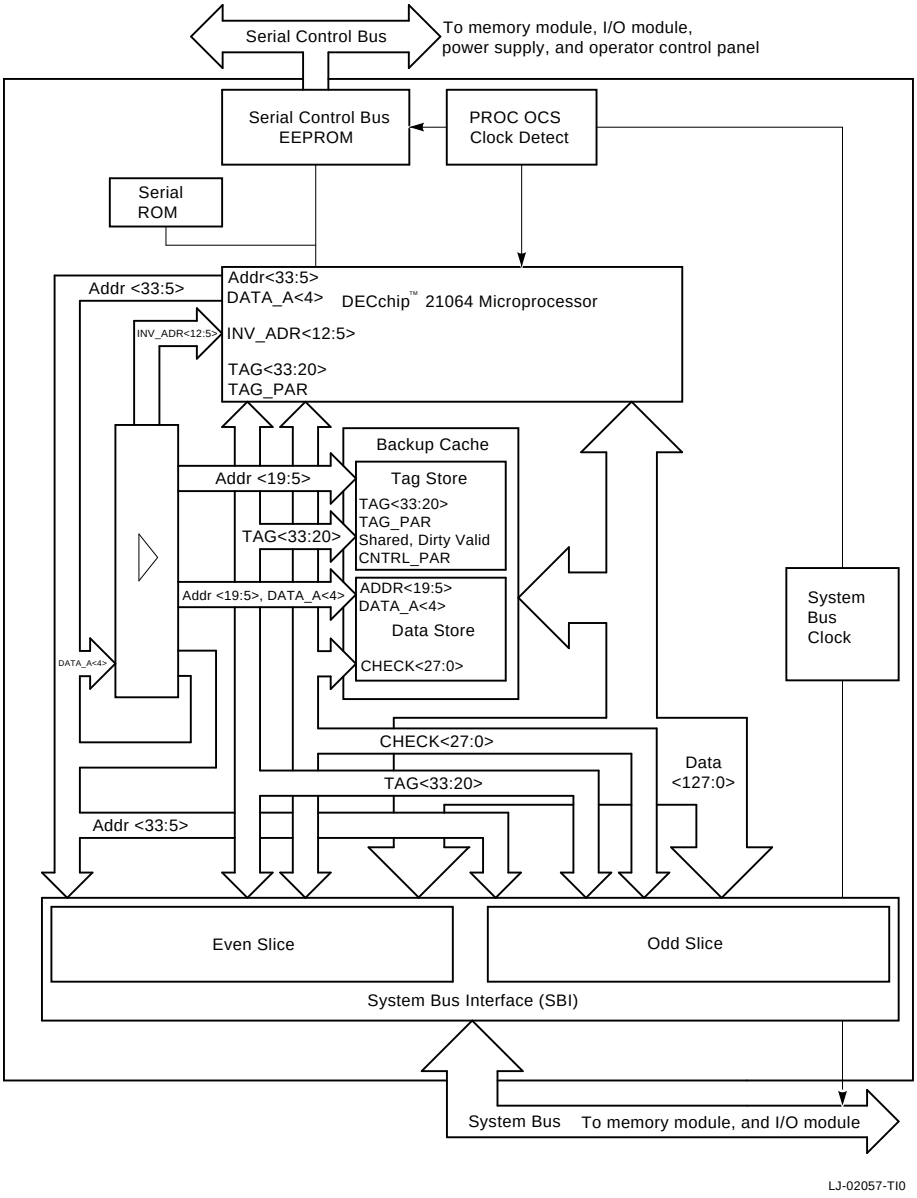
The system bus is a shared-memory bus designed to support the Alpha AXP architecture and up to two processors. It supports a "snooping" protocol that allows a CPU's first-level write-through cache and second-level write-back cache to maintain consistent data with another processor's caches, system memory, and the I/O port on a transaction-by-transaction basis.

The system bus is a synchronous, multiplexed interconnect that can transfer a 34-bit address or a 128-bit data with 32-bit parity in a bus transaction. Two CPU modules and an I/O module arbitrate for the system bus via a prioritized scheme that allows the I/O module to interleave with the two CPU modules. The arbitration function and system bus clock generators are located on the CPU 0 module.

6.1.1.1 KN430 CPU

The KN430 CPU module is based upon the DECchip 21064 processor, designed and manufactured by Digital. The system supports up to two CPU modules in a symmetric multiprocessing configuration. The first CPU is installed in slot 0. For symmetric multiprocessing (SMP), a second CPU is installed in slot 1. Figure 6–5 provides a block diagram of the CPU module.

Figure 6–5 CPU Block Diagram



LJ-02057-TI0

CPU Features

Each CPU has the following features:

- DECchip 21064 processor chip (approximately 100 MIPS, 20 MFLOPS)
- 1-MB direct-mapping backup cache (physical write-back cache, 32-byte block size)
- Interface to system bus (128 bits wide)
- System bus arbiter
- System bus clock generator/distributor

Note

Although both CPUs in a dual-processor system have system bus clock and master bus arbitration circuitry, they are enabled on CPU 0.

- Serial control bus controller for communications with other components of the system

DECchip 21064 Features

The DECchip 21064 microprocessor is a CMOS-4 superscalar, superpipelined implementation of the Alpha AXP architecture.

The microprocessor has the following features:

- All instructions are 32 bits long and have a regular instruction format
- Floating-point unit, supports Digital and IEEE floating-point data types
- 32 integer registers, 64 bits wide
- 32 floating-point registers, 64 bits wide
- On-chip 8-KB, direct-mapping, write-through physical data cache
- On-chip 8-KB, direct-mapping, read-only virtual instruction cache
- On-chip 8-entry I-stream translation buffer
- On-chip 32-entry D-stream translation buffer
- Serial ROM interface for booting and diagnostics
- Clock generator
- Packaged in a 431-pin PGA package.

6.1.1.2 Memory

MS430 memory modules provide high-bandwidth, low-latency program and data storage elements for DEC 4000 AXP systems. Up to four memory modules can be configured in a DEC 4000 AXP system.

The MS430 memory modules are designed to be compatible with two generations of DRAM technology—256K x 4 and 1-MB x 4 parts—and are configured with either two or four banks of DRAMs. Each bank is configured as 32 bytes (256 bits) of data storage and 24 bits for error detection and correction (EDC).

MS430 memory is available in three variations:

- MS430–BA (B2002–BA) 32-MB memory
- MS430–CA (B2002–CA) 64-MB memory
- MS430–DA (B2002–DA) 128-MB memory

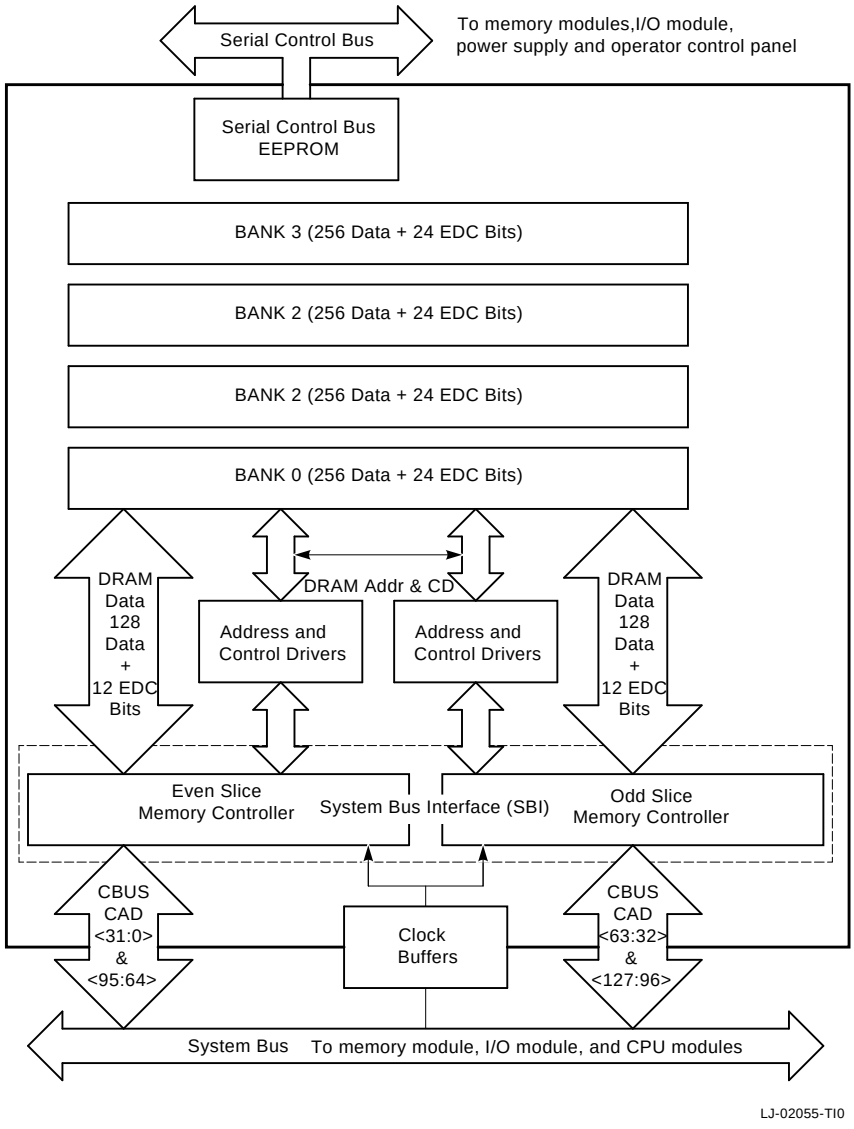
Each memory module provides a number of features in order to improve performance, reliability, and availability. See Table 6–1 below.

Table 6–1 Memory Features

Feature	Description
Error detection and correction (EDC) logic	Improves data reliability and integrity by performing detection and correction of all single-bit errors and the most prevalent forms of 2-bit, 3-bit, and 4-bit errors in the DRAM array.
Write transaction buffers	Improves total memory bandwidth by allowing write transactions to "dump and run." The write command and the write data are placed in internal queues within the memory logic for later execution, allowing the issuing commander to continue processing.
Read stream buffers	Reduces average memory latency while improving total memory bandwidth by allowing each memory module to independently prefetch DRAM data prior to an actual read request for that data. This prefetch or read lookahead activity is statistically driven and is triggered based on the system-bus activity present.
Memory interleaving	Improves total memory bandwidth by overlapping consecutive system-bus memory accesses across 2 or 4 memory modules.
Block exchange	Improves bus bandwidth utilization by paralleling a cache victim write-back with a cache miss fill.
Intelligent refresh control	Reduces average memory latency by scheduling DRAM refresh operations on an opportunistic basis.

Figure 6–6 provides a block diagram of an MS430 memory module.

Figure 6–6 MS430 Memory Block Diagram



6.1.1.3 I/O Module

The KFA40 I/O module contains the base set of necessary I/O functions and is required in all systems. Figure 6–7 provides a block diagram of the I/O module. I/O module functions include:

- Four SCSI-2/DSSI buses for fixed-media devices

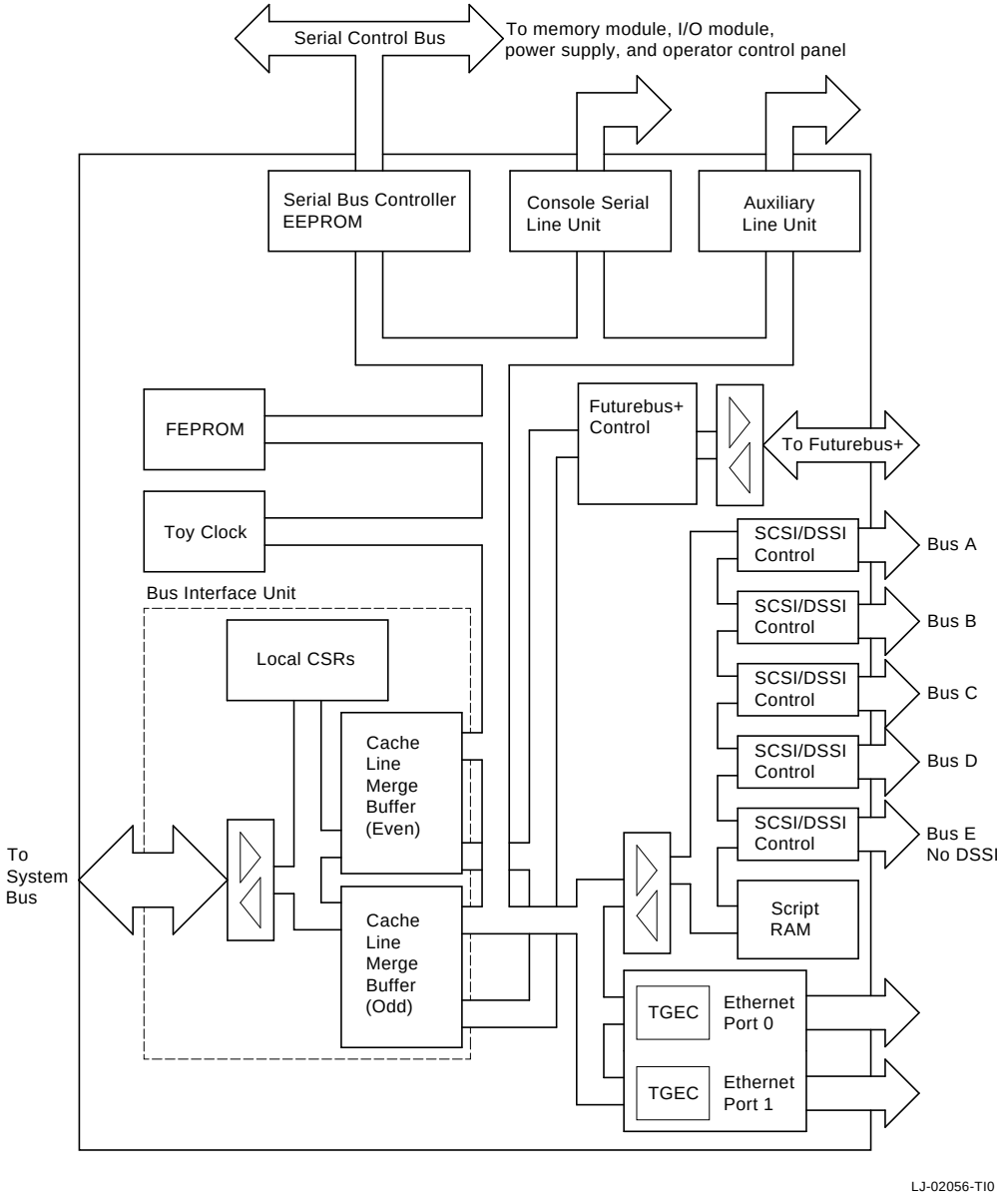
  Note

  Each of the 4 fixed-media buses may operate as a SCSI-2 bus or a DSSI bus. SCSI-2 and DSSI devices can not share the same bus, however.

- One SCSI-2 only bus for removable media devices
- Two Ethernet interfaces, using the third generation Ethernet chip (TGEC).

  Each Ethernet interface has two associated connectors: thickwire (standard Ethernet) and ThinWire. A switch located between the connectors allows you to select the connectors. To connect to a twisted-pair Ethernet, you connect a twisted-pair H3350 media access unit to the thickwire port, using a standard transceiver cable.
- Profile B Futurebus+ bus adapter (allows both 32- and 64-bit data transfers).
- Interface to system bus (128 bits wide) for arbitration with CPU and memory
- Console and diagnostic firmware (512 KB of flash-erasable read-only memory—FEPROM), used in the second stage of power-on diagnostics
- 8 KB of EEROM for console use
- Time-of-year (TOY) clock
- One asynchronous serial line unit (SLU) dedicated to the console subsystem
- One additional asynchronous SLU with modem control
- Serial control bus controller for communications with other components of the system

Figure 6–7 I/O Module Block Diagram



6.1.2 Serial Control Bus

The serial control bus is a two-conductor serial interconnect bus that is independent of the system bus. The serial control bus connects the following modules:

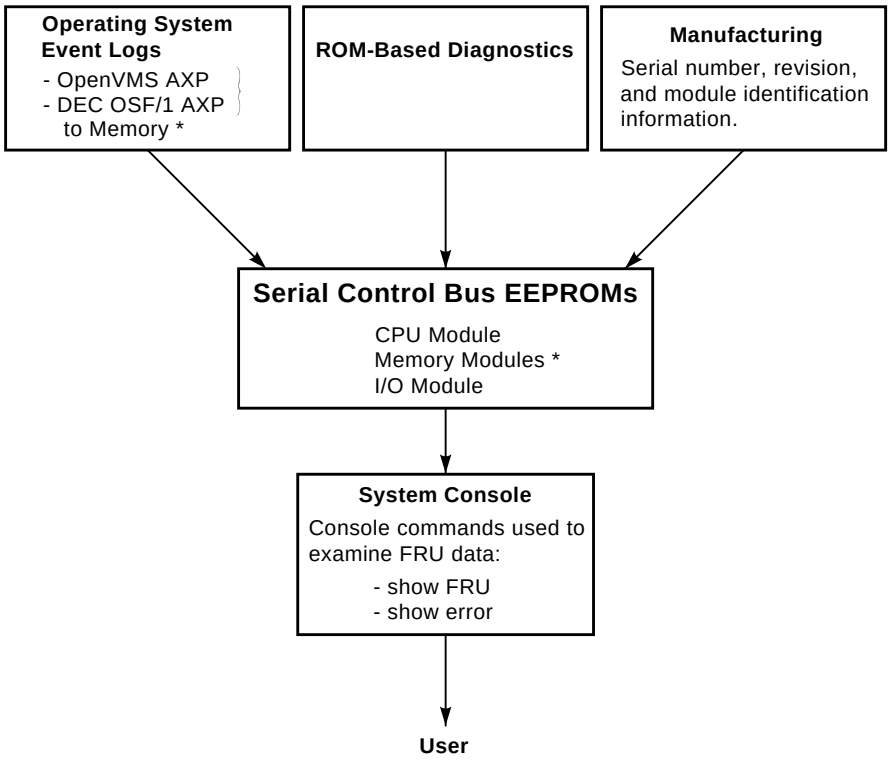
- CPUs
- I/O module
- Memory modules
- Power system controller (PSC)
- Operator control panel (OCP)

The serial control bus communicates with the interfaces on the operator control panel and power system controller, and with the 256-byte error log EEPROM devices on the CPU, I/O, and memory modules. The bus master is located on the I/O module.

The interface on the OCP provides the mechanism for indicating status information on the OCP LEDs.

Figure 6–8 shows where information comes from that is logged to the serial control bus EEPROMs and lists console commands that are commonly used to examine EEPROM data. Some functions illustrated may not be supported on early systems.

Figure 6–8 Serial Control Bus EEPROM Interaction



6.1.3 Futurebus+

DEC 4000 AXP systems implement Futurebus+ as the I/O bus. Features of Futurebus+ include:

- IEEE open standard
- 32- or 64-bit, multiplexed address and data bus
- Asynchronous protocol
- Centralized arbitration
- Multiple priority levels
- 160 MB/s bandwidth, asymptotic

Six Futurebus+ modules can reside in the Futurebus+ portion of the card cage. The slots are numbered 1–6 from right to left.

6.1.4 Power Subsystem

The power subsystem is a universal supply that is designed to operate in all countries. Power for the backplane assembly is provided by the centralized power source. Fixed-media storage devices are powered by local disk converters (LDCs) included in each storage compartment. The power subsystem has five basic components:

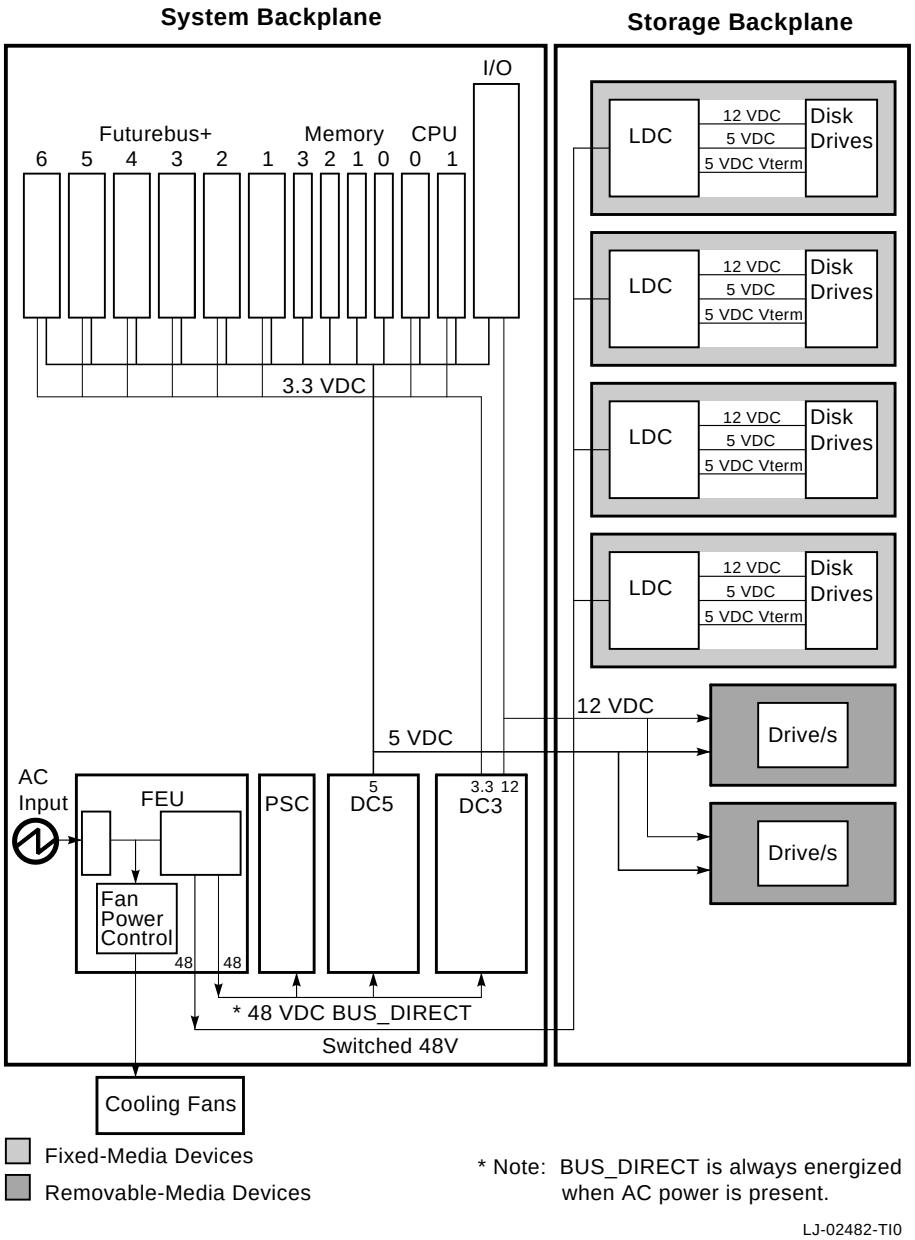
- Front end unit (FEU) (AC to 48 VDC with power factor correction)
- Power system controller (PSC)
- DC5 DC-DC converter unit—5 V. This unit is capable of providing 150 A.
- DC3 DC-DC converter unit—This unit generates three voltages; 12 V at 4 A, 3.3 V at 20 A and 2.1 V at 10 A (Futurebus+ terminator power).
- Local disk converters (LDCs). The local disk converters generate three voltages for storage devices (+5, +12 and +5 V SCSI-2/DSSI terminator voltage).

All of the power supply components (except the LDCs) plug into the system backplane. An LDC is packaged with each fixed-media storage assembly.

System availability is enhanced via an optional, external uninterruptible power supply (UPS). The UPS is able to keep the system running in the event of a power failure.

Figure 6–9 provides a block diagram of the power subsystem components and their function.

Figure 6–9 Power Subsystem Block Diagram



6.1.5 Mass Storage

System mass storage is supported by SCSI-2 and DSSI adapters that reside on the I/O module. Each SCSI-2/DSSI bus is architecturally limited to eight devices, including host adapter.

6.1.5.1 Fixed-Media Compartments

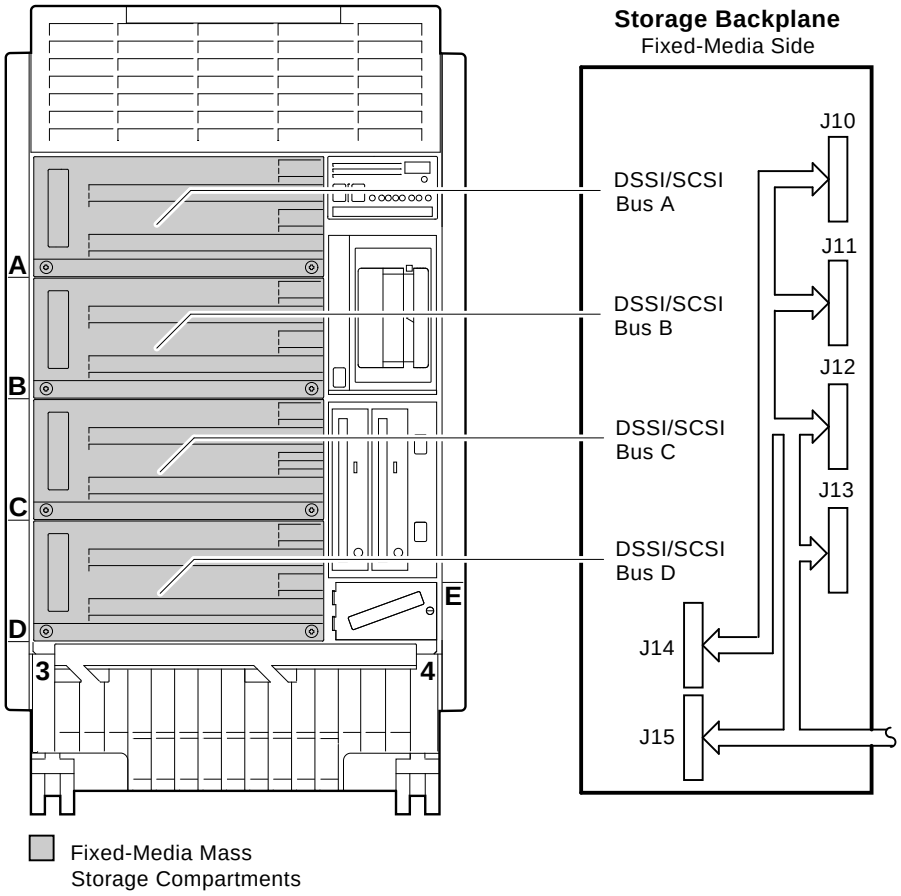
Four DSSI/SCSI-2 adapters support the four fixed-media storage compartments (A–D) (Figure 6–10). For each of the fixed-media compartments, two possible configurations are allowed:

- One full-height 5.25-inch disk
- Up to four 3.5-inch disks

Each adapter provides a separate SCSI/DSSI bus that can support up to eight nodes, where the adapter and each storage device count as one node. Hence, each storage adapter can support up to seven storage devices.

An external connector on the front of each mass storage compartment provides support for external mass storage devices. External devices reside on the same bus as the disks in the mass storage compartment to which they are connected.

Figure 6–10 Fixed-Media Storage



Fixed-Media Configuration Rules

- For each SCSI/DSSI bus, do not duplicate bus node ID numbers for the storage devices. For Bus A, you can have only one storage device identified as bus node 0, one storage device as 1, and so on; for Bus B, you can have only one storage device identified as bus node 0, one storage device as 1, and so on.

- Any one of the four fixed-media compartments can be either SCSI or DSSI, but drives of both types can never be mixed on the same bus. If SCSI devices are chosen, all devices in the mass storage compartment must be SCSI, and external drives connected to that compartment must also be SCSI.
- When more than one DSSI bus is being used and the system is using a nonzero allocation class, you need to assign new MSCP unit numbers for devices on all but the first DSSI bus (Bus A), since the unit numbers for all DSSI devices connected to a system's associated DSSI buses must be unique. Refer to Section 6.4 for more information on setting parameters for DSSI devices.
- By convention, storage devices are numbered in increasing order from right to left, beginning with zero.

Note

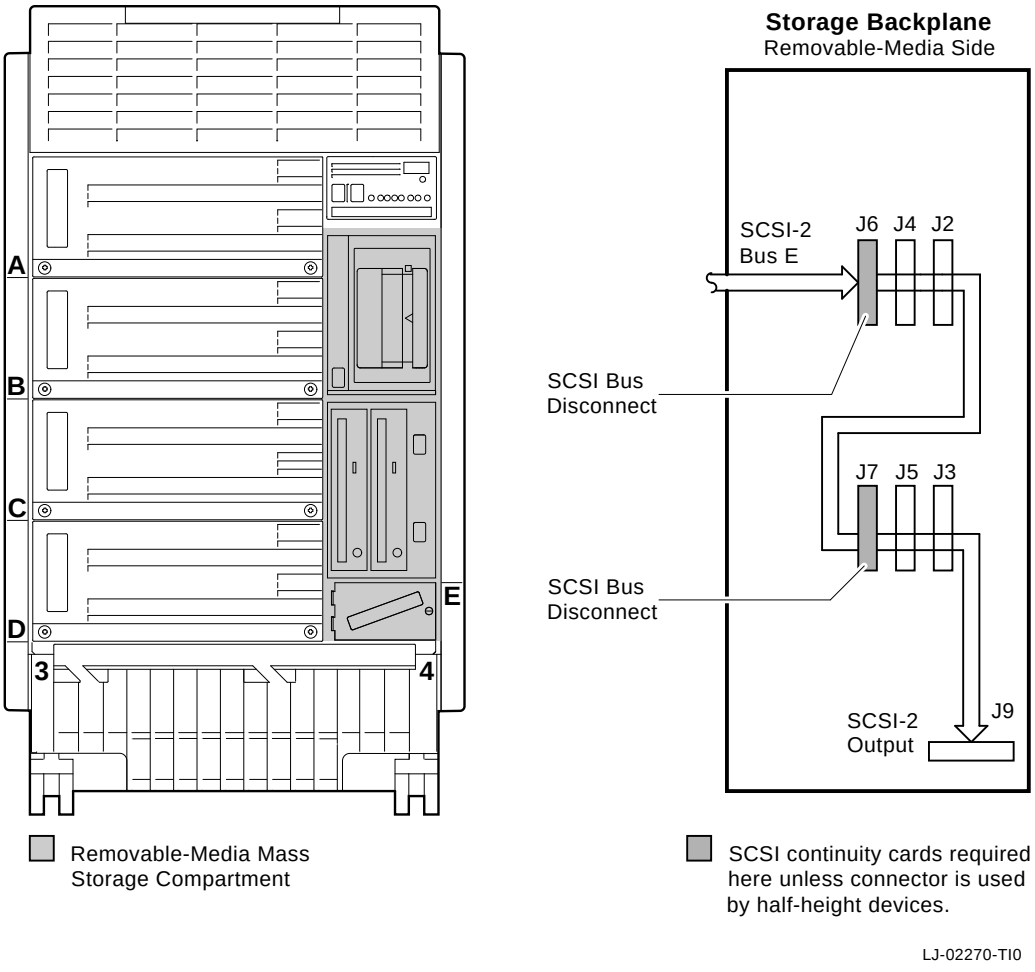
If you change the bus node ID plugs, you must recycle power (press the Reset button or turn on power with the DC on/off switch) before the new setting will take effect. The system reads the bus node ID values at power-up.

6.1.5.2 Removable-Media Storage Compartment

A fifth SCSI adapter supports the removable-media storage compartment (bus E) (Figure 6–11). The removable-media compartment supports:

- Up to four half-height removable-media devices
- Up to two full-height removable-media devices
- One full-height and up to two half-height removable-media devices

Figure 6–11 Removable-Media Storage



Removable-Media Configuration Rules

- Connectors J6 (upper left) and J7 (lower left) of the removable-media storage compartment require either a storage device (half-height) or SCSI continuity card. If a half-height device is installed, store the SCSI continuity card in connectors J4 or J5.

  The continuity card architecture in the SCSI section of the system enclosure is used to minimize the SCSI bus stub length, which is critical to correct operation.

- Do not duplicate bus node ID numbers for your storage devices. For Bus E, you can have only one storage device identified as bus node 0, one storage device as 1, and so on.
- By convention, storage devices in the removable-media storage compartment are numbered in increasing order from left to right, top to bottom, beginning with zero. The TZ30, which uses internal jumper switches to assign its bus node ID, is an exception to this rule. For ease of installation, the TZ30 uses the default setting of five.

Note

If you change the bus node ID plugs, you must recycle power (press the Reset button or turn on power with the DC on/off switch) before the new setting will take effect. The system reads the bus node ID values at power-up.

6.1.6 System Expansion

The R400X mass storage expander provides space for up to seven additional disk drives or up to six disk drives and a tape drive (TZ-, TF-, or TL-series). Using R400X expanders, you can fill four SCSI-2/DSSI buses for a total of up to 28 disks (approximately 28 GB).

6.1.6.1 Power Control Bus for Expanded Systems

The three power bus connectors on the power system controller allow you to configure a power bus for systems expanded with the R400X expander. The power bus allows you to turn power on and off for one or more expanders through the power supply designated as the main power supply (Figure 6–12 and Table 6–2).

Note

DSSI VAXcluster systems should not be configured with a power bus. Inadvertently bringing down the cluster defeats the added reliability of a DSSI VAXcluster.

Figure 6–12 Sample Power Bus Configuration

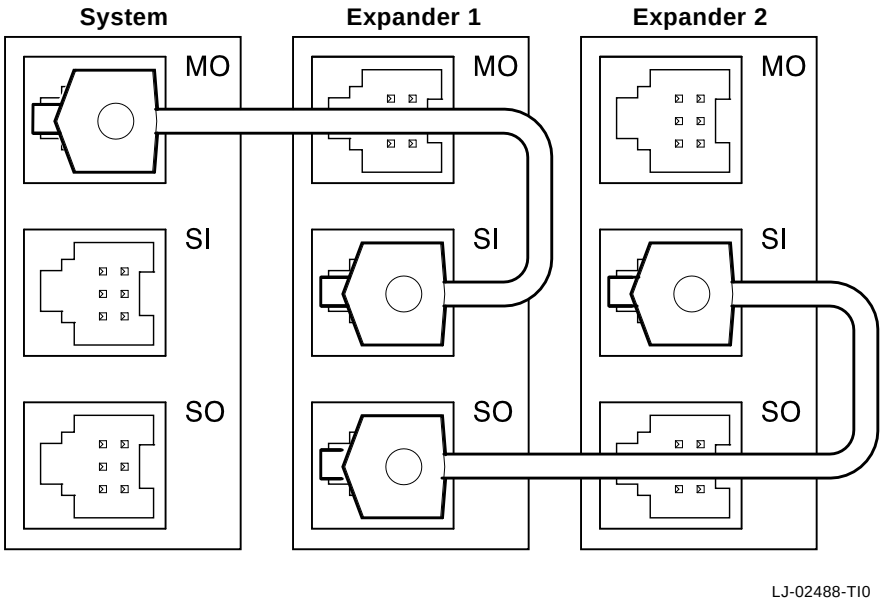


Table 6–2 Power Control Bus

Connector	Function
MO	The main out (MO) connector sends the power control bus signal to the expander. One end of a power bus cable is connected here; the other end is connected to the secondary in (SI) connector of an expander power supply.
SI	The secondary in (SI) connector receives the power control bus signal from the main power supply. In a power bus with more than one expander, the power control bus signal is passed along using the secondary in and out connectors as shown in Figure 6–12.
SO	The secondary out (SO) connector sends the power control bus signal down the power bus for configurations of more than one expander.

6.2 Examining System Configuration

Several console commands are available for examining system configuration:

- `show config` (Section 6.2.1)—Displays the buses on the system and the devices found on those buses.
- `show device` (Section 6.2.2)—Displays the devices and controllers in the system.
- `show memory` (Section 6.2.3)–Displays main memory configuration.

6.2.1 show config

The `show config` command displays the buses found on the system and the devices found on those buses. You can use the information in the display to identify target devices for commands such as `boot` and `test`, as well as to verify that the system sees all the devices that are installed.

Synopsis:

show config

Examples:

```
>>> show config
```

```
Console Vn.n-nnnn   VMS PALcode Xn.nnX, OSF PALcode Xn.nnX

CPU 0          P    B2001-AA DECchip ™ 21064-2
CPU 1          -
Memory 0       -
Memory 1       -
Memory 2       -
Memory 3       P    B2002-DA 128 MB
Ethernet 0     P    Address  08-00-2B-2A-D6-97
Ethernet 1     P    Address  08-00-2B-2A-D6-A6

                    ID 0   ID 1   ID 2   ID 3   ID 4   ID 5   ID 6   ID 7
A    SCSI      P    RZ73                                              Host
B    DSSI      P    RF73                                              Host
C    DSSI      P                                                      Host
D    DSSI      P                                                      Host
E    SCSI      P    TZ85   RRD42                                      Host
Futurebus+     P           FBA0   -      -      -      -      -

System Status Pass              Type b to boot dka0.0.0.0.0
```

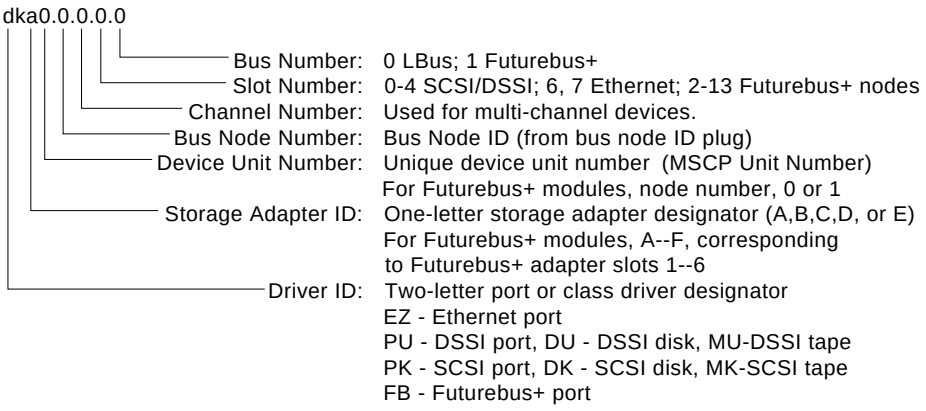
```
>>>
```

LJ-02267-TI0

6.2.2 show device

The `show device` command displays the devices and controllers in the system. The device name convention is shown in Figure 6–13.

Figure 6–13 Device Name Convention



Note

Storage adapter IDs and slot numbers correspond to the mass storage compartments as follows:

Fixed-Media:

Storage compartment A (top): storage adapter a
Storage compartment B: storage adapter b
Storage compartment C: storage adapter c
Storage compartment D (bottom): storage adapter d

Removable-Media:

Storage compartment E: storage adapter e

Synopsis:

show device [device_name]

Arguments:

[device_name] The device name or device abbreviation. When abbreviations or wildcards are used, all devices that match the type are displayed.

Examples:

```
>>> show device
dka0.0.0.0              DKA0                            RZ73
dkc0.0.0.2.0            DKC0                            RZ35
dkc100.1.0.2.0          DKC100                          RZ35
dkc200.2.0.2.0          DKC200                          RZ35
dkc300.3.0.2.0          DKC300                          RZ35
dke400.4.0.4.0          DKE400                         RRD42
dub0.0.0.1.0            R2QZFA$DIA0                     RF72
mke0.0.0.4.0            MKE0                            TZ85
eza0.0.0.6.0            EZA0                08-00-2B-2A-D6-97
ezb0.0.0.7.0            EZB0                08-00-2B-2A-D6-A6
fbc0.0.0.6.1            FBC0          Fbus+ Profile_B Exercis
pka0.7.0.0.0            PKA0                    SCSI Bus ID 7
pke0.7.0.4.0            PKE0                    SCSI Bus ID 7
pub0.7.0.1.0            PIB0                    DSSI Bus ID 7
puc0.7.0.2.0            PIC0                    DSSI Bus ID 7
pud0.7.0.3.0            PID0                    DSSI Bus ID 7

>>> show device fb
fbc0.0.0.6.1            FBC0          Fbus+ Profile_B Exercis

>>> show device dk pk
dka0.0.0.0              DKA0                            RZ73
dkc0.0.0.2.0            DKC0                            RZ35
dkc100.1.0.2.0          DKC100                          RZ35
dkc200.2.0.2.0          DKC200                          RZ35
dkc300.3.0.2.0          DKC300                          RZ35
dke400.4.0.4.0          DKE400                         RRD42
mke0.0.0.4.0            MKE0                            TZ85
pka0.7.0.0.0            PKA0                    SCSI Bus ID 7
pke0.7.0.4.0            PKE0                    SCSI Bus ID 7
>>>
```

Note

If no devices or terminators are present for a SCSI-2/DSSI bus, the display will show an indeterminant device type for that controller, such as p_a0 or p_b0.

6.2.3 show memory

The show memory command displays information for each memory module in the system.

Synopsis:

show memory

Examples:

```
>>> show memory
  ❶      ❷       ❸          ❹          ❺
Module   Size    Base Addr   Intlv Mode  Intlv Unit
------   -----   ---------   ----------  ----------
  0      Not Installed
  1      Not Installed
  2      Not Installed
  3      128MB   00000000      1-Way         0
Total Bad Pages 0 ❻
>>>
```

❶ Module slot number

❷ Size of memory module

❸ Base or starting address of memory module

❹ Interleave mode—number of modules interleaved (1–4-way interleaving)

❺ Interleave unit number

❻ Number of bad pages in memory (8 KB/page)

6.3 Setting and Showing Environment Variables

The environment variables decribed in Table 6–3 are typically set when you are configuring a system. Refer to Appendix A for a complete listing and description of all environment variables.

Table 6–3 Environment Variables Set During System Configuration

Variable	Attributes	Function
auto_action	NV,W	The action the console should take following an error halt or powerfail. Defined values are: BOOT—Attempt bootstrap. HALT—Halt, enter console I/O mode. RESTART—Attempt restart. If restart fails, try boot. No other values are accepted. Other values result in an error message and variable remains unchanged.
bootdef_dev	NV	The device or device list from which booting is to be attempted, when no path is specified on the command line. Set at factory to disk with Factory Installed Software; otherwise null.
boot_file	NV,W	The default filename used for the primary bootstrap when no filename is specified by the `boot` command. The default value when the system is shipped is NULL.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.

(continued on next page)

Table 6–3 (Cont.) Environment Variables Set During System Configuration

Variable	Attributes	Function
boot_osflags	NV,W	Default additional parameters to be passed to system software during booting if none are specified by the `boot` command. On the OpenVMS AXP operating system, these additional parameters are the root number and boot flags. The default value when the system is shipped is NULL. The following parameters are used with the DEC OSF/1 operating system: a Autoboot. Boots /vmunix from bootdef_dev, goes to multiuser mode. Use this for a system that should come up automatically after a power failure. s Stop in single-user mode. Boots /vmunix to single-user mode and stops at the # (root) prompt. i Interactive boot. Request the name of the image to boot from the specified boot device. Other flags, such as -kdebug (to enable the kernel debugger), may be entered using this option. D Full dump, implies "s" as well. By default, if DEC OSF/1 V2.1 crashes, it completes a partial memory dump. Specifying "D" forces a full dump at system crash. Common settings are a, autoboot; and Da, autoboot; but create full dumps if the system crashes.
tta*_baud	NV	Here "*" may be 0 or 1, corresponding to the primary console serial port, tta0 or the auxiliary console serial port, tta1. Specifes the baud rate of the primary console serial port, tta0. Allowable values are 600, 1200, 2400, 4800, 9600, and 19200. The initial value for tta0 is read from the baud rate select switch on the OCP.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.

Synopsis:

set [-default] [-integer] -[string] envar value

show envar

Arguments:

envar	The name of the environment variable to be modified.
value	The value that is assigned to the environment variable. This may be an ASCII string.

Options:

-default	Restores variable to its default value.
-integer	Creates variable as an integer.
-string	Creates variable as a string (default).

Examples:

```
>>> set bootdef_dev eaz0
>>> show bootdef_dev
eza0
>>> show auto_action
boot
>>> set boot_osflags 0,1
>>>
```

6.4 Setting and Examining Parameters for DSSI Devices

For a tutorial on DSSI parameters and their function, refer to Section 6.4.3.

The following console commands are used in setting and examining DSSI device parameters.

- `show device du pu` (Section 6.4.1)—Displays information for each DSSI device on the system (du specifies drives, pu specifies storage adapters).
- `cdp` (Section 6.4.2)—Allows you to modify the following device parameters from console mode: NODENAME, ALLCLASS, and UNITNUM. The `cdp` command automatically connects to the device's DUP server for all devices or any number of specified devices.

6.4.1 show device du pu

The `show device du pu` command displays information for all DSSI devices in the system. The du argument lists all DSSI drives; the pu argument lists the storage adapters for all DSSI buses found on the system.

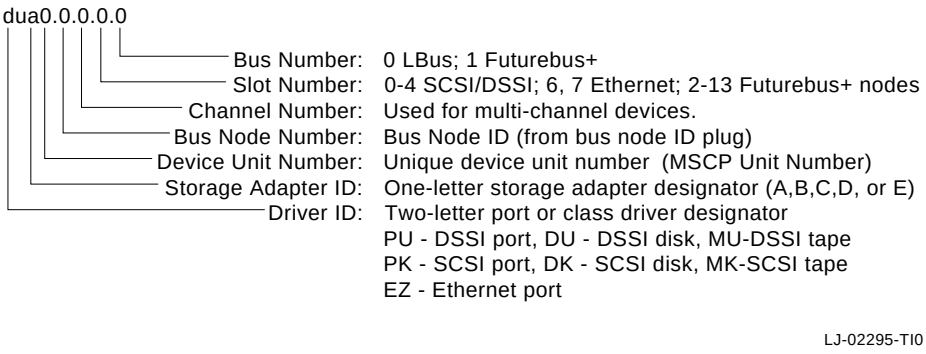
Synopsis:

show device du pu

Example:

```
>>> show device du pu
    ❶                    ❷       ❸                  ❹
dua0.0.0.0.0             $2$DIA0 (ALPHA0)             RF35
dua1.1.0.0.0             $2$DIA1 (ALPHA1)             RF35
dua2.2.0.0.0             $2$DIA2 (ALPHA2)             RF35
dua3.3.0.0.0             $2$DIA3 (ALPHA3)             RF35
pua0.7.0.0.0             PIA0                DSSI Bus ID 7
pub0.7.0.1.0             PIB0                DSSI Bus ID 7
>>>
```

❶ Console device name:



❷ Operating system device name:

- For an allocation class of zero: NODENAME$DIA*u*

  NODENAME is a unique node name and *u* is the unit number. For example, R7BUCC$DIA0.

- For a nonzero allocation class:

  $ALLCLASS$DIA*u*

  ALLCLASS is the allocation class for the system and devices, and *u* is a unique unit number. For example, 1DIA0.

❸ Node name (alphanumeric, up to 6 characters)

❹ Device type

6.4.2 cdp

The `cdp` command allows you to modify NODENAME, ALLCLASS, and UNITNUM from the console program without explicit connection to a node's DUP server.

Entering `cdp` without an option or target device will list the DSSI parameters for all DSSI drives on the system.

Synopsis:

cdp ([-{i,n,a,u,o}] [-sn] [-sa allclass] [-su unitnum] [dssi_device])

Arguments:

[dssi_device]	Name of the DSSI device or DSSI adapter. Only the parameters for the specified device or devices on this adapter will be modified.

Options:

[-i]	Selective interactive mode, set all parameters.
[-n]	Set device node name, NODENAME (alphanumeric, up to 6 characters).
[-a]	Set device allocation class, ALLCLASS.
[-u]	Set device unit number, UNITNUM.
[-sn]	Set node name (NODENAME) for all DSSI drives on the system to either RF*hscn* or TF*hscn*, where: *h* is the device hose number (0) *s* is the device slot number (0–3) *c* is the device channel number (0) *n* is the bus node ID (0–6).
[-sa]	Set ALLCLASS for all DSSI devices on the system to a specified value.
[-su]	Specify a starting unit number for a device on the system. The unit number for subsequent DSSI devices will be incremented (by 1) from the starting unit number.

Examples:

```
>>> cdp
    ❶           ❷           ❸              ❹ ❺    ❻
pua0.0.0.0.0    ALPHA0      0411214901371      2  0  $2$DIA0
pua0.1.0.0.0    ALPHA1      0411214901506      2  1  $2$DIA1
pua0.2.0.0.0    ALPHA2      041122A001625      2  2  $2$DIA2
pua0.3.0.0.0    ALPHA3      0411214901286      2  3  $2$DIA3
>>>
```

❶ Storage adapter device name

❷ Node name (NODENAME)

❸ System ID (SYSTEMID)—modified during warmswap.

❹ Allocation class (ALLCLASS)

❺ Unit number (UNITNUM)

❻ Operating system device name

```
>>> cdp dua* -su 10
pua0.0.0.0.0    ALPHA0      0411214901371      2  10  $2$DIA10
pua0.1.0.0.0    ALPHA1      0411214901506      2  11  $2$DIA11
pua0.2.0.0.0    ALPHA2      041122A001625      2  12  $2$DIA12
pua0.3.0.0.0    ALPHA3      0411214901286      2  13  $2$DIA13
>>>

>>> cdp -sn
```

```
pua0.0.0.0.0    RF0000     0411214901371      2  10  $2$DIA10
pua0.1.0.0.0    RF0001     0411214901506      2  11  $2$DIA11
pua0.2.0.0.0    RF0002     041122A001625      2  12  $2$DIA12
pua0.3.0.0.0    RF0003     0411214901286      2  13  $2$DIA13
>>>

>>> cdp -i dua13

pua13.3.0.0.0:
Node Name [RF0003]? ALPHA13
Allocation Class [2]? 1
Unit Number [13]? 5
>>>
```

6.4.3 DSSI Device Parameters: Definitions and Function

Five principal parameters are associated with each DSSI device:

- Bus node ID
- ALLCLASS
- UNITNUM
- NODENAME
- SYSTEMID

Note

ALLCLASS, NODENAME, and UNITNUM are examined and modified using the `cdp` command (Section 6.4.2).

SYSTEMID is examined and modified using the console-based Diagnostic Utility Program (DUP) server utility.

The bus node ID is physically determined by the numbered bus node ID plug that inserts into the front panel of the storage compartment.

A brief description of each parameter follows:

Bus Node ID

The bus node ID parameter is provided by the bus node ID plug on the front panel of the storage compartment. Each DSSI bus can support up to eight nodes, (bus nodes 0–7). Each DSSI adapter and each device count as a node. Hence, in a single-system configuration, a DSSI bus can support up to seven devices, bus nodes 0–6 (with node 7 reserved for the adapter).

ALLCLASS

The ALLCLASS parameter determines the device allocation class. The allocation class is a numeric value from 0–255 that is used by the OpenVMS AXP operating system to derive a path-independent name for multiple access paths to the same device. The ALLCLASS firmware parameter corresponds to the OpenVMS AXP IOGEN parameter ALLOCLASS.

DSSI devices are shipped from the factory with a default allocation class of zero.

Note

Each device to be served to a cluster must have a nonzero allocation class that matches the allocation class of the system.

Refer to the *VMS VAXcluster* manual for rules on specifying allocation class values.

UNITNUM

The UNITNUM parameter determines the unit number of the device. By default, the device unit number is supplied by the bus node ID plug on the front panel of the storage compartment.

Note

Systems using multiple DSSI buses, as described later in this section, require that the default values be replaced with unique unit numbers.

To set unit numbers and override the default values, you use the `cdp` console command to supply values to the UNITNUM parameter.

NODENAME

The NODENAME parameter allows each device to have an alphanumeric node name of up to six characters. DSSI devices are shipped from the factory with a unique identifier, such as R7CZZC, R7ALUC, and so on. You can provide your own node name.

SYSTEMID

The SYSTEMID parameter provides a number that uniquely identifies the device to the operating system. This parameter is modified when you replace a device using warm-swapping procedures.

6.4.3.1 How OpenVMS AXP Uses the DSSI Device Parameters

This section describes how the OpenVMS AXP operating system uses the parameters to create unique identifiers for each device. Configurations that require you to assign new unit numbers for devices are also described.

- With an allocation class of zero, the operating system can use the default parameter values to provide each device with a unique device name. The operating system uses the node name along with the device logical name in the following manner:

  NODENAME$DIA*u*

  NODENAME is a unique node name and *u* is the unit number. For example, R7BUCC$DIA0.

- With a nonzero allocation class, the operating system relies on unit number values to create a unique device name. The operating system uses the allocation class along with the device logical name in the following manner:

  $ALLCLASS$DIA*u*

  ALLCLASS is the allocation class for the system and devices, and *u* is a unique unit number. For example, 1DIA0.

With DEC 4000 AXP systems, you can fill multiple DSSI buses: buses A–D (slot numbers 0–3). Each bus can have up to seven DSSI devices (bus nodes 0–6). When more than one bus is being used, and your system is using a nonzero allocation class, you need to assign new unit numbers for devices on all but one of the DSSI buses, since the unit numbers for all DSSI storage devices connected to a system's associated DSSI buses must be unique.

Figure 6–14 illustrates the problem of duplicate operating system device names for a system that is using more than one DSSI bus and a nonzero allocation class. In the case of the nonzero allocation class, the operating system sees four of the devices as having duplicate device names. This is an error, as all unit numbers must be unique. The unit numbers for one of the two DSSI buses in this example need to be reprogrammed.

Figure 6–14 How OpenVMS Sees Unit Numbers for DSSI Devices

Allocation Class=0	Nonzero Allocation Class (Example: ALLCLASS=1)
R7BUCC$DIA0	1DIA0 ◄ *Duplicate 0
R7CZZC$DIA1	1DIA1 ◄ *Duplicate 1
R7ALUC$DIA2	1DIA2 ◄ *Duplicate 2
R7EB3C$DIA3	1DIA3 ◄ *Duplicate 3
R7IDFC$DIA0	1DIA0 ◄
R7IBZC$DIA1	1DIA1 ◄
R7IKJC$DIA2	1DIA2 ◄
R7ID3C$DIA3	1DIA3 ◄
R7XA4C$DIA4	1DIA4
R7QIYC$DIA5	1DIA5
R7DA4C$DIA6	1DIA6

* Nonzero allocation class examples with an asterisk indicate duplicate device names. For one of the DSSI buses, the unit numbers need to be reprogrammed to avoid this error.

LJ-02063-TI0

6.4.3.2 Example: Modifying DSSI Device Parameters

In the following example, the allocation class will be set to 1, the devices for Bus A (in the DEC 4000 AXP system) will be assigned new unit numbers (to avoid the problem of duplicate unit numbers), and the system disk will be assigned a new node name.

Figure 6–15 shows sample DSSI buses and bus node IDs for a sample expanded DEC 4000 AXP system.

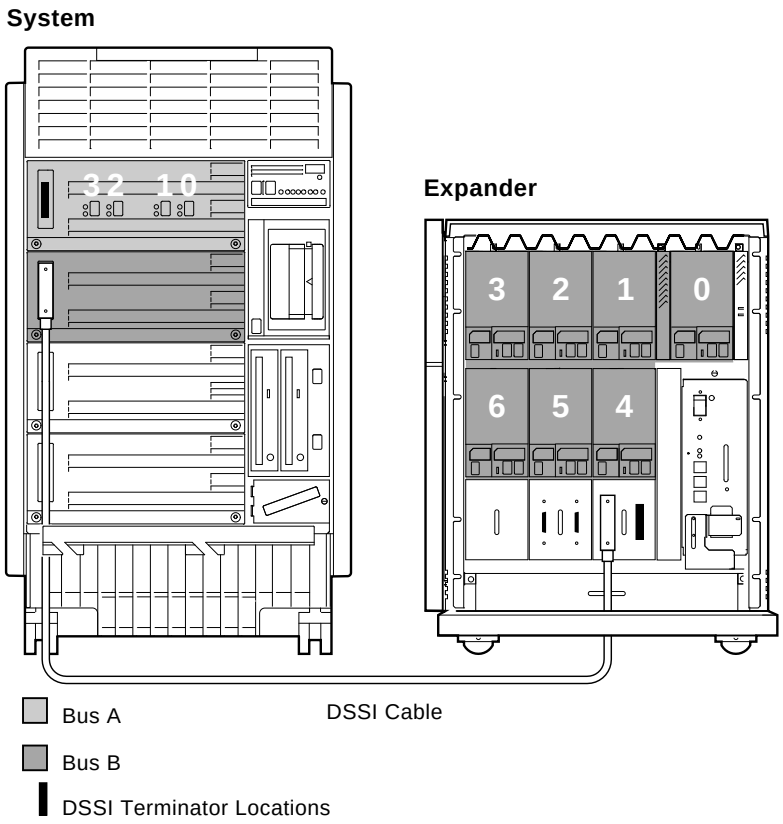
```
>>> show device du pu #Displays all DSSI devices

dua0.0.0.0.0              $2$DIA0 (ALPHA0)             RF35
dua1.1.0.0.0              $2$DIA1 (ALPHA1)             RF35
dua2.2.0.0.0              $2$DIA2 (ALPHA2)             RF35
dua3.3.0.0.0              $2$DIA3 (ALPHA3)             RF35
dub0.0.0.1.0              $2$DIA0 (SNEEZY)             RF73
dub1.1.0.1.0              $2$DIA1 (DOPEY)              RF73
dub2.2.0.1.0              $2$DIA2 (SLEEPY)             RF73
dub3.3.0.1.0              $2$DIA3 (GRUMPY)             RF73
dub4.4.0.1.0              $2$DIA4 (BASHFUL)            RF73
dub5.5.0.1.0              $2$DIA5 (HAPPY)              RF73
dub6.6.0.1.0              $2$DIA6 (DOC)                RF73
pua0.7.0.0.0              PIA0                DSSI Bus ID 7
pub0.7.0.1.0              PIB0                DSSI Bus ID 7
>>> cdp -sa 1 -su 10 dua* #Assigns ALLCLASS of 1
 #to all drives in the system; assigns UNITNUM 10, 11, 12,
 #and 13 to the drives on bus a.

pua0.0.0.0.0    ALPHA0     0411214901371      1  10  $1$DIA10
pua0.1.0.0.0    ALPHA1     0411214901506      1  11  $1$DIA11
pua0.2.0.0.0    ALPHA2     041122A001625      1  12  $1$DIA12
pua0.3.0.0.0    ALPHA3     0411214901286      1  13  $1$DIA13
pub0.0.0.1.0    SNEEZY     0411214906794      1  0   $1$DIA0
pub1.1.0.1.0    DOPEY      0411214457623      1  1   $1$DIA1
pub2.2.0.1.0    SLEEPY     0478512447890      1  2   $1$DIA2
pub3.3.0.1.0    GRUMPY     0571292500565      1  3   $1$DIA3
pub4.4.0.1.0    BASHFL     0768443122700      1  4   $1$DIA4
pub5.5.0.1.0    HAPPY      0768443122259      1  5   $1$DIA5
pub6.6.0.1.0    DOC        0768442231111      1  6   $1$DIA6
>>> cdp -n dub0 #Allows you to modify NODENAME
 #for the specified drive.

pub0.0.0.1.0:
Node Name [SNEEZY]? SYSTEM
>>> show device du pu
dua10.0.0.0.0             $1$DIA10 (ALPHA0)            RF35
dua11.1.0.0.0             $1$DIA11 (ALPHA1)            RF35
dua12.2.0.0.0             $1$DIA12 (ALPHA2)            RF35
dua13.3.0.0.0             $2$DIA13 (ALPHA3)            RF35
dub0.0.0.1.0              $1$DIA0  (SYSTEM)            RF73
dub1.1.0.1.0              $1$DIA1  (DOPEY)             RF73
dub2.2.0.1.0              $1$DIA2  (SLEEPY)            RF73
dub3.3.0.1.0              $1$DIA3  (GRUMPY)            RF73
dub4.4.0.1.0              $1$DIA4  (BASHFL)            RF73
dub5.5.0.1.0              $1$DIA5  (HAPPY)             RF73
dub6.6.0.1.0              $1$DIA6  (DOC)               RF73
pua0.7.0.0.0              PIA0                DSSI Bus ID 7
pub0.7.0.1.0              PIB0                DSSI Bus ID 7
>>>
```

Figure 6–15 Sample DSSI Buses for an Expanded DEC 4000 AXP System



6.5 Console Port Baud Rate

Two serial console ports are provided on the I/O module:

- The console serial port that connects to the console terminal via a DECconnect cable
- The auxiliary serial port with modem support

6.5.1 Console Serial Port

The baud rate for the console serial is set at the factory to 9600 bits per second. Most Digital terminals are also shipped with a baud rate of 9600.

You can select a baud rate for the console serial port using the volatile environment variable, tta0_baud. Allowable values are 600, 1200, 2400, 4800, 9600, and 19200. Use the `set` command to assign values to the tta0_baud environment variable. At power-up, the console serial port baud rate is read from the baud rate select switch.

You can manually select a baud rate for the console serial port using the baud rate select switch located behind the OCP (Figure 6–16). The switch also allows you to power up without initiating drivers (switch position 0, robust mode). Refer to Section 2.2.3 for information on using robust mode to solve problems getting to the console program. Table 6–4 provides the baud rates as they correspond to the rotary switch setting.

Note

The baud rate select switch should be changed only when power is off, as it is read by the system during power-on self-tests.

Figure 6–16 Console Baud Rate Select Switch

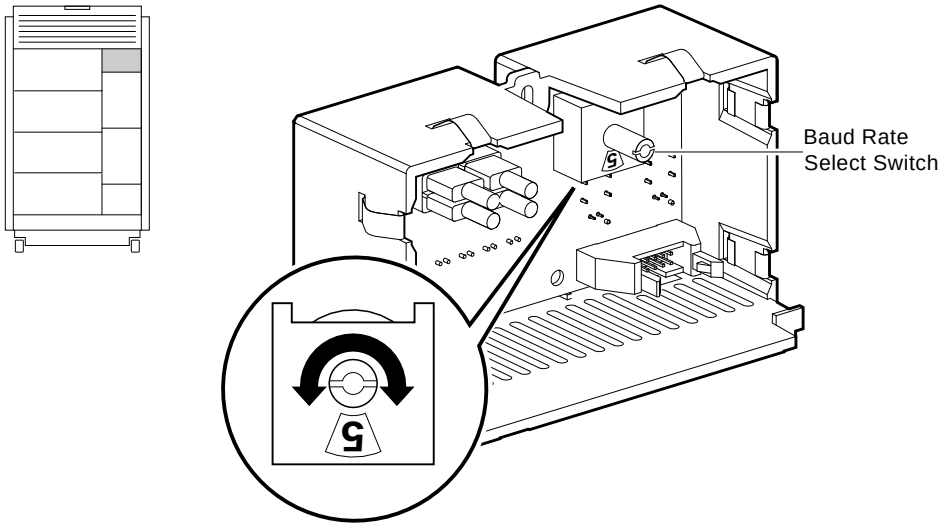


Table 6–4 Console Line Baud Rates

Switch Number	Baud Rate (Bits/S)
0	9600 Robust Mode—Power up without running diagnostics or initiating drivers.
1	600
2	1200
3	2400
4	4800
5	9600
6	19200
7	38400

6.5.2 Auxiliary Serial Port

The baud rate for the auxiliary serial port is set via the nonvolatile environment variable, tta1_baud. Allowable values are 600, 1200, 2400, 4800, 9600, and 19200. Use the `set` command to assign values to the tta1_baud environment variable.

A
Environment Variables

All supported environment variables are listed in Table A–1.

Table A–1 Environment Variables

#	Variable	Attributes	Function
		Alpha AXP SRM-Defined Environment Variables	
00			Reserved
01	auto_action	NV,W	The action the console should take following an error halt or powerfail. Defined values are: BOOT—Attempt bootstrap. HALT—Halt, enter console I/O mode. RESTART—Attempt restart. If restart fails, try boot. No other values are accepted. Other values result in an error message and variable remains unchanged.
03	bootdef_dev	NV	The device or device list from which booting is to be attempted, when no path is specified on the command line (set at factory to disk with Factory Installed Software; otherwise null).
04	booted_dev	RO	The device from which booting actually occurred.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
		Alpha AXP SRM-Defined Environment Variables	
05	boot_file	NV,W	The default filename used for the primary bootstrap when no filename is specified by the `boot` command. The default value when the system is shipped is NULL.
06	booted_file	RO	The filename used for the primary bootstrap during the last boot. The value is NULL if boot_file is NULL and no bootstrap filename was specified by the `boot` command.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
			Alpha AXP SRM-Defined Environment Variables
07	boot_osflags	NV,W	Default additional parameters to be passed to system software during booting if none are specified by the `boot` command. On the OpenVMS AXP operating system, these additional parameters are the root number and boot flags. The default value when the system is shipped is NULL. The following parameters are used with the DEC OSF/1 operating system: a Autoboot. Boots /vmunix from bootdef_dev, goes to multiuser mode. Use this for a system that should come up automatically after a power failure. s Stop in single-user mode. Boots /vmunix to single-user mode and stops at the # (root) prompt. i Interactive boot. Request the name of the image to boot from the specified boot device. Other flags, such as -kdebug (to enable the kernel debugger), may be entered using this option. D Full dump, implies "s" as well. By default, if DEC OSF/1 V2.1 crashes, it completes a partial memory dump. Specifying "D" forces a full dump at system crash. Common settings are a, autoboot; and Da, autoboot; but create full dumps if the system crashes.
08	booted_osflags	RO	Additional parameters, if any, specified by the last `boot` command that are to be interpreted by system software. The default value when the system is shipped is NULL.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
			Alpha AXP SRM-Defined Environment Variables
09	boot_reset	NV,W	Indicates whether a full system reset is performed in response to an error halt or `boot` command. Defined values and the action taken are: OFF—warm boot, no full reset is performed. ON —cold boot, a full reset is performed. The default value when the system is shipped is OFF.
0A	dump_dev	NV,W	The complete device specification of the device to which operating system dumps should be written. The default value when the system is shipped indicates a valid implementation-dependent device.
0B	enable_audit	NV,W	Indicates whether audit trail messages are to be generated during bootstrap. OFF—Suppress audit trail messages. ON —Generate audit trail messages. The system is shipped with this set to ON.
0D	char_set	NV,W	Indicates the character set encoding currently selected to be used for the console terminal. 0—ISO-LATIN-1 character encoding The default value when the system is shipped is 0.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
		Alpha AXP SRM-Defined Environment Variables	
0E	language	NV,W	The default language to display critical system messages. 00 none (cryptic) 30 Dansk 32 Deutsch 34 Deutsch (Schweiz) 36 English (American) 38 English (British/Irish) 3A Espanol 3C Francais 3E Francais (Canadian) 40 Francais (Suisse Romande) 42 Italiano 44 Nederlands 46 Norsk 48 Portugues 4A Suomi 4C Svenska 4E Vlaams
0F	tty_dev	NV,W,RO	Specifies the current console terminal unit. Indicates which entry of the CTB table corresponds to the actual console terminal. The value is preserved across warm bootstraps. The default value is "0" 30 (hex).
10-3F			Reserved for Digital.
40-7F			Reserved for console use.
80-FF			Reserved for operating system use.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
	System-Dependent Environment Variables		
	cpu_enabled	NV	A bit mask indicating which processors are enabled to run (leave console mode). If this variable is not defined, all available processors are considered enabled.
	d_bell		Specifies whether or not to bell on error if error is detected. OFF (default) ON
	d_cleanup		Specifies whether or not cleanup code is executed at the end of a diagnostic. ON (default) OFF
	d_complete		Specifies whether or not to display the diagnostic completion message. OFF (default) ON
	d_eop		Specifies whether or not to display end-of-pass messages. OFF (default)—Disable end-of-pass messages. ON—Enable end-of-pass messages.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
			System-Dependent Environment Variables
	d_group		Specifies the diagnostic group to be executed. FIELD (default) MFG Other diagnostic group string (up to 32 characters)
	d_harderr		Specifies the action taken following hard error detection. CONTINUE HALT (default) LOOP
	d_oper		Specifies whether or not an operator is present. ON —Indicates operator present. OFF (default)—Indicates no operator present.
	d_passes		Specifies the number of passes to run a diagnostic module. 1 (default) 0—Indicates to run indefinitely an arbitrary value
	d_report		Specifies the level of information provided by diagnostic error reports. SUMMARY (default) FULL OFF

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
	System-Dependent Environment Variables		
	d_softerr		Specifies the action taken following soft error detection. CONTINUE (default) HALT LOOP
	d_startup		Specifies whether or not to display the diagnostic startup message. OFF (default)—Disables the startup message. ON—Enables the startup message.
	d_trace		Specifies whether or not to display test trace messages. OFF (default)—Disables trace messages. ON—Enables trace messages.
	enable_servers		Allows a diskless storage bus to respond as if it contains a DSSI disk drive—for use in DSSI loopback testing. OFF (default)—Disables phantom RX50 DSSI device. ON—Enables phantom RX50 DSSI device.
	etherneta		Specifies the Ethernet station address for port eza0.
	ethernetb		Specifies the Ethernet station address for port ezb0.
	exdep_data	RO	Specifies the data value referenced by the last `examine` or `deposit` command.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
	System-Dependent Environment Variables		
	exdep_location	RO	Specifies the location referenced by the last `examine` or `deposit` command.
	exdep_size	RO	Specifies the data size referenced by the last `examine` or `deposit` command.
	exdep_space	RO	Specifies the address space referenced by the last `examine` or `deposit` command.
	exdep_type	RO	Specifies the data type referenced by the last `examine` or `deposit` command.
	ez*0_arp_tries	NV	Sets the number of transmissions that are attempted before the ARP protocol fails. Values less than 1 cause the protocol to fail immediately. Default value is 3, which translates to an average of 12 seconds before failing. Interfaces on busy networks may need higher values.
	ez*0_bootp_file	NV	Supplies the generic filename to be included in a BOOTP request. The BOOTP server will return a fully qualified filename for booting. This can be left empty.
	ez*0_bootp_server	NV	Supplies the server name to be included in a BOOTP request. This can be set to the name of the server from which the machine is to be booted, or can be left empty.
	ez*0_bootp_tries	NV	Sets the number of transmissions that are attempted before the BOOTP protocol fails. Values less than 1 cause the protocol to fail immediately. Default value is 3, which translates to an average of 12 seconds before failing. Interfaces on busy networks may need higher values.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
	System-Dependent Environment Variables		
	ez*0_def_ginetaddr	NV	Supplies the initial value for ez*0_ginetaddr when the interface's internal Internet database is initialized from NVRAM (ez*0_inet_init is set to "nvram").
	ez*0_def_inetaddr	NV	Supplies the initial value for ez*0_inetaddr when the interface's internal internet database is initialized from NVRAM (ez*0_inet_init is set to "nvram").
	ez*0_def_inetfile	NV	Supplies the initial value for ez*0_inetfile when the interface's internal Internet database is initialized from NVRAM (ez*0_inet_init is set to "nvram").
	ez*0_def_sinetaddr	NV	Supplies the initial value for ez*0_sinetaddr when the interface's internal Internet database is initialized from NVRAM (ez*0_inet_init is set to "nvram").
	ez*0_driver_flags		Specifies the flags to be used by the driver. Current values are: 1 NDL$M_ENA_BROADCAST will enable broadcast messages. 2 NDL$M_ENA_HASH will enable hash filtering. 4 NDL$M_ENA_INVF will enable inverse filtering. 8 NDL$M_MEMZONE will allocate the message buffers from memzone.
	ez*0_ginetaddr		Accesses the gateway address field of the interface's internal Internet database. This is normally the address of the local network's gateway to other networks.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
	System-Dependent Environment Variables		
	ez*0_inet_init	NV	Determines whether the interface's internal Internet database is initialized from NVRAM or from a network server (via the BOOTP protocol). Legal values are "nvram" and "bootp"; default is "bootp."
	ez*0_inetaddr		The local address field of the interface's internal Internet database.
	ez*0_inetfile		Accesses the filename field of the interface's internal Internet database. This is normally the file to be booted from the TFTP server. This variable supplies the default remote filename for TFTP transactions.
	ez*0_loop_count		Specifies the number of times each message is looped.
	ez*0_loop_inc		Specifies the amount the message size is increased from message to message.
	ez*0_loop_patt		Specifies the type of data pattern to be used when doing loopback. Current patterns are accessed by the following: 0xffffffff = All the patterns 0 = all zeros 1 = all ones 2 = all fives 3 = all A's 4 = incrementing 5 = decrementing
	ez*0_loop_list_size		Specifies the size of the preallocated list used during loopback.
	ez*0_loop_size		Specifies the size of the loop data to be used.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
	System-Dependent Environment Variables		
	ez*0_lp_msg_node		Specifies the number of messages originally sent to each node.
	ez*0_mode		Specifies the value for the SGEC mode when the device is started. This value is a mirror of CSR6. It can be different from device to device.
	ez*0_msg_buf_size		Specifies the message size. Receive data chaining can be achieved by picking a small value for this variable.
	ez*0_msg_mod		Specifies the modulus for message alignment.
	ez*0_msg_rem		Specifies the remainder for message alignment.
	ez*0_protocols	NV	Determines which network protocols are enabled for booting and other functions. Legal values include BOOTP, MOP, and BOOTP,MOP. A null value is equivalent to "BOOTP,MOP."
	ez*0_rcv_buf_no		Specifies the number of receive buffers.
	ez*0_rcv_mod		Specifies the modulus for receive descriptor alignment.
	ez*0_rcv_rem		Specifies the remainder for receive descriptor alignment.
	ez*0_rm_boot	NV	Enables or disables remote booting or triggering of a system using a DECnet Maintenance Operations Protocol (MOP) Version 4 boot message directed at the Ethernet port, either eza0 or ezb0. Setting this to 1 enables remote booting. The default setting is 0 or disabled.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
	System-Dependent Environment Variables		
	ez*0_rm_boot_ passwd	NV	Sets the MOP Version 4 boot message password for the Ethernet port, either eza0 or ezb0. This password should be entered in hexadecimal in the form "01-longword-longword," for instance, "01-01234567-89abcdef." The leading byte should normally be "01" when enabled. The default setting is "00-00000000-00000000."
	ez*0_sinetaddr		Accesses the server address field of the interface's internal Internet database. This is normally the address of the BOOTP and TFTP server. This variable supplies the default remote address for TFTP transactions.
	ez*0_tftp_tries	NV	Sets the number of transmissions that are attempted before the TFTP protocol fails. Values less than 1 cause the protocol to fail immediately. Default value is 3, which translates to an average of 12 seconds before failing. Interfaces on busy networks may need higher values.
	ez*0_xmt_buf_no		Specifies the number of transmit buffers.
	ez*0_xmt_int_msg		Specifies the number of transmit interrupts per message.
	ez*0_xmt_max_size		Specifies the maximum message size that can be transmitted. Transmit data chaining can be achieved by picking a small value for this variable.
	ez*0_xmt_mod		Specifies the modulus for transmit descriptor alignment.
	ez*0_xmt_msg_post		Specifies the number of messages before posting a transmit.
	ez*0_xmt_rem		Specifies the remainder for transmit descriptor alignment.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
			System-Dependent Environment Variables
	ferr1		Quadword of error information that Futurebus+ modules can store.
	ferr2		Quadword of error information that Futurebus+ modules can store.
	fis_name		Specifies a string indicating the Factory Installed Software.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
			System-Dependent Environment Variables
	interleave	NV	Specifies the memory interleave configuration for the system. The value must be one of: "default," "none," or an explicit interleave list. The syntax for specifying the configuration is: 0,1,2,3—Indicates the memory module (or slot) numbers. : Indicates that the adjacent memory modules are combined to form a logical module or single interleave unit. + Indicates that the adjacent memory modules or units are to be interleaved, forming a set. , Indicates that the adjacent memory modules, units, or sets are not to be interleaved. For example, assume a system where memory module 0 and 1 are 64 MB each, module 2 is 128 MB, and module 3 is 32 MB. Memory is configured such that module 0 and 1 are combined as a logical unit, 128 MB. This unit is interleaved with module 2, which is also 128 MB to form an interleaved set, 256 MB. Module 3 is not interleaved, but configured as the next 32 MB after the interleave set. set interleave 0:1+2,3 The system is shipped with interleave set to "default". With this value, the optimal interleave configuration for the memory modules will be set. Normally, there is no reason to change the interleave setting.
	mopv3_boot		Specifies whether to use MOP Version 3 format messages first in the boot requests sequence, instead of MOP Version 4.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
			System-Dependent Environment Variables

	ncr*_setup	NV	Here "*" may be 0, 1, 2, 3, or 4, corresponding to the storage bus adapters A, B, C, D, or E, respectively.
			
			Four bus mode parameters are associated with ncr*_setup:
			
			AUTO # — Automatically selects SCSI or DSSI depending on the type of storage device connected to the storage bus (default setting). The node ID for the host storage adapter, usually 7, is represented by #.
			
			DSSI # — Forces storage bus to DSSI. When configuring a DSSI VMScluster system, you should force shared buses to DSSI. The node ID for the host storage adapter—5, 6, or 7 in a DSSI VMScluster system—is represented by #.
			
			SCSI — Forces storage bus to SCSI.
			
			FAST *n* — Forces storage bus to SCSI at fastest rate the devices can support. When using FAST storage mode, you can specify the bus rate, *n*, from 5–12 MB/sec.
			
			In the following example, the bus modes for buses 0 and 1 are forced to DSSI, and the bus mode for bus 2 is forced to FAST SCSI:
			
			`>>> set ncr0_setup "DSSI 7"`
			`>>> set ncr1_setup "DSSI 7"`
			`>>> set ncr2_setup "FAST 5"`
			`>>> show ncr*`
			`ncr0_setup DSSI 7`
			`ncr1_setup DSSI 7`
			`ncr2_setup FAST 5`
			`ncr3_setup AUTO 7`
			`ncr4_setup AUTO 7`
			`>>>`

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
	System-Dependent Environment Variables		

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
	System-Dependent Environment Variables		
	pal	RO	Specifies the versions of OpenVMS and OSF/1 PALcode in the firmware. For instance, OpenVMS PALcode X5.12B, OSF/1 PALcode X1.09A.
	screen_mode	NV	Specifies whether or not the power-up screens or console event log are displayed during power-up. ON (default; FIS process sets to ON)—Displays the two power-up screens during power-up. OFF—Displays the console event log during power-up.
	sys_serial_num	NV	FIS process writes a system serial number to this variable.
	tt_allow_login	NV	Turned off at manufacturing during console loopback testing. 1 (default)—Normal console setting 0—Allows console loopback tests to run
	tta_merge		Ties the console serial port and auxiliary serial port together, so that a customer can monitor remote services. 0 (default)—Console and auxiliary serial ports operate independently. 1—Input entered through the auxiliary port, and output to the auxiliary port, is mirrored on the console port.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

(continued on next page)

Table A–1 (Cont.) Environment Variables

#	Variable	Attributes	Function
			System-Dependent Environment Variables
	tta*_baud	NV	Here "*" may be 0 or 1, corresponding to the primary console serial port, tta0 or the auxiliary console serial port, tta1. Specifies the baud rate of the primary console serial port, tta0. Allowable values are 600, 1200, 2400, 4800, 9600, and 19200. The initial value for tta0 is read from the baud rate select switch on the OCP.
	tta*_halts	NV	Specifies halt characters recognized on the console serial ports, tta0 and tta1. The value is an integer bitmap, where: bit 0—Enables (1) or disables (0) Ctrl/P to init from the console. bit 1—Enables (1) or disables (0) Ctrl/P halts from the operating system. bit 2—Enables (1) or disables (0) BREAK/halts from the operating system. Since tta1 is intended for modem support, this bit is ignored on tta1 (BREAK/halts are not permitted on the auxiliary port). The default for tta0 is 2, enabling Ctrl/P halts from the operating system. The default for tta1 is 0, disabling halts from the operating system.
	version	RO	Specifies the version of the console code in the firmware. For instance, V2.3-2001 Aug 21 1992 14:25:19.

Key to variable attributes:

NV - Nonvolatile. The last value saved by system software or set by console commands is preserved across system initializations, cold bootstraps, and long power outages.
W - Warm nonvolatile. The last value set by system software is preserved across warm bootstraps and restarts.
RO - Read-only. The variable cannot be modified by system software or console commands.

B

Power System Controller Fault Displays

The microprocessor in the PSC reports the fault conditions listed in Table B–1 on the Fault ID display.

Table B–1 Power System Controller Fault ID Display

Fault ID Display (Hex)	Meaning	FRU
PSC Self-Test Faults During AC Power-Up		
F + PSC fault LED on	PSC bias supply not okay	PSC
E + PSC fault LED on	ROM checksum invalid	PSC
D + PSC fault LED on	Port FF20 (PSC/FEU LEDs) 00/FF test failed	PSC
C + PSC fault LED on	Port FF23 (DC–DC LEDs) 00/FF test failed	PSC
B + PSC fault LED on	Port FF24 (LDC enable) not initially 00	PSC
A + PSC fault LED on	Port FF22 (module enables) not initially 00	PSC
9 + PSC fault LED on	Port FF28 (OV/UV status) 00/AA test failed	PSC
8 + PSC fault LED on	External RAM test failed	PSC
7 + PSC fault LED on	80C196 internal RAM test failed	PSC
6 + PSC fault LED on		PSC
5 + PSC fault LED on	80C196 arithmetic test failed	PSC
4 + PSC fault LED on	8259 (external interrupt controller) test failed	PSC
3 + PSC fault LED on	8584 registers did not program correctly	PSC
2 + PSC fault LED on	Temperature sensor bad—low reading	PSC
1 + PSC fault LED on	Temperature sensor bad—high reading	PSC
0 + Overtemperature shutdown Led on	System shutdown (red zone)	Air block

(continued on next page)

Table B–1 (Cont.) Power System Controller Fault ID Display

Fault ID Display (Hex)	Meaning	FRU
PSC Self-Test Faults During AC Power-Up		
0	Normal, PSC passed AC power on	

(continued on next page)

Table B–1 (Cont.) Power System Controller Fault ID Display

Fault ID Display (Hex)	Meaning	FRU
PSC Module Faults		
F + PSC fault LED on	PSC bias supply failed (NMI occurred)	PSC
F + PSC fault LED on	Unimplemented opcode interrupt occurred (invalid instruction)	PSC
F + PSC fault LED on	Software trap interrupt occurred (F7 instruction executed)	PSC
EFFF + PSC fault LED on	Invalid error number (in display_error procedure)	PSC
E000 + PSC fault LED on	Unused error condition	PSC
E012 + PSC fault LED on	Masked interrupt occurred (A/D conversion complete)	PSC
E013 + PSC fault LED on	Masked interrupt occurred (HSI data available)	PSC
E014 + PSC fault LED on	Masked interrupt occurred (HSO)	PSC
E015 + PSC fault LED on	Masked interrupt occurred (HSI pin 0)	PSC
E016 + PSC fault LED on	Masked interrupt occurred (serial I/O)	PSC
E019 + PSC fault LED on	Masked interrupt occurred (HSI FIFO fourth entry)	PSC
E020 + PSC fault LED on	Masked interrupt occurred (Timer 2 capture)	PSC
E021 + PSC fault LED on	Masked interrupt occurred (Timer 2 overflow)	PSC
E023 + PSC fault LED on	Invalid interrupt number (> 31) received from 8259	PSC
E024 + PSC fault LED on	IRQ4 occurred (slave 0 to master 8259)	PSC
E025 + PSC fault LED on	IRQ5 occurred (slave 1 to master 8259)	PSC
E026 + PSC fault LED on	IRQ6 occurred (slave 2 to master 8259)	PSC
E027 + PSC fault LED on	Masked IRQ13 occurred (FEU DIRECT 48 became okay)	PSC
E028 + PSC fault LED on	Masked IRQ14 occurred (FEU SWITCHED 48 became okay)	PSC

(continued on next page)

Table B–1 (Cont.) Power System Controller Fault ID Display

Fault ID Display (Hex)	Meaning	FRU
PSC Module Faults		
E029 + PSC fault LED on	Masked IRQ16 occurred (FEU POWER became okay)	PSC
E030 + PSC fault LED on	Masked IRQ29 occurred (unused FEU signal)	PSC
E031 + PSC fault LED on	Masked IRQ30 occurred (unused FEU signal)	PSC
E032 + PSC fault LED on	Masked IRQ25 occurred (OCP DC ON—turned on)	PSC
E033 + PSC fault LED on	Masked IRQ26 occurred (PSC DC ON—turned on)	PSC
E034 + PSC fault LED on	Invalid converter number (start of enable_ converter procedure)	PSC
E035 + PSC fault LED on	Invalid converter number (end of enable_ converter procedure)	PSC
E036 + PSC fault LED on	Invalid converter number (start of disable_ converter procedure)	PSC
E037 + PSC fault LED on	Invalid converter number (end of disable_ converter procedure)	PSC
E040 + PSC fault LED on	PSC 8584 self-address register did not program	PSC
E041 + PSC fault LED on	PSC 8584 clock register did not program	PSC
E042 + PSC fault LED on	PSC 8584 interrupt vector register did not program	PSC
E043 + PSC fault LED on	PSC 8584 control register did not program	PSC

(continued on next page)

Table B–1 (Cont.) Power System Controller Fault ID Display

Fault ID Display (Hex)	Meaning	FRU
DC–DC Converter Faults		
E100	Delta overvoltage fail between +5v and +3v converters	DC5, DC3
E110	2.1V converter—out of regulation, low	DC3
E111	2.1V converter—out of regulation, high	DC3
E112	2.1V converter—under voltage	DC3
E113	2.1V converter—over voltage	DC3
E114	2.1V converter—voltage present when disabled	DC3
E115	2.1V converter—did not turn off	DC3
E120	3.3V converter—out of regulation, low	DC3
E121	3.3V converter—out of regulation, high	DC3
E122	3.3V converter—under voltage	DC3
E123	3.3V converter—over voltage	DC3
E124	3.3V converter—voltage present when disabled	DC3
E125	3.3V converter—did not turn off	DC3
E130	5.0V converter—out of regulation, low	DC5
E131	5.0V converter—out of regulation, high	DC5
E132	5.0V converter—under voltage	DC5
E133	5.0V converter—over voltage	DC5
E134	5.0V converter—voltage present when disabled	DC5
E135	5.0V converter—did not turn off	DC5
E140	12V converter—out of regulation, low	DC3
E141	12V converter—out of regulation, high	DC3
E142	12V converter—under voltage	DC3
E143	12V converter—over voltage	DC3
E144	12V converter—voltage present when disabled	DC3
E145	12V converter—did not turn off	DC3

(continued on next page)

Table B–1 (Cont.) Power System Controller Fault ID Display

Fault ID Display (Hex)	Meaning	FRU
FEU Module Faults		
E200	SWITCHED 48 okay before enabling	FEU
E201	Fan converter operating before enabling	FEU
E202	HVDC is okay, but POWER is not okay (contradictory status)	FEU
E204	DIRECT 48 not okay and POWER is okay (IRQ18)	FEU
E205	SWITCHED 48 not okay and switched bus requested (IRQ19)	FEU
E206	HVDC is okay, but POWER is not okay (IRQ20)	FEU
E210	SWITCHED BUS did not turn on a startup	FEU
E211	SWITCHED BUS did not turn off at power down	FEU
E220	Fan converter voltage is low	FEU
Fan, LDC, and Temperature Faults		
1 + Fan Failure LED on	Fan 1 failed	Fan 1
2 + Fan Failure LED on	Fan 2 failed	Fan 2
3 + Fan Failure LED on	Fan 3 failed	Fan 3
4 + Fan Failure LED on	Fan 4 failed	Fan 4
9 + Fan Failure LED On	Cable guide is not secured or 2 fans failed	
A + Disk Power Failure LED on	LDC A failed	LDC A
B + Disk Power Failure LED on	LDC B failed	LDC B
C + Disk Power Failure LED on	LDC C failed	LDC C
D + Disk Power Failure LED on	LDC D failed	LDC D
7 + PSC Failure LED on	Temperature sensor bad—low reading	PSC
8 + PSC Failure LED on	Temperature sensor bad—high reading	PSC
0 + Overtemperature shutdown LED on	System temperature in red zone	

C

Worksheet for Recording Customer Environment Variable Settings

When replacing the I/O module, use Table C–1 to record the customer's nonvolatile environment variable settings. After you install the new I/O module, you can restore the customer's settings.

Table C–1 Nonvolatile Environment Variables

Environment Variable	Factory Default	Customer Setting
auto_action	BOOT	
bootdef_dev	Null (FIS process defines device with operating system)	
boot_file	Null	
boot_osflags	Null	
boot_reset	OFF	
char_set	0	
cpu_enabled	OxFF (all processors present enabled)	
def_term	LOCAL	
dump_dev	ON	
enable_audit	ON	
enable_servers	OFF	
ez*0_arp_tries	3	
ez*0_bootp_file	Null	

(continued on next page)

Table C–1 (Cont.) Nonvolatile Environment Variables

Environment Variable	Factory Default	Customer Setting
ez*0_bootp_server	Null	
ez*0_bootp_tries	3	
ez*0_def_inetaddr	Null	
ez*0_def_inetfile	Null	
ez*0_def_ginetaddr	Null	
ez*0_def_sinetaddr	Null	
ez*0_inet_init	BOOTP	
ez*0_protocols	MOP	
ez*0_rm_boot	0 or disable	
ez*0_rm_boot_ passwd	00_00000000_00000000.	
ez*0_tftp_tries	3	
fis_name	Null	
interleave	Default	
language	36 English	
ncr*_setup	AUTO 7	
password	Null	
screen_mode	Off (FIS process sets to on)	
scsnode	Null	
scssystemid	65534	
scssystemidh	0	
sys_serial_num	Null (FIS process writes system serial #)	
tta_merge	0	
tta*_baud	9600	
tta*_halts	2 for tta2; 0 for tta1	
tt_allow_login	1	

Glossary

arbiter

The entity responsible for controlling a bus—it controls bus mastership.

assert

To cause a signal to change to its logical true state.

autoboot

The process by which the system boots automatically.

auxiliary serial port

The EIA 232 serial port on the I/O module of the DEC 4000 AXP system. This port provides asynchronous communication with a device, such as a modem.

availability

The amount of scheduled time that a computing system provides application service during the year. Availability is typically measured as either a percentage of "uptime" per year or as system "unavailability," the number of hours or minutes of downtime per year.

BA640

The enclosure that houses the DEC 4000 AXP system. The BA640 is compatible with the departmental environment and is designed for maximum flexibility in system configuration. Employing an open system architecture, the BA640 incorporates a state-of-the-art Futurebus+ area, which allows for expansion of the DEC 4000 AXP system with options available from Digital and other vendors.

backplane

The main circuit board or panel that connects all of the modules in the system. In desktop systems, the backplane is analogous to the motherboard.

backup cache

A second, very fast memory that is used in combination with slower large-capacity memories.

bandwidth

Bandwidth is often used to express "high rate of data transfer" in an I/O channel. This usage assumes that a wide bandwidth may contain a high frequency, which can accommodate a high rate of data transfer.

baud rate

The speed at which data is transmitted over a serial data line; baud rates are measured in bits per second.

bit

Binary digit. The smallest unit of data in a binary notation system, designated as 0 or 1.

BIU

See bus interface unit.

block exchange

Memory feature that improves bus bandwidth by paralleling a cache victim write-back with a cache miss fill.

boot

Short for bootstrap. Loading an operating system into memory is called booting.

bootblock

The first logical block on the boot device. It contains information about the location of the primary bootstrap on the device.

boot device

The device from which the system bootstrap software is acquired.

boot flags

Boot flags contain information that is read and used by the bootstrap software during a system bootstrap procedure.

boot primitives

Device handler routines that read the bootblock and, subsequently, the primary bootstrap program, into memory from the boot device. *See also* bootblock.

boot server

A system that provides boot services to remote devices such as network routers and VAXcluster satellite nodes.

bootstrap

See boot.

buffer

An internal memory area used for temporary storage of data records during input or output operations.

bugcheck

A software condition, usually the response to software's detection of an "internal inconsistency," which results in the execution of the system bugcheck code.

bus

A group of signals that consists of many transmission lines or wires. It interconnects computer system components to provide communications paths for addresses, data, and control information.

bus interface unit

Logic designed to interface internal logic, a module or a chip, to a bus.

bystander

A system bus node that is not addressed by a current system bus commander transaction address.

byte

Eight contiguous bits starting on an addressable byte boundary. The bits are numbered right to left, 0 through 7.

byte granularity

Memory systems are said to have byte granularity if adjacent bytes can be written concurrently and independently by different processes or processors.

C^3 chip

An acronym for command, control, and communication chip. On the DEC 4000 AXP system, the ASIC gate array chip located on the CPU module. This chip contains CPU command, control, and communication logic, as well as the bus interface unit for the processor module.

cache

See cache memory.

cache block

The fundamental unit of manipulation in a cache. Also known as cache line.

cache interference

The result of an operation that adversely affects the mechanisms and procedures used to keep frequently used items in a cache. Such interference may cause frequently used items to be removed from a cache or incur significant overhead operations to ensure correct results. Either action hampers performance.

cache line

The fundamental unit of manipulation in a cache. Also known as cache block.

cache memory

A small, high-speed memory placed between slower main memory and the processor. A cache increases effective memory transfer rates and processor speed. It contains copies of data recently used by the processor and fetches several bytes of data from memory in anticipation that the processor will access the next sequential series of bytes.

card cage

A mechanical assembly in the shape of a frame that holds modules against the system and storage backplanes.

CD–ROM

Compact disc read-only memory. The optical removable media used in a compact disc reader mass storage device.

central processing unit (CPU)

The unit of the computer that is responsible for interpreting and executing instructions.

channel

A path along which digital information can flow in a computer.

checksum

A sum of digits or bits that is used to verify the integrity of a piece of data.

CI

See computer interconnect.

CISC

Complex instruction set computer. An instruction set consisting of a large number of complex instructions that are managed by microcode. *Contrast with* RISC.

clean

In the cache of a system bus node, refers to a cache line that is valid but has not been written.

client-server computing

An approach to computing that enables personal computer and workstation users—the "client"—to work cooperatively with software programs stored on a mainframe or minicomputer—the "server."

clock

A signal used to synchronize the circuits in a computer system.

cluster

A group of systems and hardware that communicate over a common interface. *See also* VMScluster system.

CMOS

Complementary metal-oxide semiconductor. A silicon device formed by a process that combines PMOS and NMOS semiconductor material.

cold bootstrap

A bootstrap operation following a power-up condition or system initialization (restart).

command

A field of the system bus address and command cycle (cycle 1), which encodes the transaction type.

commander

A system bus node that participates in arbitration and initiates a transaction. Also called a commander node.

concurrency

Simultaneous operations by multiple agents on a shared object.

conditional invalidation

Invalidation of a cached location based upon a set of conditions, which are the state of other caches, or the source of the information causing the invalidate.

console mode

The state in which the system and the console terminal operate under the control of the console program.

console program

The code that the CPU executes during console mode.

console subsystem

The subsystem that provides the user interface for a system when operating system software is not running. The console subsystem consists of the following components:

console program
console terminal
console terminal port
remote access device
remote access port
Ethernet ports

console terminal

The terminal connected to the console subsystem. The console is used to start the system and direct activities between the computer operator and the computer system.

console terminal port

The connector to which the console terminal cable is attached.

control and status register (CSR)

A device or controller register that resides in the processor's I/O space. The CSR initiates device activity and records its status.

CPU

See central processing unit.

CSR

See control and status register.

cycle

One clock interval.

data alignment

An attribute of a data item that refers to its placement in memory (therefore its address).

data bus

A bus used to carry signals between two or more components of the system.

D-bus

On the DEC 4000 AXP system, the bus between the 21064 CPU chip and the "D-bus micro" and the serial ROMs.

D-cache

Data cache. A high-speed memory reserved for the storage of data. *Contrast with* I-cache.

DC-DC converter

A device that converts one DC voltage to another DC voltage.

deassert

To cause a signal to change to its logical false state.

DECchip 21064 microprocessor

The CMOS-4, Alpha AXP architecture, single-chip processor used on Alpha AXP based computers.

DECnet

Networking software designed and developed by Digital. DECnet is an implementation of the Digital Network Architecture.

DEC OSF/1 operating system

A general-purpose operating system based on the Open Software Foundation OSF/1 1.0 technology. DEC OSF/1 V1.2 runs on the range of Alpha AXP systems, from workstations to servers.

DEC VET

Digital's DEC Verifier and Exerciser Tool. DEC VET is a multipurpose system maintenance tool that performs exerciser-oriented maintenance testing.

direct-mapping cache

A cache organization in which only one address comparison is needed to locate any data in the cache, because any block of main memory data can be placed in only one possible position in the cache.

direct memory access (DMA)

Access to memory by an I/O device that does not require processor intervention.

dirty

Used in reference to a cache block in the cache of a system bus node. The cache block is valid and has been written so that it differs from the copy in system memory.

disk fragmentation

The writing of files in noncontiguous areas on a disk. Fragmentation can cause slower system performance because of repeated read or write operations on fragmented data.

disk mirroring

See volume shadowing.

distributed processing

A processing configuration in which each processor has its own autonomous operating environment. The processors are not tightly coupled and globally controlled as they are with multiprocessing. A distributed processing environment can include multiprocessor systems, uniprocessor systems, and cluster systems. It entails the distribution of an application over more than one system. The application must have the ability to coordinate its activity over a dispersed operating environment. *Contrast with* symmetric multiprocessing.

DRAM

Dynamic random-access memory. Read/write memory that must be refreshed (read from or written to) periodically to maintain the storage of information.

DSSI

Digital's proprietary data bus that uses the System Communication Architecture (SCA) protocols for direct host-to-storage communications.

DSSI VMScluster

A VMScluster system that uses the DSSI bus as the interconnect between DSSI disks and systems.

DUP server

The Diagnostic Utility Program (DUP) server is a firmware program on-board DSSI devices that allows a user to set host to a specified device in order to run internal tests or modify device parameters.

ECC

Error correction code. Code and algorithms used by logic to facilitiate error detection and correction. *See also* ECC error; EDC logic.

ECC error

An error detected by EDC logic, to indicate that data (or the protected "entity" has been corrupted. The error may be correctable (ECC error) or uncorrectable (ECCU error). *See also* EDC logic.

EDC logic

Error detection and correction logic. Used to detect and correct errors. *See also* ECC; ECC error.

EEPROM

Electrically erasable programmable read-only memory. A memory device that can be byte-erased, written to, and read from. *Contrast with* FEPROM.

environment variable

Global data structures that can be accessed from console mode. The setting of these data structures determines how a system powers up, boots operating system software, and operates.

Ethernet

A local area network that was originally developed by Xerox Corporation and has become the IEEE 802.3 standard LAN. Ethernet LANs use bus topology.

Ethernet ports

The connectors through which the Ethernet is connected to the system.

extents

The physical locations in a storage device allocated for use by a particular data set.

Factory Installed Software (FIS)

Operating system software that is loaded into a system disk during manufacture. On site, the FIS is bootstrapped in the system, prompting a predefined menu of questions on the final configuration.

fast SCSI

An optional mode of SCSI-2 that allows transmission rates of up to 10 MB/s. *See also* SCSI.

FDDI

Fiber Distributed Data Interface. A high-speed networking technology that uses fiber optics as the transmissions medium.

FEPROM

Flash-erasable programmable read-only memory. FEPROMs can be bank- or bulk-erased. *Contrast with* EEPROM.

FIS

See Factory Installed Software.

firmware

Software code stored in hardware.

fixed-media compartments

Compartments that house nonremovable storage media.

front end unit (FEU)

One of four modules in the DEC 4000 AXP system power supply. The FEU converts alternating current from a wall plug to 48 VDC that the rest of the power subsystem can use and convert.

FRU

Field-replaceable unit. Any system component that the service engineer is able to replace on-site.

full-height device

Standard form factor for 5 1/4-inch storage devices.

Futurebus+

A computer bus architecture that provides performance scalable over both time and cost. It is the IEEE 896 open standard.

Futurebus+ Profile B

A profile is a specification that calls out a subset of functions from a larger specification. Profile B satisfies the requirements for an I/O bus. *See also* Futurebus+.

half-height device

Standard form factor for storage devices that are not the height of full-height devices.

halt

The action of transferring control to the console program.

hard error

An error that has induced a nonrecoverable failure in a system.

hexword

Short for "hexadecimalword." Thirty-two contiguous bytes (256 bits) starting on an addressable byte boundary. Bits are numbered from right to left, 0 through 255.

I-cache

Instruction cache. A high-speed memory reserved for the storage of instructions. *Contrast with* D-cache.

initialization

The sequence of steps that prepare the system to start. Initialization occurs after a system has been powered up.

interleaving

See memory interleaving.

internal processor register (IPR)

A register internal to the CPU chip.

KN430 CPU

The CPU module used by DEC 4000 AXP Model 600 series systems. The KN430 CPU modeule is based on the DECchip 21064 microprocessor.

LAN (local area network)

A network that supports servers, PCs, printers, minicomputers, and mainframe computers that are connected over limited distances.

latency

The amount of time it takes the system to respond to an event.

LDC

See local disk converter.

LED

Light-emitting diode. A semiconductor device that glows when supplied with voltage.

local disk converter (LDC)

Refers to modules that regulate voltages for fixed-media storage devices. An LDC module is located in each of the fixed-media storage compartments (A–D), provided that the compartment is not storageless.

longword

Four contiguous bytes starting on an arbitrary byte boundary. The bits are numbered from right to left, 0 through 31.

loopback tests

Diagnostic tests used to isolate a failure by testing segments of a particular control or data path.

low-level language

Any language that exposes the details of the hardware implementation to the programmer. Typically this refers to assembly languages that allow direct hardware manipulation. *See also* high-level language.

machine check/interrupts

An operating system action triggered by certain system hardware-detected errors that can be fatal to system operation. Once triggered, machine check handler software analyzes the error.

mailbox

A memory data structure used to communicate between different components of the system.

masked write

A write cycle that only updates a subset of a nominal data block.

mass storage device

An input/output device on which data is stored. Typical mass storage devices include disks, magnetic tapes, and floppy disks.

memory interleaving

The process of assigning consecutive physical memory addresses across multiple memory controllers. Improves total memory bandwidth by overlapping system bus command execution across two or four memory modules.

MIPS

Millions of instructions per second.

MOP

Maintenance Operations Protocol. The transport protocol for network bootstraps and other network operations.

multiplex

To transmit several messages or signals simultaneously on the same circuit or channel.

multiprocessing system

A system that executes multiple tasks simultaneously.

NAS

See Network Applications Support.

Network Applications Support

A comprehensive set of software supplied by Digital Equipment Corporation that enables application integration across a distributed multivendor environment. NAS consists of well-defined programming interfaces, toolkits, and products that help developers build applications that are well-integrated and more easily portable across different systems.

node

A device that has an address on, is connected to, and is able to communicate with other devices on the bus. In a computer network, an individual computer system connected to the network that can communicate with other systems on the network.

NVRAM

Nonvolatile random-access memory. Memory that retains its information in the absence of power such as magnetic tape, drum, or core memory.

octaword

Sixteen contiguous bytes starting on an arbitrary byte boundary. The bits are numbered from right to left, 0 through 127.

open system

A system that implements sufficient open specifications for interfaces, services, and supporting formats to enable applications software to:

- Be ported across a wide range of systems with minimal changes
- Interoperate with other applications on local and remote systems
- Interact with users in a style that facilitates user portability

OpenVMS AXP operating system

Digital's open version of the VMS operating system, which runs on Alpha AXP machines. *See also* open system.

operand

The data or register upon which an operation is performed.

operator control panel

The panel on the top right side of the DEC 4000 AXP system that contains the power, Reset, and Halt switches and system status lights.

page size

A number of bytes, aligned on an address evenly divisible by that number, which a system's hardware treats as a unit for virtual address mapping, sharing, protection, and movement to and from secondary storage.

PAL

Programmable array logic (hardware), a device that can be programmed by a process that blows individual fuses to create a circuit.

PALcode

Alpha AXP Privileged Architecture Library code, written to support Alpha AXP processors. PALcode implements architecturally defined behavior.

parity

A method for checking the accuracy of data by calculating the sum of the number of ones in a piece of binary data. Even parity requires the correct sum to be an even number, odd parity requires the correct sum to be an odd number.

pipeline

A CPU design technique whereby multiple instructions are simultaneously overlapped in execution.

portability

Degree to which a software application can be easily moved from one computing environment to another.

porting

Adapting a given body of code so that it will provide equivalent functions in a computing environment that differs from the original implementation environment.

power-down

The sequence of steps that stops the flow of electricity to a system or its components.

power system controller (PSC)

One of four units in the DEC 4000 AXP power supply subsystem. The H7851AA PSC monitors signals from the rest of the system including temperature, fan rotation, and DC voltages, as well as provides power-up and power-down sequencing to the DC-DC converters and communicates with the system CPU across the serial control bus.

power-up

The sequence of events that starts the flow of electrical current to a system or its components.

primary cache

The cache that is the fastest and closest to the processor.

processor corrected machine check

Processor machine checks indicate that a processor B-cache error was detected and successfully corrected by hardware or PALcode. Examples of processor correctable machine check conditions include corrected processor B-cache errors.

processor machine check

Processor machine checks indicate that a processor internal error was detected synchronously to the processors execution and was not successfully corrected by hardware or PALcode. Examples of processor machine check conditions include processor B-cache buffer parity errors, memory uncorrectable errors, or read access to a nonexistent location.

processor module

Module that contains the CPU chip.

program counter

That portion of the CPU that contains the virtual address of the next instruction to be executed. Most current CPUs implement the program counter (PC) as a register. This register may be visible to the programmer through the instruction set.

program mode

See operating system mode.

quadword

Eight contiguous bytes starting on an arbitrary byte boundary. The bits are numbered from right to left, 0 through 63.

R400X mass storage expander

A Digital enclosure used for mass storage expansion.

RAID

Redundant array of inexpensive disks. A technique that organizes disk data to improve performance and reliability. RAID has three attributes:

1. It is a set of physical disks viewed by the user as a single logical device.
2. The user's data is distributed across the physical set of drives in a defined manner.
3. Redundant disk capacity is added so that the user's data can be recovered even if a drive fails.

Contrast with striping.

read data wrapping

Memory feature that reduces apparent memory latency by allowing octawords within a selected hexword block to be accessed in reverse order.

read-merge

Indicates that an item is read from a responder/bystander, and new data is then added to the returned read data. This occurs when a masked write cycle is requested by the processor or when unmasked cycles occur and the CPU is configured to allocate on full block write misses.

read-modify-write operation

A hardware operation that involves the reading, modifying, and writing of a piece of data in main memory as a single, uninterruptible operation.

read stream buffers

Arrangement whereby each memory module independently prefetches DRAM data prior to an actual read request for that data. Reduces average memory latency while improving total memory bandwidth.

read-write ordering

Refers to the order in which memory on one CPU becomes visible to an execution agent (a different CPU or device within a tightly coupled system).

redundant

Describes duplicate or extra computing components that protect a computing system from failure.

register

A temporary storage or control location in hardware logic.

reliability

The probability a device or system will not fail to perform its intended functions during a specified time interval when operated under stated conditions.

removable-media compartment

Compartment in the enclosure that houses removable media.

responder

A system bus node that accepts or supplies data in response to an address and command from a system bus commander. Also called a responder node.

RISC

Reduced instruction set computer. A computer with an instruction set that is reduced in complexity.

robust mode

A power-up mode (baud rate select switch set to 0) that allows you to power up without initializing drivers or running power-up diagnostics. The console program has limited functionality in robust mode.

ROM-based diagnostics

Diagnostic programs resident in read-only memory. ROM-based diagnostics are the primary means of console mode testing and diagnosis of the CPU, memory, Ethernet, Futurebus+, SCSI, and DSSI subsystems.

script

A data structure that defines a group of commands to be executed. Simalar to a command file.

SCSI

Small Computer System Interface. An ANSI-standard interface for connecting disks and other peripheral devices to computer systems. *See also* fast SCSI.

SDD

See symptom-directed diagnostics.

self-test

A test that is invoked automatically when the system powers up.

serial control bus

A two-conductor serial interconnect that is independent of the system bus. This bus links the processor modules, the I/O, the memory, the power subsystem, and the operator control panel. It reports any failed devices to the processor module so the processor module can illuminate LEDs on the operator control panel.

shadowing

See volume shadowing.

shadow set

In volume shadowing, the set of disks on which the data is duplicated. Access to a shadow set is achieved by means of a virtual disk unit. After a shadow set is created, applications and users access the virtual disk unit as if it were a physical disk. *See also* volume shadowing.

SMP

See symmetric multiprocessing.

snooping protocol

A cache coherence protocol whereby all nodes on a common system bus monitor all bus activity. This allows a node to keep its copy of a particular datum up-to-date and/or supply data to the bus when it has the newest copy.

SROM

Serial read-only memory.

stack

An area of memory set aside for temporary data storage or for procedure and interrupt service linkages. A stack uses the last-in/first-out concept. As items are added to (pushed on) the stack, the stack pointer decrements. As items are retrieved from (popped off) the stack, the stack pointer increments.

storage assembly

All the components necessary to configure storage devices into a DEC 4000 AXP storage compartment. These components include the storage device, brackets, screws, shock absorbers, and cabling.

storage backplane

One of two backplanes in the BA640 enclosure. Fixed and removable media devices plug into this backplane. *See also* backplane.

stripe set

A group of physical disks that are used for disk striping. *See also* striping.

striping

A storage option that increases I/O performance. With disk striping, a single file is split between multiple physical disks. Read and write disk performance is increased by sharing input/output operations between multiple spindles, which allows an I/O rate greater than that of any one disk member of the stripe set. In striping, the loss of any one member of the stripe set causes loss of the set. Striping is particularly useful for applications that move large amounts of disk-based information, for example, graphic imaging. *Contrast with* RAID.

superscalar

Describes a machine that issues multiple independent instructions per clock cycle.

symmetric multiprocessing (SMP)

A processing configuration in which multiple processors in a system operate as equals, dividing and sharing the workload. OpenVMS SMP provides two forms of multiprocessing: multiple processes can execute simultaneously on different CPUs, thereby maximizing overall system performance; and single-stream application programs can be partitioned into multistream jobs, minimizing the processing time for a particular program. *Contrast with* distributed processing.

symptom-directed diagnostics (SDD)

Online analysis of system errors to locate potential system faults. SDD helps isolate system problems.

synchronization

A method of controlling access to some shared resource so that predictable, well-defined results are obtained when operating in a multiprocessing environment.

system backplane

One of two backplanes in the BA640 enclosure. CPU, memory, I/O, Futurebus+, and power modules plug into this backplane. *See also* backplane.

system bus

The private interconnect used on the DEC 4000 AXP CPU subsystem. This bus connects the B2001 processor module, the B2002 memory module, and the B2101 I/O module.

system disk

The device on which operating system software resides.

system fatal error

An error that is fatal to the system operation, because the error occurred in the context of a system process or the context of an error cannot be determined.

system machine check

System machine checks are generated by error conditions that are detected asynchronously to processor execution. Examples of system machine check conditions include protocol errors on the processor-memory interconnect, unrecoverable memory errors detected by the I/O module or other CPU, and memory correctable errors.

TCP/IP

Transmission Control Protocol/Internet Protocol. A set of software communications protocols widely used in UNIX operating environments. TCP delivers data over a connection between applications on different computers on a network; IP controls how packets (units of data) are transferred between computers on a network.

thickwire

An IEEE standard 802.3-compliant Ethernet network made of standard Ethernet cable, as opposed to ThinWire Ethernet cable. Also called standard Ethernet. *Contrast with* ThinWire.

ThinWire

Digital's proprietary Ethernet products used for local distribution of data communications. *Contrast with* thickwire.

UETP

User Environment Test Package. An OpenVMS AXP software package designed to test whether the OpenVMS operating system is installed correctly. UETP puts the system through a series of tests that simulate a typical user environment, by making demands on the system that are similar to demands that might occur in everyday use.

uninterruptible power supply (UPS)

A battery-backup option that maintains AC power if a power failure occurs.

unmasked write

In memory, a write cycle that updates all locations of a nominal data block. That is, a hexword update to a cache block.

UPS

See uninterruptible power supply.

VMScluster system

A highly integrated organization of Digital's VMS systems that communicate over a high-speed communications path. VMScluster configurations have all the functions of single-node systems, plus the ability to share CPU resources, queues, and disk storage.

volume shadowing

The process of maintaining multiple copies of the same data on two or more disk volumes. When data is recorded on more than one disk volume, you have access to critical data even when one volume is unavailable. Also called disk mirroring.

Vterm module

The module located behind the OCP that provides the termination voltages for storage bus E. The Vterm module also contains the logic for reporting SCSI continuity card errors.

warm bootstrap

A subset of the cold bootstrap operations: during a warm bootstrap, the console does not load PALcode, size memory, or initialize environment variables.

warm swap

The shutdown and removal and replacement of a failing DSSI disk from an active bus.

word

Two contiguous bytes (16 bits) starting on an arbitrary byte boundary. The bits are numbered from right to left, 0 through 15.

write back

A cache management technique in which data from a write operation to cache is written into main memory only when the data in cache must be overwritten. This results in temporary inconsistencies between cache and main memory. *Contrast with* write through.

write-enabled

A device is write-enabled when data can be written to it. *Contrast with* write-protected.

write-protected

A device is write-protected when transfers are prevented from writing information to it. *Contrast with* write-enabled.

write through

A cache management technique in which data from a write operation is copied to both cache and main memory. Cache and main memory data is always consistent. *Contrast with* write back.

Index

D

E

F

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Reader's Comments

DEC 4000 AXP
Service Guide

EK–KN430–SV. B01

Your comments and suggestions help us improve the quality of our publications.

Please rate the manual in the following categories:	Excellent	Good	Fair	Poor
Accuracy (product works as described)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Table of contents (ability to find topic)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page design (overall appearance)	☐	☐	☐	☐
Print quality	☐	☐	☐	☐

What I like best about this manual: _____

_____

What I like least about this manual: _____

_____

Additional comments or suggestions: _____

_____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____

For which tasks did you use this manual?

☐ Installation
☐ Maintenance
☐ Marketing
☐ Operation/Use
☐ Programming
☐ System Management
☐ Training
☐ Other (please specify) _____

Name/Title _____

Company _____

Address _____

Do Not Tear – Fold Here and Tape

digital

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
INFORMATION DESIGN AND CONSULTING
PKO3–1/D30
129 PARKER STREET
MAYNARD, MA 01754–9975

Do Not Tear – Fold Here and Tape