COMPAQ

AlphaServer GS80/160/320

Firmware Reference Manual

Order Number: EK-GS320-RM. C01

This manual is for users and field service engineers for *Compaq AlphaServer* GS80/160/320 systems. It describes the firmware interface to these systems.

Compaq Computer Corporation

Revised September 2000

FCC Notice

This equipment 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 digital device pursuant to Part 15 of FCC rules, which are designed to provide reasonable protection against such radio frequency interference.

Operation of this equipment in a residential area may cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Any modifications to this device—unless expressly approved by the manufacturer—can void the user's authority to operate this equipment under part 15 of the FCC rules.

Modifications

The FCC requires the user to be notified that any changes or modifications made to this device that are not expressly approved by Compaq Computer Corporation may void the user's authority to operate the equipment.

Cables

Connections to this device must be made with shielded cables with metallic RFI/EMI connector hoods in order to maintain compliance with FCC Rules and Regulations.

Taiwanese Notice

警告使用者:

這是甲類的資訊產品，在居住的
環境中使用時，可能會造成射頻
干擾，在這種情況下，使用者會
被要求採取某些適當的對策。

Japanese Notice

に基づくクラスＡ情報技術装置です。この装置を家庭環境で使用すると電波妨害を引き起こすことがあります。この場合には使用者が適切な対策を講ずるよう要求されることがあります。

Canadian Notice

This Class A digital apparatus meets all requirements of the Canadian Interference-Causing Equipment Regulations.

Avis Canadien

Cet appareil numérique de la classe A respecte toutes les exigences du Règlement sur le matériel brouilleur du Canada.

European Union Notice

Products with the CE Marking comply with both the EMC Directive (89/336/EEC) and the Low Voltage Directive (73/23/EEC) issued by the Commission of the European Community.

Compliance with these directives implies conformity to the following European Norms (in brackets are the equivalent international standards):

EN55022 (CISPR 22) - Electromagnetic Interference

EN50082-1 (IEC801-2, IEC801-3, IEC801-4) - Electromagnetic Immunity

EN60950 (IEC950) - Product Safety

Warning!

This is a Class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

Achtung!

Dieses ist ein Gerät der Funkstörgrenzwertklasse A. In Wohnbereichen können bei Betrieb dieses Gerätes Rundfunkstörungen auftreten, in welchen Fällen der Benutzer für entsprechende Gegenmaßnahmen verantwortlich ist.

Attention!

Ceci est un produit de Classe A. Dans un environnement domestique, ce produit risque de créer des interférences radioélectriques, il appartiendra alors à l'utilisateur de prendre les mesures spécifiques appropriées.

Contents

Chapter 4 Partitions

Chapter 5 Booting an Operating System

Chapter 6 SCM Command Reference

Chapter 7 SRM Command Reference

Appendix A Boot Options

Appendix B Loadable Firmware Update Utility

Index

Examples

Figures

Tables

Preface

Intended Audience

This manual is intended for users and field service engineers who use the *AlphaServer* GS80/160/320 firmware.

Document Structure

This manual uses a structured documentation design. Topics are organized into small sections for efficient online and printed reference. Each topic begins with an abstract. You can quickly gain a comprehensive overview by reading only the abstract. In conceptual chapters, the next item is an illustration or example, which also provides quick reference. Next is descriptive text.

The reference chapters follow the format of UNIX man pages. This includes syntax, arguments, and options (if appropriate), and an example.

This manual has seven chapters and two appendixes, as follows:

- **Chapter 1, Introduction to the Firmware**, introduces the *AlphaServer* GS80/160/320 firmware and the devices through which it is accessed.
- **Chapter 2, Hardware Overview,** gives an overview of the structure and architecture of the *AlphaServer* GS80/160/320 systems.
- **Chapter 3, Remote Console Operations,** describes the SCM commands that allow remote access, and how to use the firmware from a remote terminal.
- **Chapter 4, Partitions**, describes the SCM commands used to set up hard partitions and the SRM commands used to set up soft partitions.
- **Chapter 5, Booting an Operating System,** tells how to boot an operating system on *AlphaServer* GS80/160/320 systems.
- **Chapter 6, SCM Command Reference,** describes the SCM commands in alphabetical order for easy reference.
- **Chapter 7, SRM Command Reference,** describes the SRM commands in alphabetical order for easy reference.

- **Appendix A, Boot Options,** lists the options used with the SRM **boot** command for the *Tru64 UNIX* and *OpenVMS* operating systems to control various phases of booting.
- **Appendix B, Updating Firmware,** explains how to run the Loadable Firmware Update (LFU) utility.

Documentation Titles

Title	Order Number
AlphaServer GS80/160/320 Documentation Kit	QA-6GAAA-G8
AlphaServer GS80/160/320 User's Guide	EK-GS320-UG
AlphaServer GS80/160/320 Firmware Reference Manual	EK-GS320-RM
AlphaServer GS80/160/320 Getting Started with Partitions	EK-GSPAR-RM
AlphaServer G160/320 Installation Guide	EK-GS320-IN
AlphaServer GS80 Installation Guide	EK-GSR80-IN
AlphaServer GS80/160/320 User Information CD (HTML files)	AG-RKSWB-BE
AlphaServer GS80/160/320 User Information CD (translations)	AG-RLVJA-BE
AlphaServer GS80/160/320 Service Documentation Kit	QA-6GAAB-G8
AlphaServer GS80/160/320 Service Manual	EK-GS320-SV
AlphaServer GS80/160/320 Firmware Reference Manual	EK-GS320-RM
AlphaServer GS80/160/320 Service Information CD	AG-RKSZ*-BE
AlphaServer GS160/320 Upgrade Manual	EK-GS320-UP
AlphaServer GS80/160/320 System Management Console Installation and User's Guide	EK-GSCON-IN
AlphaServer GS80 Upgrade Manual	EK-GSR80-UP
AlphaServer GS80/160/320 Site Preparation	EK-GS320-SP

Related Software Documentation

Title	Order Number
Tru64 UNIX Installation Guide	AA-RH8SC-TE
Tru64 UNIX System Management	AA-RH9FC-TE
OpenVMS Alpha Version 7.2-1H1 New Features and Release Notes	AA-RLMUA-TE
OpenVMS Alpha Galaxy and Partitioning Guide	AA-REZQC-TE

Information on the Internet

Visit the Compaq Web site at http://www.compaq.com/site_index.html for service tools and more information about the system.

Chapter 1
Introduction to the Firmware

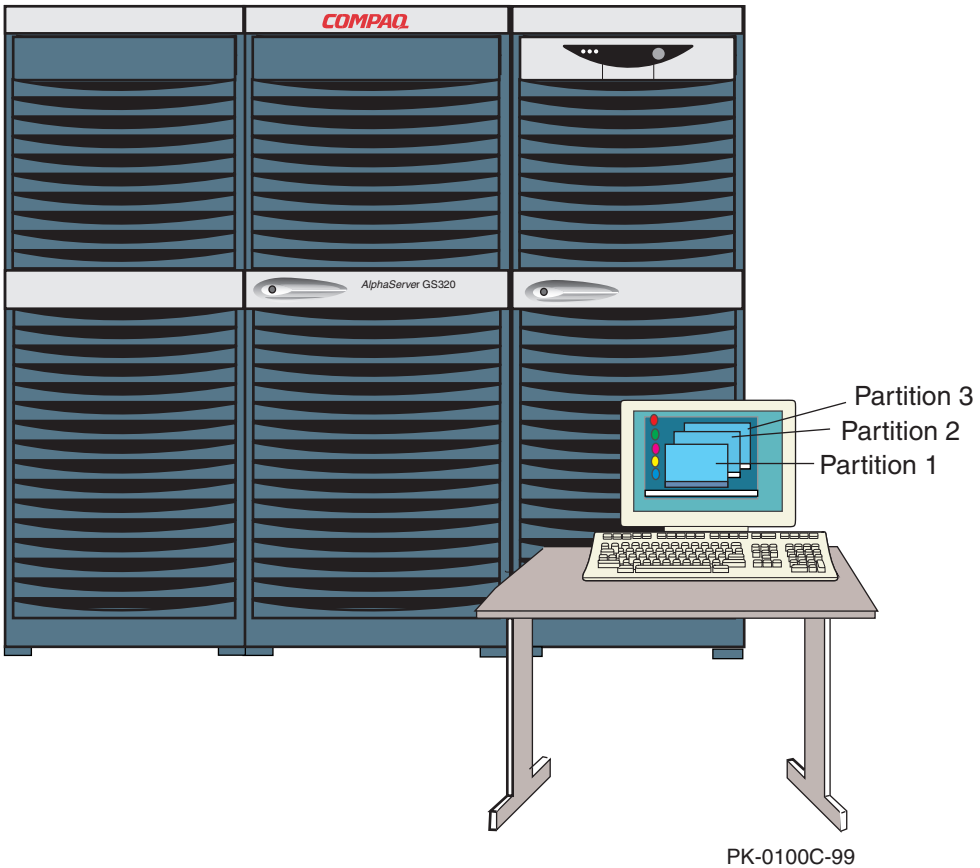
This chapter introduces the *Compaq AlphaServer* GS80, GS160, and GS320 firmware and the devices through which it is accessed. Sections include:

- System Console Device
- Remote Console Operations
- Booting an Operating System
- Setting Up and Initializing Partitions
- Firmware User Interface Overview

1.1 System Console Device

A local user or field service engineer interacts with the system firmware through the system management console. The system management console is connected directly to the system. A user may also communicate with the system remotely through the use of a modem.

Figure 1–1 System Management Console



During everyday operations, you will not interact with the firmware on an *AlphaServer* GS80, GS160, or GS320 system. Once fundamental parameters have been defined, powering on the system activates the firmware that controls power-up and boots the operating system, all automatically.

The firmware on an *AlphaServer* GS80/160/320 system supports six primary functions:

- Accessing the system from a local or remote console
- Booting an operating system
- Setting up and initializing partitions
- Setting up system parameters for booting
- Monitoring environmental conditions
- Testing the system

The other functions largely revolve around displaying information on the current state of the system.

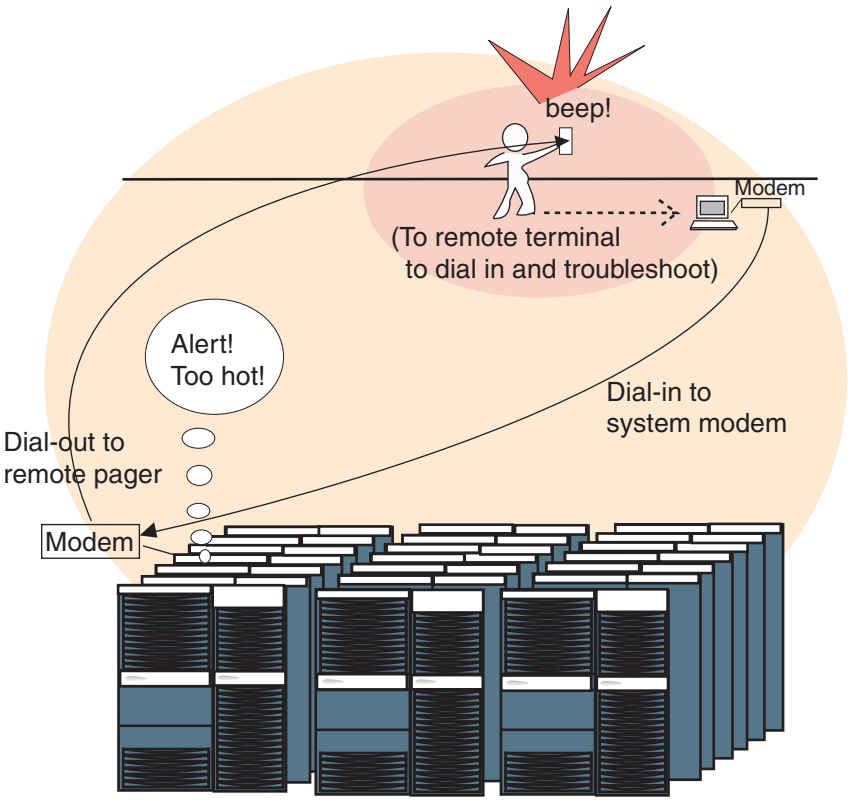
All these functions are accessed through either a local or remote *console device*. A console device may consist of a serial display monitor and a keyboard, or a terminal emulator on a PC, UNIX, or VMS workstation attached to the system.

Figure 1–1 shows the system management console monitor with windows for three partitions open on the monitor screen. In this case, a terminal emulator displays the screens; the commands typed at each screen direct the firmware to perform the actions requested.

1.2 Remote Console Operations

Some installations of *AlphaServer* GS80, GS160, or GS320 systems may require that the systems be managed by a person remote from the systems, in another room, building, or many miles away. The firmware supports such remote maintenance.

Figure 1–2 Managing a System from a Remote Console



PK-0125A-99

The *AlphaServer* GS80/160/320 systems are large, fast, modular systems. Some installations require operation 24 hours a day, seven days a week, and some installations may contain many of these systems. In these cases, it is reasonable to expect that these systems be monitored by a person remote from the systems, in another room, another building, or many miles away. The firmware supports such remote monitoring and management.

The key to remote management of these systems is a function called an *alert*. For example, the firmware monitors system conditions such as temperature and voltage. After setting up the capability for remote console operations, the firmware can recognize an alert condition, and dial out to an assigned telephone number, usually a pager, to notify the system manager that an alert condition exists (Figure 1–2).

The system manager can then use a serial terminal with a modem, or a PC or workstation with terminal emulation software, to access the remote system that sent the alert. System control manager (SCM) commands allow a user to troubleshoot a system from the remote terminal (or, for that matter, the local console terminal) without interrupting operating system execution. These commands include various **show** options that display status and the **examine** command to view error registers. If the system has crashed, you can reinitialize the system to reboot or to power off or reset, at which time you can run software to display error log information to isolate faulty hardware.

In addition to alerts, the systems, including the firmware and operating system software, have been carefully designed to keep the operating system running even though individual CPU, memory, and I/O modules, as well as AC/DC converters, fail. The remote user can routinely examine the status of the system and schedule service personnel for maintenance as appropriate.

1.3 Booting an Operating System

Most of the time, GS80/160/320 systems are running user programs under control of an operating system, either *Tru64 UNIX* or *OpenVMS*. When the operating system is shut down, control returns to the firmware. The firmware provides the means to boot the operating system, initially and after shutdown and maintenance.

Example 1–1 Booting an Operating System

```
P00>>> b dkb4 -fl A –file vmunix     ❶
(boot dkb400.4.0.1.2 -file vmunix -flags A)
block 0 of dkb400.4.0.1.2 is a valid boot block
reading 13 blocks from dkb400.4.0.1.2
bootstrap code read in
base = 344000, image_start = 0, image_bytes = 1a00
initializing HWRPB at 2000    ❷
initializing page table at 3fffb2000
initializing machine state
setting affinity to the primary CPU
jumping to bootstrap code    ❸
    .
    .
    .
Digital UNIX Version V4.0 (wfsys6.eng.pko.dec.com) console
login: root    ❹
Password:
Last login: Wed Sep  8 23:49:02 on console
   .
   .
   .
sys6>    ❺
```

Example 1–1 shows an operator booting the *Tru64 UNIX* operating system, using the SRM console command **boot** (abbreviated as **b** in the example). The SRM console firmware runs on the *primary processor,* selected during power-up, as described in Chapter 2.

❶ The operator issues the **boot** command. The boot disk is named directly as **dkb4**, the **–fl** option is used to request boot flag **A**, which boots the system disk to multiuser mode for *Tru64 UNIX*. The **–file** option specifies the name of the file to boot.

❷ The HWRPB (hardware restart parameter block) is an area in memory where parameters are passed between the operating system and the SRM console. (The HWRPB is discussed in detail in the *Alpha Architecture Manual*.)

❸ The SRM jumps to bootstrap code that has been loaded into memory from the boot device to boot the operating system from the disk named in the **boot** command.

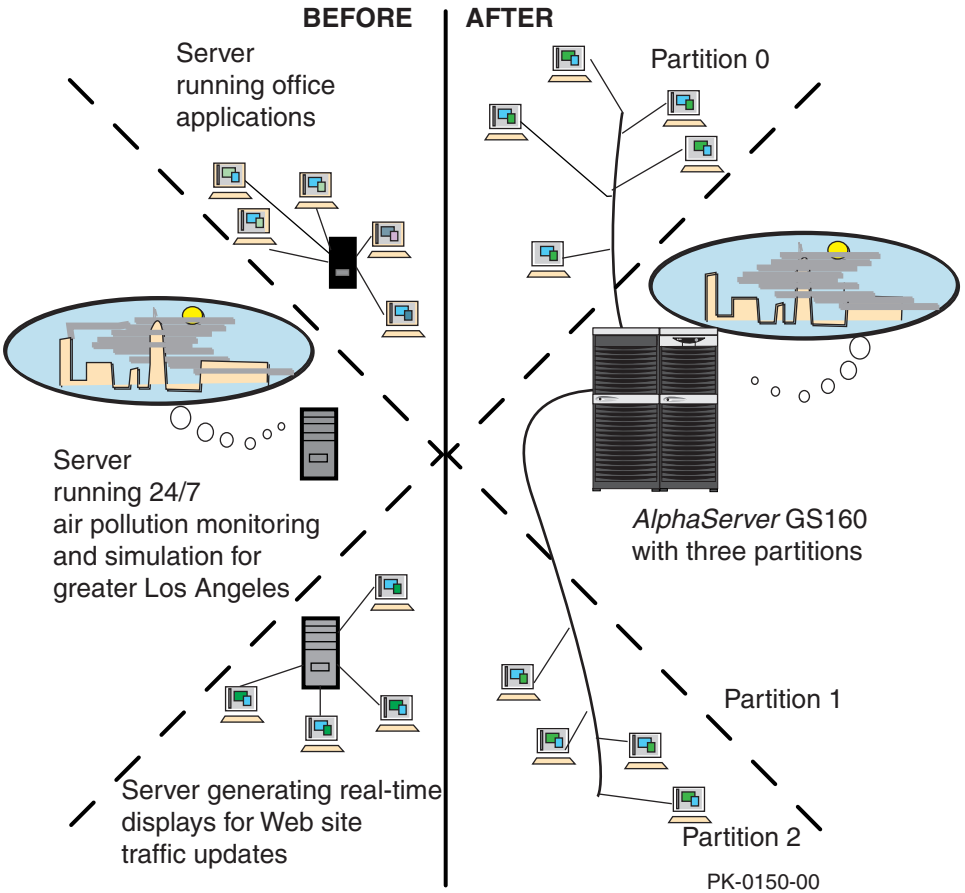
❹ The *Tru64 UNIX* operating system is given control and displays the prompt for login name and password.

❺ The operator is at the operating system prompt and can initiate operations to be performed by the operating system.

1.4 Setting Up and Initializing Partitions

The firmware allows you to set up and initialize partitions.

Figure 1–3 Using Partitions to Consolidate Resources



Multiple copies, or *instances*, of the *Tru64 UNIX* and/or *OpenVMS* operating systems can be run on *AlphaServer* GS80, GS160, and GS320 systems. This partitioning of the system is set up and initialized using the console firmware. You define the number of partitions and what resources are to be allocated to each partition.

One of the primary reasons for using partitions is shown in Figure 1–3. The left side of the figure shows three separate servers, each performing user tasks in a separate environment. The right side of the figure shows how these same functions can be performed on one larger server, using partitions.

1.5 Firmware User Interface Overview

A user at a console device has access to two interfaces: the SCM command language and the SRM command language. The Loadable Firmware Update (LFU) utility can be loaded from CD-ROM to update resident firmware.

Figure 1–4 Firmware User Interface Components

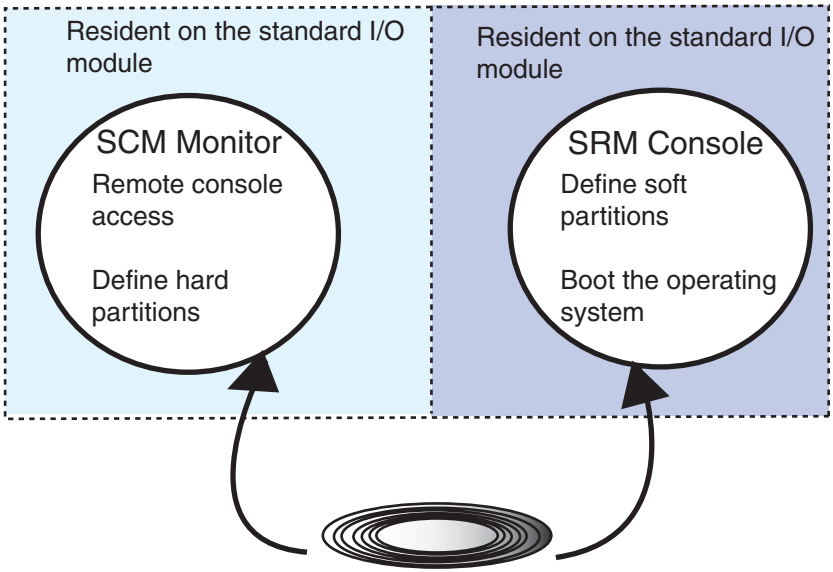


Figure 1–4 shows the two main user interfaces to the firmware and where the firmware resides. The figure also shows the Loadable Firmware Update (LFU) utility, which is loaded from DVD/CD-ROM when necessary to load or update the firmware.

SCM Monitor Command Language Interface

The system control manager (SCM) command language provides the interface to firmware that resides on nodes attached to the console serial bus, or CSB (see Section 2.8). The primary functions of this interface are to allow you to define and activate *remote console* services and to define *hard partitions*.

Remote console access allows a user to respond to an alert condition from a remote location without halting operating system execution.

Partitions allow the user to run multiple copies or *instances* of an operating system within one GS80/160/320 system. *Hard partitions* indicate that the boundaries of each partition remain fixed along boundaries defined by system building blocks called QBBs (see Section 2.4). See Chapter 4 for a discussion of partitions.

SRM Console Command Language Interface

The SRM console firmware resides in flash EEROM on the standard I/O module (or modules, if partitions are used). The command language lets you define whether the operating system is to be booted automatically at system power-up or by request. This feature applies to all operating systems offered with the *AlphaServer* GS80/160/320 systems.

In addition, the SRM console firmware provides the means to set up and initialize *soft partitions*, which can be run in conjunction with *OpenVMS Galaxy*. Soft partitions allow you to dedicate specific CPUs, memories, and I/O modules to run separate instances of an operating system. See Chapter 4 for a discussion of partitions.

A discussion of how the SRM command language affects booting is given in Chapter 5. The SRM command language interface is described in detail in Chapter 7.

Loadable Firmware Update (LFU) Utility

The LFU utility allows you to update the firmware components. LFU is on CD-ROM. The LFU interface is described in Appendix B.

Chapter 2
Hardware Overview

To understand firmware operations and displays for GS80/160/320 systems, it is helpful to understand the structure and architecture of the hardware.

If you are already familiar with the hardware comprising these systems, skip over this chapter to the one discussing the firmware you wish to use.

Sections in this chapter include:

- System Overview
  - Front and Rear View of a GS80 System
  - Front and Rear View of a GS160/320 System
- Control Panel
- Master PCI Ports and PCI Box ID
- Quad Building Block
  - QBB in GS80 Systems
  - QBB in GS160/GS320 Systems
- System Drawers and System Boxes
  - System Drawers in a GS80 System
  - System Box in a GS160 System
  - Two System Boxes in System Cabinet 1
  - System Boxes in System Cabinet 2
- Hard QBB Numbers
- QBB and PCI Box Connections
- Console Serial Bus
- System Power-Up

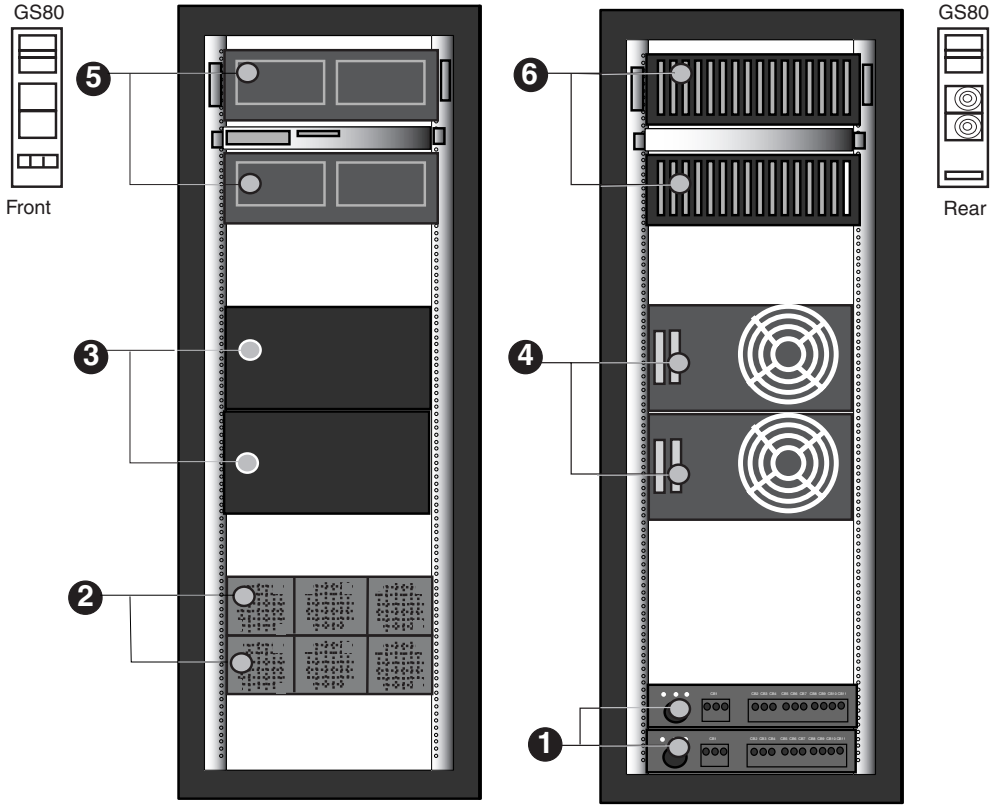
2.1 System Overview

In the *AlphaServer* GS80 system, all the components of the system are housed in one cabinet. For the GS160 system, the components are housed in two cabinets, and for the GS320, in three cabinets.

2.1.1 Front and Rear View of a GS80 System

The processors and other modules in a GS80 system are in either one or two system drawers.

Figure 2–1 Front and Rear View of a GS80 System



❶ One AC input box is required for each drawer in the GS80 system. An AC input box accepts single-phase 30-amp AC power and routes it to the AC-to-DC converters.

❷ Power supplies; six are shown. Two are required for each system drawer; a third provides n+1 redundancy. That is, if one power supply in a power subrack fails, the system will still function at full power and the failing power supply can be "hot swapped" for a replacement without powering down the whole system.

❸ Two system drawers are shown. Each drawer can be pulled out for maintenance.

❹ The rear view of two system drawers is shown. The cooling fan of a system drawer can be accessed from the rear of the cabinet. Two I/O risers protrude from the system drawers (shown as the light rectangles in the figure). Each I/O riser provides two I/O cables. Two cables connect to one PCI box.

❺ The front of the cabinet shows two PCI boxes. The cooling fans for the PCI boxes can be accessed from the front of the cabinet.

❻ The rear view of two PCI boxes is shown. The PCI boxes can be pulled forward to remove and replace modules.

2.1.2 Front and Rear View of a GS160/320 System

System components are housed in a power cabinet and either one (GS160) or two (GS320) system cabinets, depending on the size of the system.

Figure 2–2 System Front and Rear Views (GS160/320)

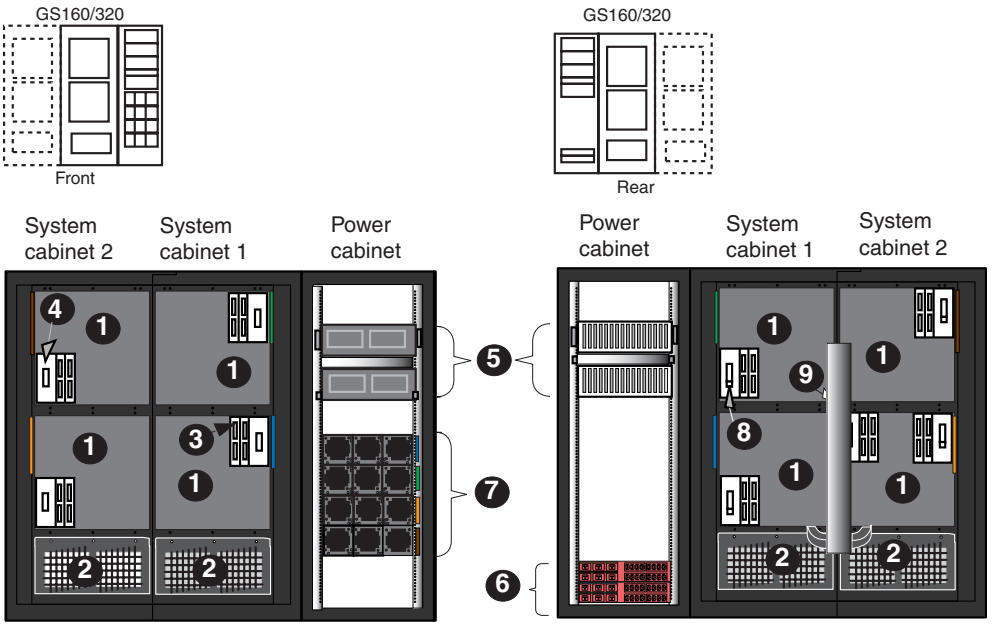


Figure 2–2 shows front and rear views of a GS320 system with the doors open. (At the front of the machine, the *power cabinet* is on the right. *System cabinet 1* is in the middle, and *system cabinet 2* is on the left. A GS160 system consists of the power cabinet and system cabinet 1; there is no system cabinet 2 on a GS160 system.

❶ Four system boxes are shown. The modules in each system box are covered by EMI protective shields, exposing only the I/O risers and the signal input connectors at the front, and I/O risers and power input at the rear. Colors strips at the side of each system box show the sequence for adding system boxes: blue, green, orange, brown.

❷ A blower in each system cabinet draws air down through the system boxes and out the front and back of the cabinet. The speed varies as the ambient temperature changes, keeping the modules at the proper temperature.

❸ I/O connectors. Four connectors are shown for each system box. Each connector handles one cable and two cables connect to one PCI box.

❹ Signal input connector. Signals conveying data between modules, components of the console serial bus, and the operator control panel (OCP) are transmitted over the signal lines. These lines go to the system boxes at the front of the machine.

❺ The front of the power cabinet reveals the back side of a master PCI box and one additional PCI box. The rear of the power cabinet shows the front side of a master PCI box and one additional PCI box.

❻ Two AC input boxes are required for GS160 and GS320 systems. Each box accepts three-phase 30-amp AC power and routes it to the power supplies.

❼ Power supplies. Each subrack (row of individual power supplies) provides power to one system box. The rows are color-coded to match the cover of the system box to which they provide power. Each power subrack accepts AC input and converts it to 48-volt DC output.

One power supply is routed to one QBB backplane, and there are two per system box. The extra power supply provides n+1 redundancy.

❽ Power input connectors. Cables providing 48V to the QBB backplanes enter each system box here.

❾ The hierarchical switch provides logic that controls the transfer of instructions and data between memory, CPU, and I/O modules in the QBBs, and between nodes on the console serial bus (see Section 2.8). A hierarchical switch is needed for 4-QBB GS160 systems and for all GS320 systems. A component called a distribution board provides a similar purpose in 2-QBB GS160 systems.

2.2 Control Panel

The operator control panel (OCP) is located on the front door of a GS80 system or the front door of the power cabinet in a GS160/320 system. The OCP contains a keyswitch that allows a user to fully power up or power down the system, or secure the system from remote access. Three buttons allow the user to halt or reset the system, or halt the system with error registers latched (fault).

Figure 2–3 Control Panel

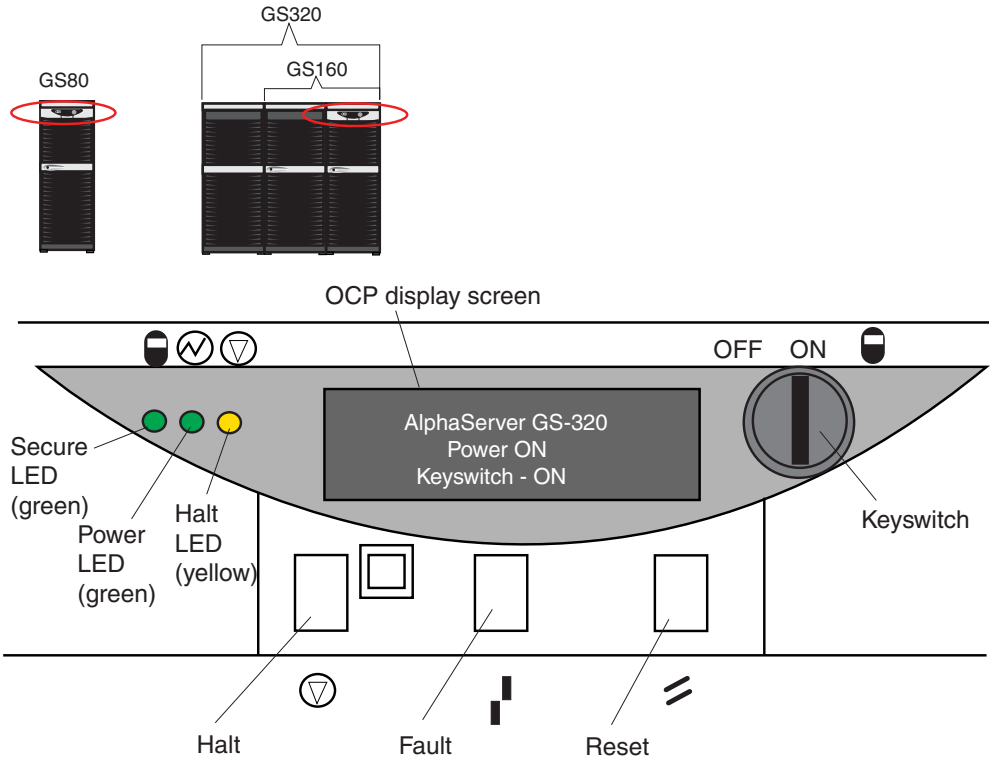


Figure 2–3 shows the operator control panel (OCP) that is located on the front door of the GS80 cabinet and the front door of the power cabinet in GS160 and GS320 systems.

Keyswitch

The three-position keyswitch allows the local user to control power and remote management features. When in the 1 or On position, the system is enabled to be powered up. A remote system user can power on or power off the system.

When in the 0 or Off position, the system is powered off and cannot be remotely powered on by a remote system user.

When in the Secure or lock position, the system is powered on but the Halt, Fault, and Reset buttons and the local and remote management ports are disabled.

Pushbuttons

Three pushbutton switches are below the display.

The Halt button causes the operating system to perform a halt, and prevents the console from booting the operating system. This halt stays in effect until the SRM **continue** command is executed.

The Fault button is a momentary switch that causes a system reset without clearing captured error information in the control and status registers (CSRs).

The Reset button is a momentary switch that causes a system reset that clears error information captured in the CSRs.

LEDs

The Secure LED indicates that the keyswitch has been turned to the Secure (lock) position. The Halt, Fault, and Reset buttons, as well as the local and remote management ports, are disabled.

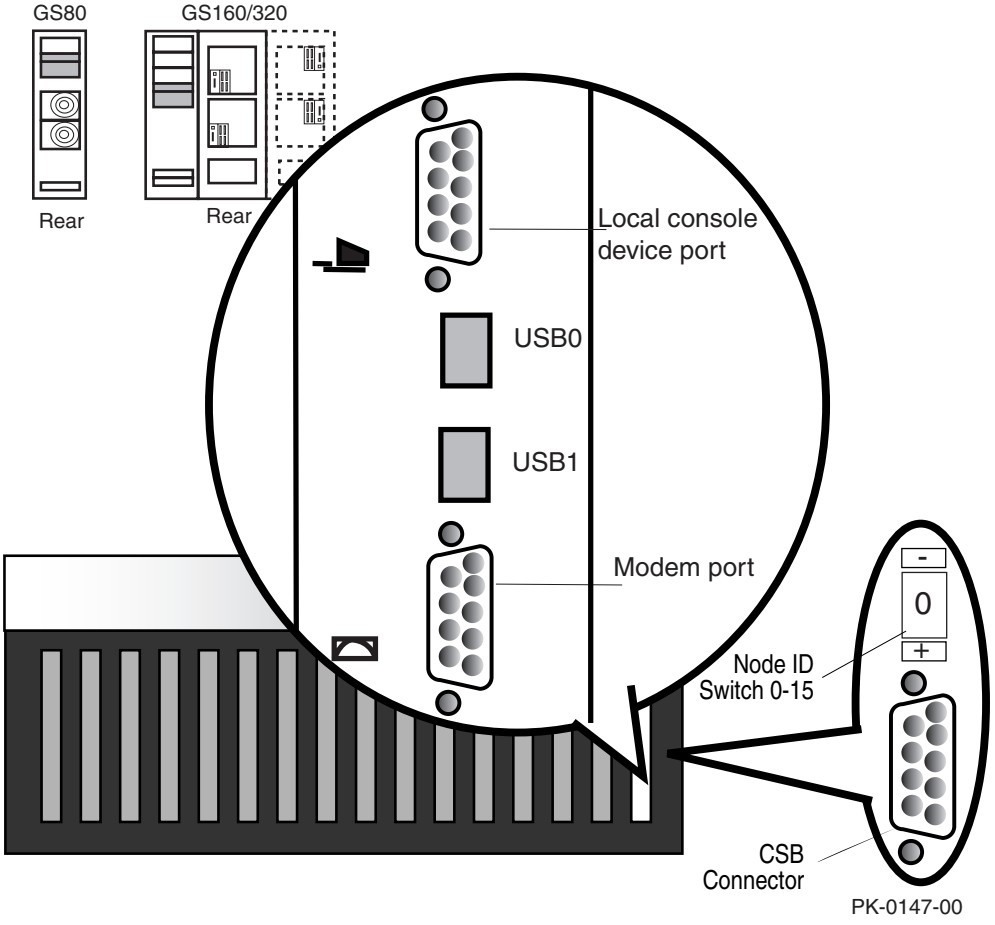
The Power LED indicates that the system is powered on.

The Halt LED will never be the only LED lit. It may be lit in conjunction with Power, indicating that the system is powered on, remote console operations are enabled, and the Halt button is pressed or a **halt in** command has been issued at the local or remote console. It may be lit along with both other LEDs, indicating a powered on system, the Halt button has been pressed or a **halt in** command has been issued at the local or remote console, followed by turning the keyswitch to the Secure (lock) position.

2.3 Master PCI Ports and PCI Box ID

A master PCI box provides ports for a local system console device as well as a modem port for remote connections from the standard I/O module. The reverse side of a master PCI box allows access to the DVD/CD-ROM device as well as various ports, including the connection from the SCM to the operator control panel.

Figure 2–4 Console and Modem Ports on Standard I/O Module

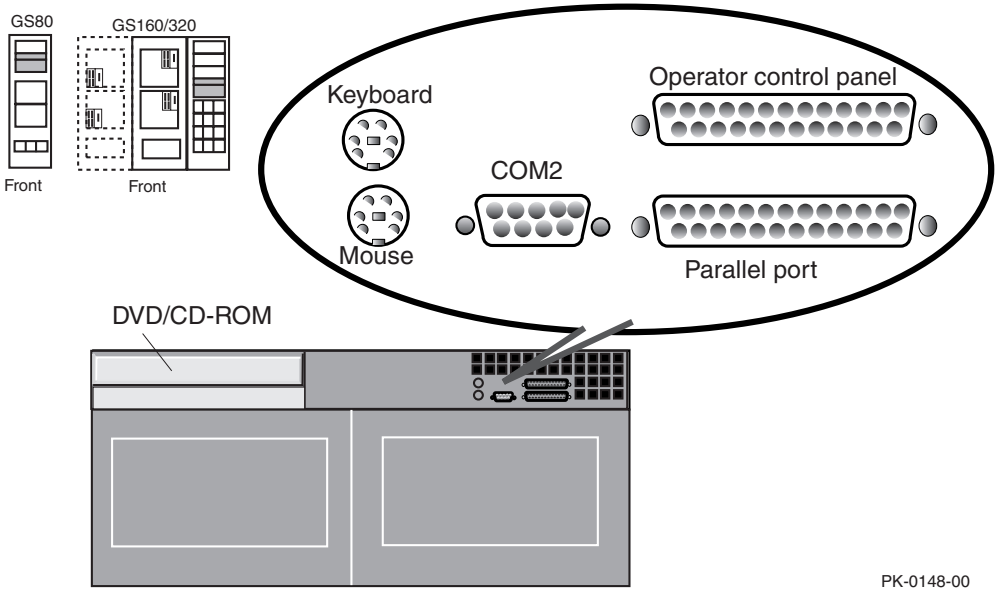


Console and modem ports are on each standard I/O module in a master PCI box for either the system or an individual partition. The console port is at the top of the module, and the modem port is at the bottom of the module, as shown in Figure 2–4.

Figure 2–4 also shows the CSB connector and the node ID switch that is on the right side of each PCI box. The node ID switch is set when a PCI box is installed to form the PCI box ID; each setting is unique to the system and can range from 0 –15. This ID appears in **show** displays that provide information on the I/O subsystem. For example, the ID appears as the number in the **show system** display for PCI nodes. The PCI ID also appears in the SRM **show config** display.

Figure 2–5 shows the DVD/CD-ROM device and the ports provided by a master PCI.

Figure 2–5 Master PCI Ports



2.4 Quad Building Block

The basic processing component of *AlphaServer* GS80/160/320 systems is the quad building block (QBB). The backplane for a QBB differs slightly in GS80 and GS160/320 systems.

2.4.1 QBB in GS80 Systems

A GS80 system can have one or two QBBs. Each QBB resides in a drawer.

Figure 2–6 GS80 QBB Backplane

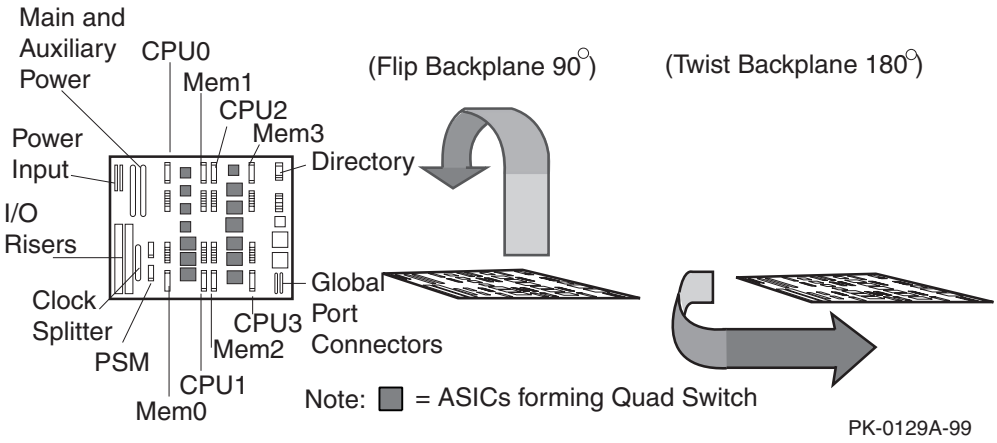
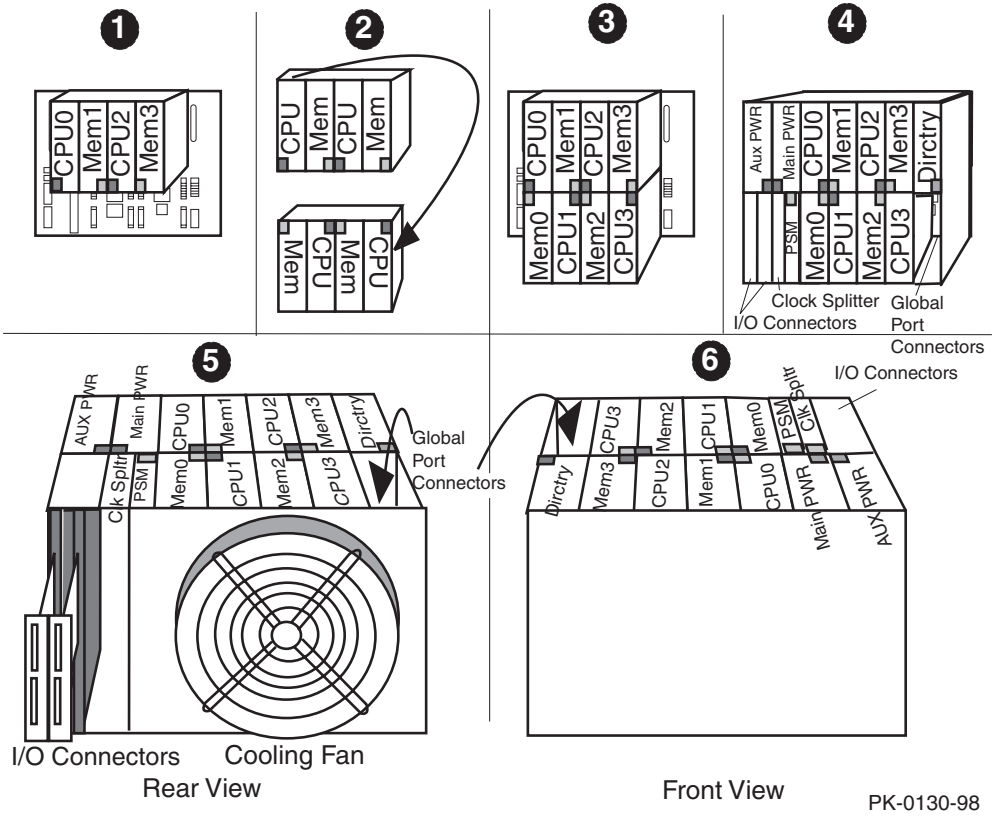


Figure 2–6 shows the backplane of a GS80 QBB; the arrows show how the backplane is oriented to form the bottom of a *system drawer* within which the QBB is placed. The backplane has connectors for four processors and four memories, from which the name quad building block derives. The backplane also has connectors for power input, a main and an auxiliary power module, two I/O risers, a clock splitter module, and logic for a global port, used to connect two backplanes.

The gray shaded blocks are ASICs (small processors). The logic in these components form what is called the *quad switch*, which regulates data traffic between modules in a QBB.

Figure 2–7 illustrates how modules are mounted in a system drawer. The first section (❶) shows two CPUs and two memories. In section ❷, these modules are replicated and rotated 180°. Section ❸ shows the replicated modules in place. The modules are color-coded (small shaded blocks), but for GS80 systems, you identify the placement of modules by a label on the inside of the cover of each system drawer. Section ❹ shows the addition of the remaining modules. Section ❺ shows the QBB rotated 90º, as it would be in a system drawer in a GS80 system. Note the cooling fan and I/O extenders as seen from the rear of the system. Section ❻ shows the QBB rotated 180º, as it would be seen from the front of the system.

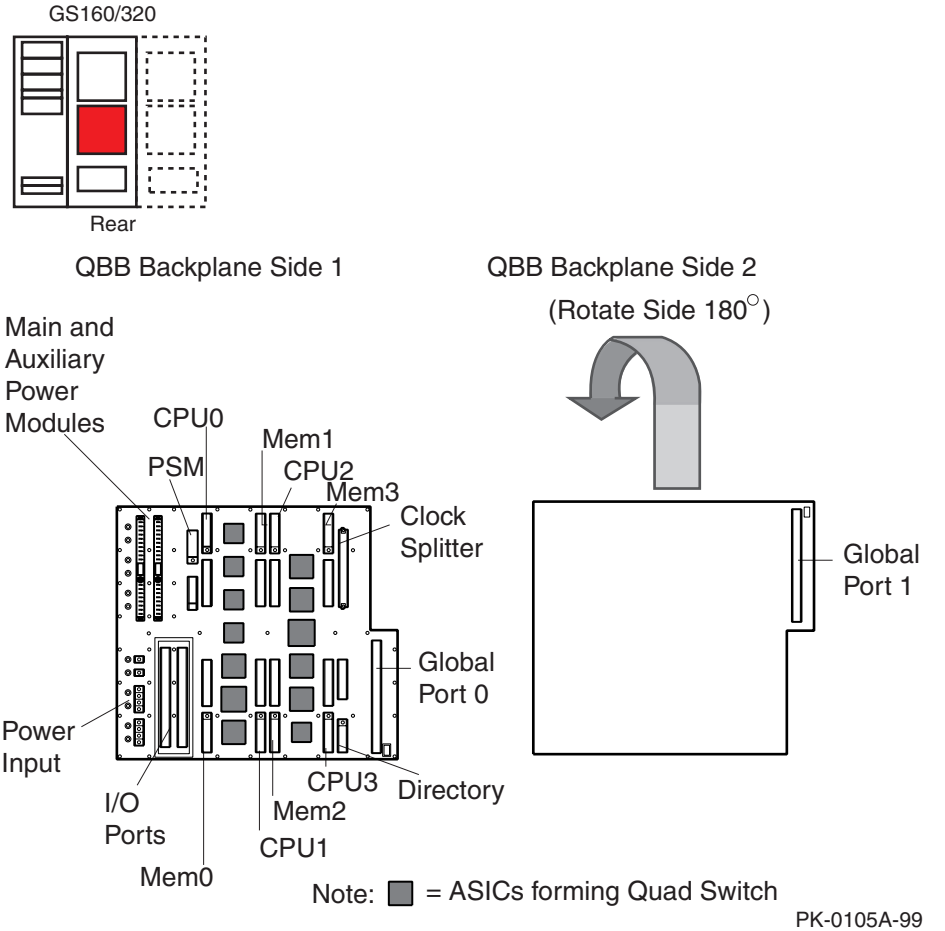
Figure 2–7 Elements of a GS80 QBB (System Drawer)



2.4.2 QBB in GS160/320 Systems

GS160/320 system cabinets contain *system boxes*, each containing two QBBs. This section shows the backplane as seen from the rear of the machine in the lowest system box in system cabinet 1.

Figure 2–8 QBB Backplane for GS160/320 Systems

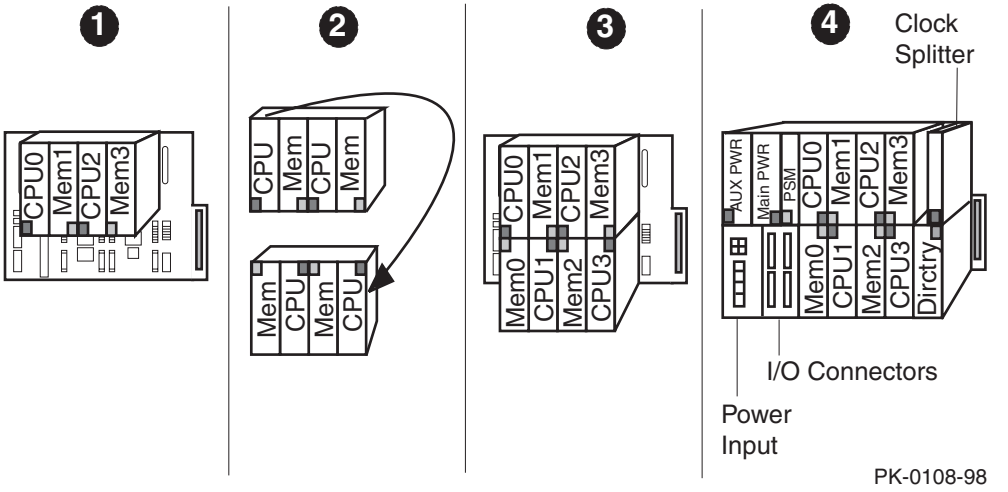


A QBB backplane in a GS160/320 system box is shown in Figure 2–8 as seen in system cabinet 1 at the rear of the system. The backplane has connectors for four processors and four memories, from which the name quad building block derives. The backplane also contains connectors for power input, main and auxiliary power modules, two I/O risers, and a clock splitter module. Note global port 0 on side 1 and global port 1 on side 2 of the backplane.

The gray shaded blocks are ASICs (small processors). The logic in these components form what is called the *quad switch*, which regulates data traffic between modules in a QBB.

Figure 2–9 shows stages of adding modules to construct a QBB. At ❶, we begin with two CPUs and two memories. At ❷, these modules are replicated and rotated 180° to produce the mountings shown in ❸. Modules are color-coded (small shaded blocks). Each type of module has a different color, such as blue for CPUs. A metal bar bisecting the front of the QBB's support structure is coded so that module colors match colors on the bar (see Figure 2–12). Section ❹ shows the addition of the remaining modules. Numbers on the bar and guides on the backplane ensure that modules are inserted in the correct position and with the right orientation.

Figure 2–9 Elements of a QBB for GS160/320 Systems



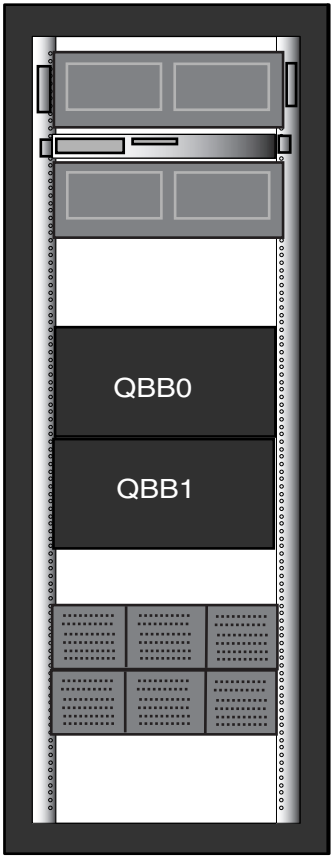
2.5 System Drawers and System Boxes

GS80 systems contain one or two system drawers. GS160 and GS320 systems contain one to four system boxes.

2.5.1 System Drawers in a GS80 System

A GS80 system may contain one or two system drawers. Each system drawer holds a QBB.

Figure 2–10 System Drawers in a GS80 System



PK-0135-00

QBBs are numbered according to the location they occupy in the system cabinet. As shown in Figure 2–10, QBB0 is the upper and QBB1 is the lower (in a one-drawer GS80 system, there is only QBB0). The SCM and SRM console firmware **show** commands display self-test results, system configuration, and other information that includes these *hard QBB numbers*, along with *soft QBB numbers* used by the firmware and software.

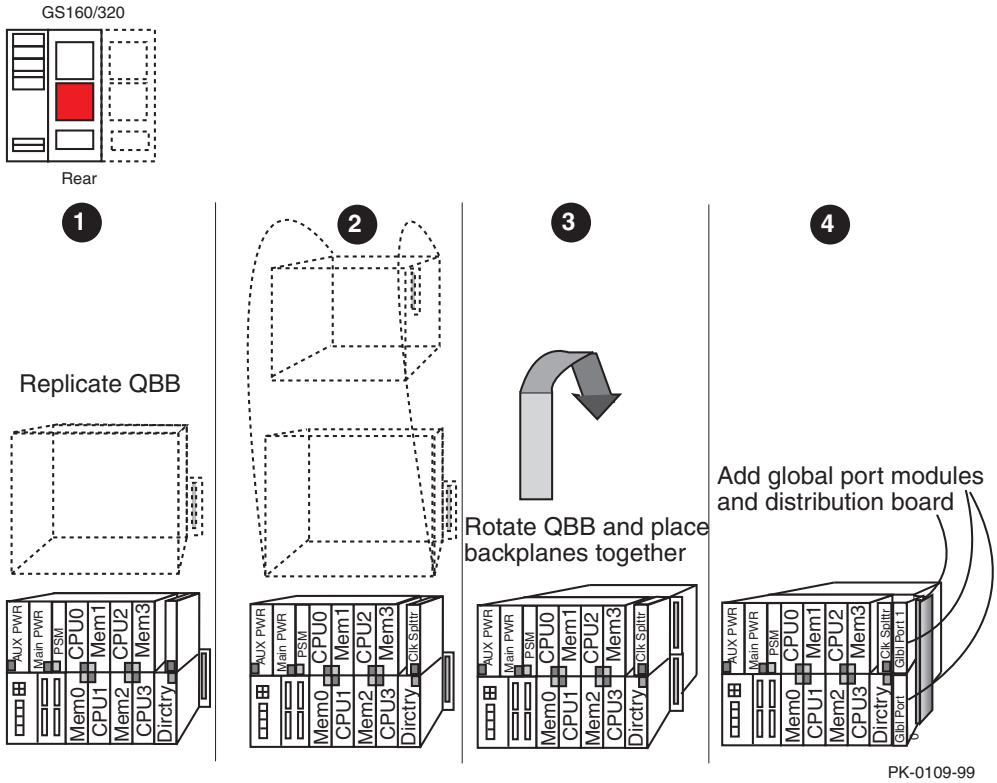
Both numbers are displayed because, for reasons discussed in Section 2.9, the firmware and software use soft QBB numbers for addressing. While you might need only the hard QBB number to find a faulty module revealed by a self-test display or a **show** command, you may need the soft QBB number when doing lower-level debugging.

For example, you will need the soft QBB number to interpret any error register contents displayed with an **examine** command. And, you need to be able to relate the soft QBB number to a particular physical location using the hard QBB number. Thus, both numbers are displayed in power-up self-test displays and **show** command displays.

2.5.2 System Box in a GS160 System

A system box consists of two quad building blocks, connected by global port modules and a distribution board. This section shows how the system box for a 2-QBB GS160 system is constructed.

Figure 2–11 Elements of a System Box (GS160 Rear Orientation)

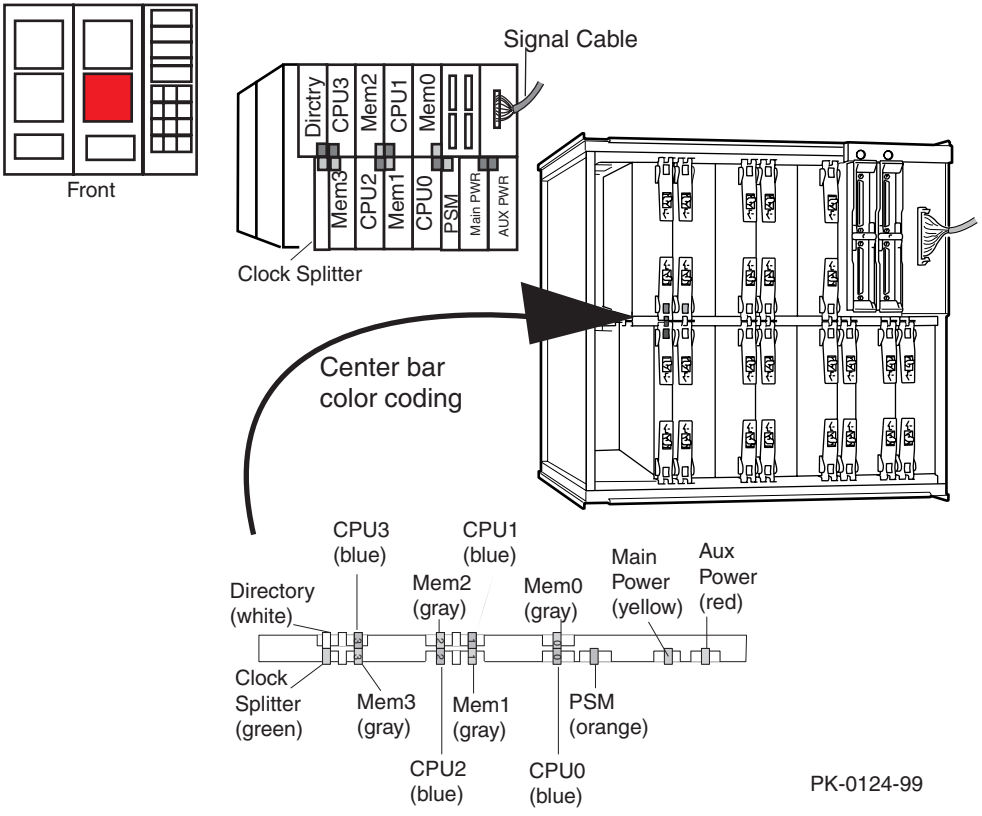


The smallest GS160 system contains one *system box*. A system box consists of two QBBs. Figure 2–11 illustrates the construction of a system box. A QBB is replicated (❶), and rotated 180º (❷). The backplanes are placed together, so that both global port connections are on the same side (❸).

The global port modules (❹) connect the QBBs to a distribution board, allowing data to flow between modules from both QBBs of the system box. The reverse side of a system box (see Figure 2–12) shows signal cabling rather than power cabling.

Figure 2–12 also shows the system box in detail. The bottom shows the center bar color coding for the reverse side of a system box. Note that there are digits as well as colors to identify specific CPUs and memories.

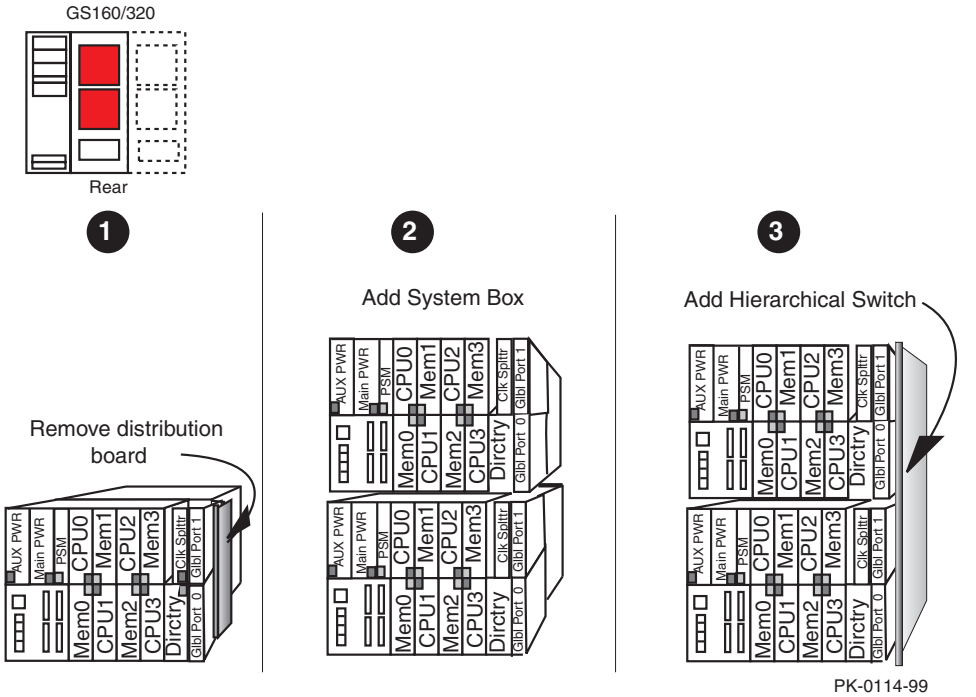
Figure 2–12 Reverse Side of System Box (Front of System Cabinet 1)



2.5.3 Two System Boxes in System Cabinet 1

System cabinet 1 of a GS160/320 system can contain two system boxes, the second stacked on top of the first. The system boxes are connected through the four global port modules by a hierarchical switch.

Figure 2–13 System Boxes in System Cabinet 1



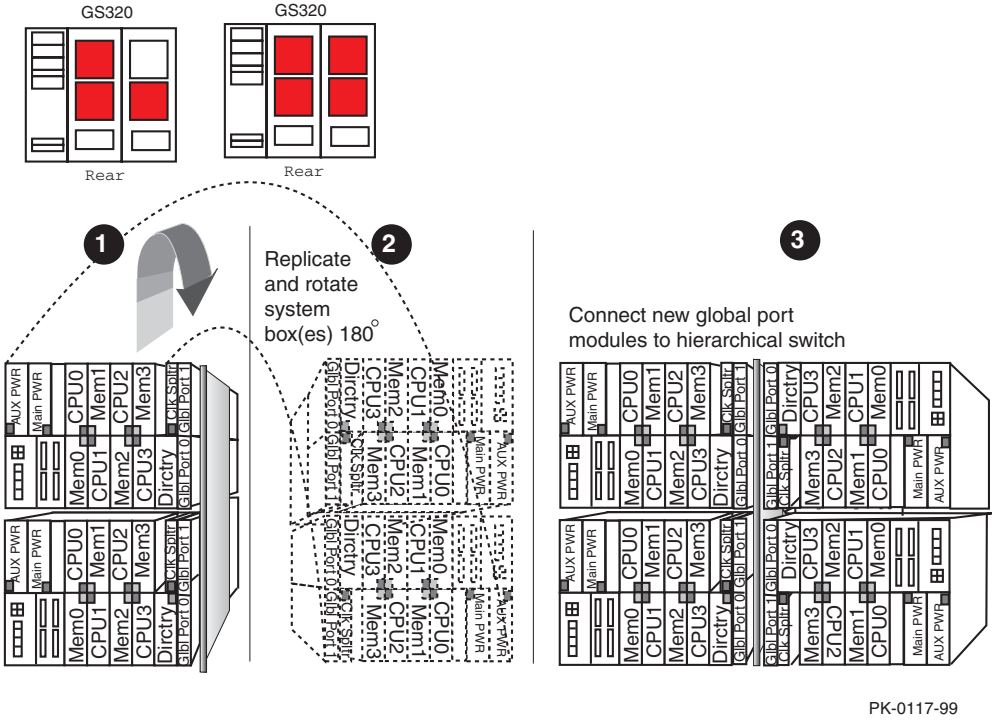
System cabinet 1 of a GS160/320 system can contain two system boxes. Figure 2–13 shows how two system boxes are placed and connected by the *hierarchical switch*. The distribution board is removed from the system box (❶), and the second system box is replicated and placed over the first system box (❷).

All four of the global port modules on the two system boxes are connected by the hierarchical switch (❸). The positioning of the modules on the reverse (front) side of system cabinet 1 are as shown in Figure 2–12 on page 2-17.

2.5.4 System Boxes in System Cabinet 2

A GS320 system contains three or four system boxes. The system boxes reside in system cabinets 1 and 2 and are constructed as shown in Figure 2–14. The system boxes are connected through the global port modules by the hierarchical switch.

Figure 2–14 Elements of System Cabinets 1 and 2

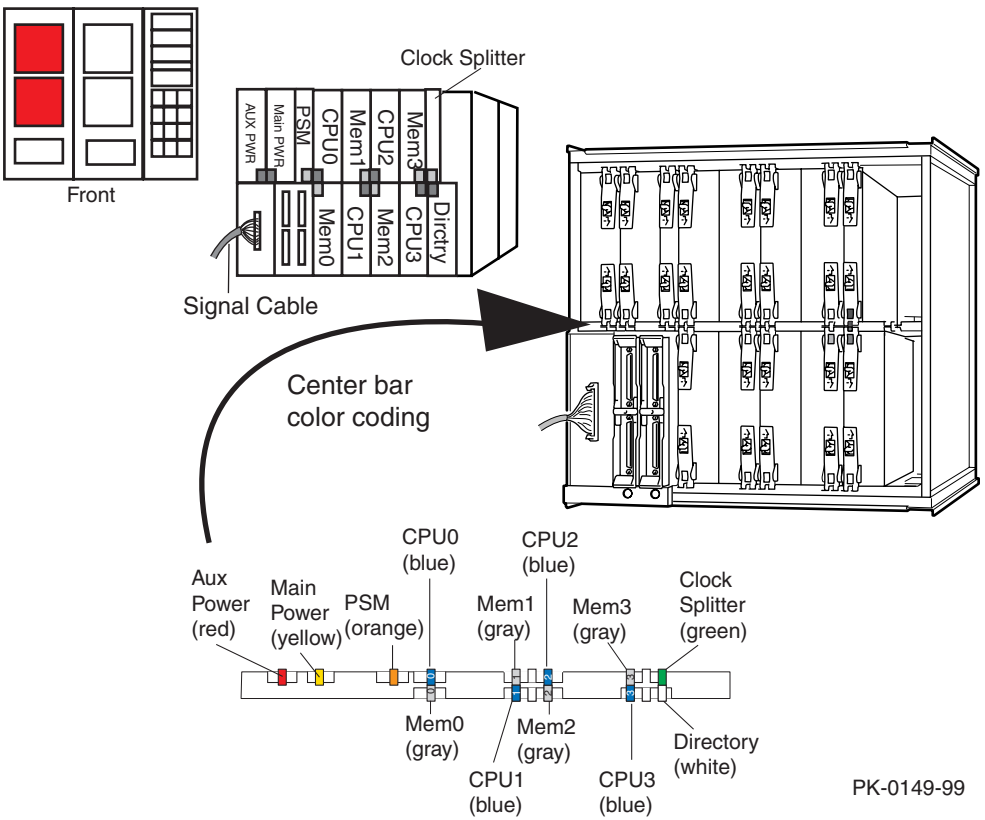


PK-0117-99

Figure 2–14 shows how three or four system boxes are connected in a GS320 system. The system boxes are connected by the *hierarchical switch*. In an 8-QBB system, the two system boxes in system cabinet 1 (❶) are replicated and rotated 180° sideways (❷). In a 6-QBB system, one system box is replicated, rotated 180° sideways, and occupies the lower position, as shown in the first icon at the top of the figure.

The four global ports on the replicated unit are connected to the hierarchical switch to form an 8-QBB system (❸). Note how the module placement appears in the replicated unit. The positioning of the modules on the reverse (front) side of the system are as shown in Figure 2–15.

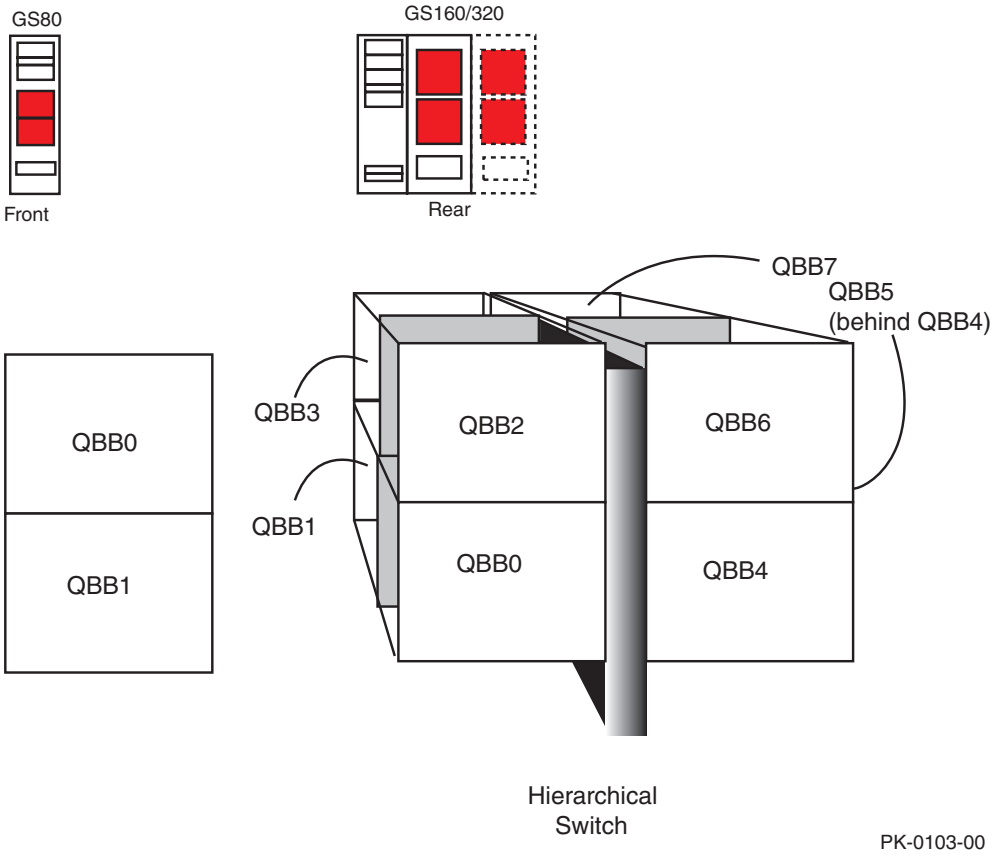
Figure 2–15 Front of System Box in System Cabinet 2



2.6 Hard QBB Numbers

Hard QBB numbers define the location of a particular QBB in the system.

Figure 2–16 Hard QBB Numbers



QBBs are numbered according to the location they occupy in the system, as shown in Figure 2–16. These are "hard" QBB numbers. The SCM and SRM console firmware **show** commands display self-test results, system configuration, and other information that includes these *hard QBB numbers*, which never change, along *with soft QBB numbers,* which can change with configuration changes.

Both numbers are displayed because, for reasons discussed in Section 2.9, the firmware and software use soft QBB numbers for addressing. While you might need only the hard QBB number to find a faulty module revealed by a self-test display or a **show** command, you may need the soft QBB number when doing lower-level debugging.

For example, you will need the soft QBB number to interpret any error register contents displayed with an **examine** command. And, you need to be able to relate the soft QBB number to a particular physical location using the hard QBB number. Thus, both numbers are displayed by the **show config** command.

2.7 QBB and PCI Box Connections

This section discusses I/O paths between QBBs and PCI boxes.

Figure 2–17 QBB Local I/O Port Numbers

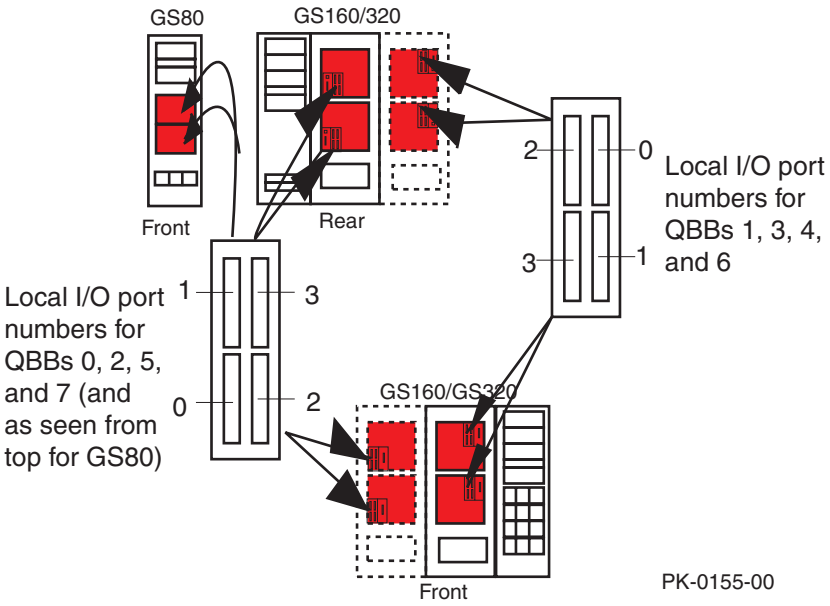


Figure 2–17 shows the port numbers for the four connectors on the two I/O riser modules in a QBB. Labels on the cabinets identify *local I/O ports*.

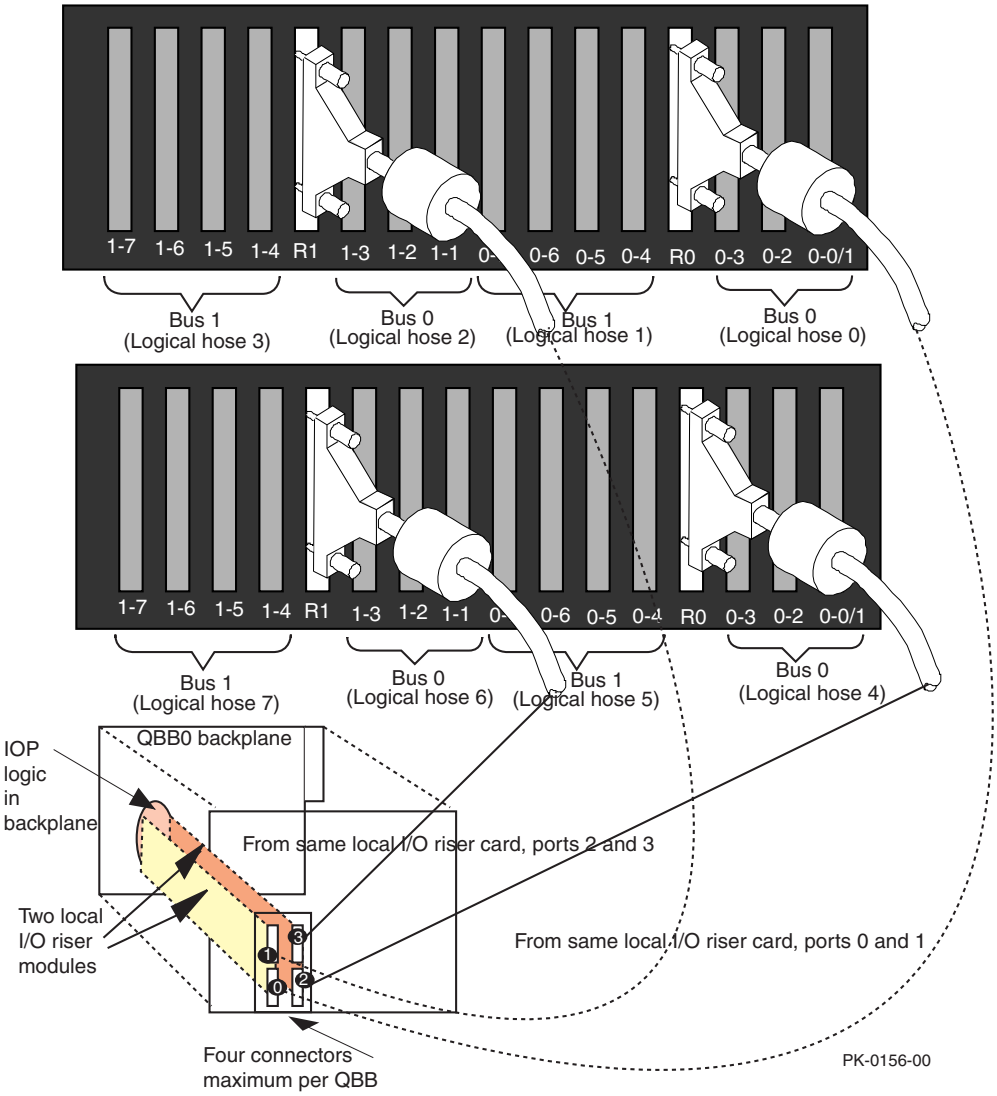
Figure 2–18 shows how the four local I/O port connectors in QBB0 could be connected to remote I/O riser ports in two PCI boxes. Although the figure shows even local ports being connected to even remote risers, and odd to odd, there are no electrical or physical restrictions on local port/remote riser cabling. Guidelines are given in the *AlphaServer* GS80/160/320 *Installation Guides*.

Equation 2-1 gives the general algorithm for calculating the logical hose numbers shown in the SRM **show config** command.

Equation 2-1 Calculating Logical Hose Numbers

(hard QBB number × 8) + (local port number × 2) + remote riser number

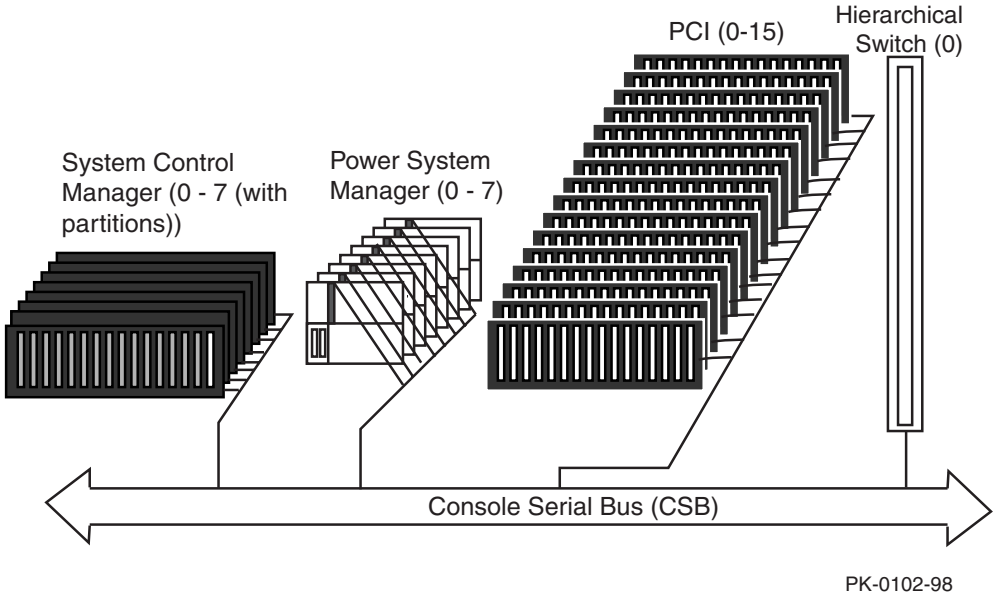
Figure 2–18 Sample Cabling to PCI Boxes



2.8 Console Serial Bus

System firmware resides on nodes connected to the console serial bus (CSB). This firmware controls power-up, monitors conditions within the system, alerts users of potential problems, and provides the capability for partitions and for remote access.

Figure 2–19 Logic Diagram of the Console Serial Bus



When a GS80/160/320 system is plugged in and the main breaker switch is on, the console serial bus (CSB) and its nodes receive power and are functional. The system as a whole is not powered up, however. Figure 2–19 shows the total possible nodes on the CSB for an eight-QBB GS320 system with 8 partitions.

System Control Manager (SCM)

In a non-partitioned system, the central point of control on the CSB is the master system control manager (SCM). (There may also be a slave SCM that serves as a standby, defined by the SCM **scm_csb_master_eligible** environment variable.)

In a partitioned system, partitions must be maintained independently. Each partition requires an SCM. Based on information provided when partitions are set up, one SCM is selected as the master and the rest are slaves. Each SCM, master or slave, controls parameters relating to the partition where they reside.

SCM hardware resides on the standard I/O module (SIO), and includes an AM186ES microprocessor, 256 KB of flash ROM, 8K EEPROM, 128 KB of RAM, and two dual universal asynchronous receiver/transmitters (UARTs). The master SCM's microprocessor handles all communication to and from other nodes on the console serial bus and the control panel. Each SCM, master or slave, processes I/O to or from local or remote terminals.

Power System Manager(s) (PSMs)

A power system manager (PSM) is a module in a QBB. PSM hardware includes a microprocessor, flash EEPROM, RAM, three internal I^2C buses, and three analog/digital hardware monitoring devices, which monitor voltages, temperatures, and fan speeds. A PSM also supports a serial interface to each CPU in a QBB; it uses this interface to direct the testing of, and collect information from, the CPUs during power-up.

PCI Backplane Manager(s) (PBMs)

A PCI backplane manager (PBM) resides on the backplane of each PCI subsystem. PBM hardware consists of a microprocessor and associated peripherals. The PBM monitors environmental sensors (voltage, temperature, fan) and sends *alerts* when appropriate. The PBM also maintains an inventory, including the backplane, standard I/O module (if any), remote I/O riser modules, PCI module presence (not specific PCI options), and power supplies.

Hierarchical Switch Power Manager (HPM)

AlphaServer GS160 and GS320 systems contain a hierarchical switch whose logic regulates data and instruction flow between QBBs. The HPM monitors environmental sensors (voltage, temperature) for the hierarchical switch.

2.9 System Power-Up

At power-up, component self-tests complete, a primary processor is selected, and control passes to the SRM console.

Figure 2–20 Location of Nodes on the Console Serial Bus

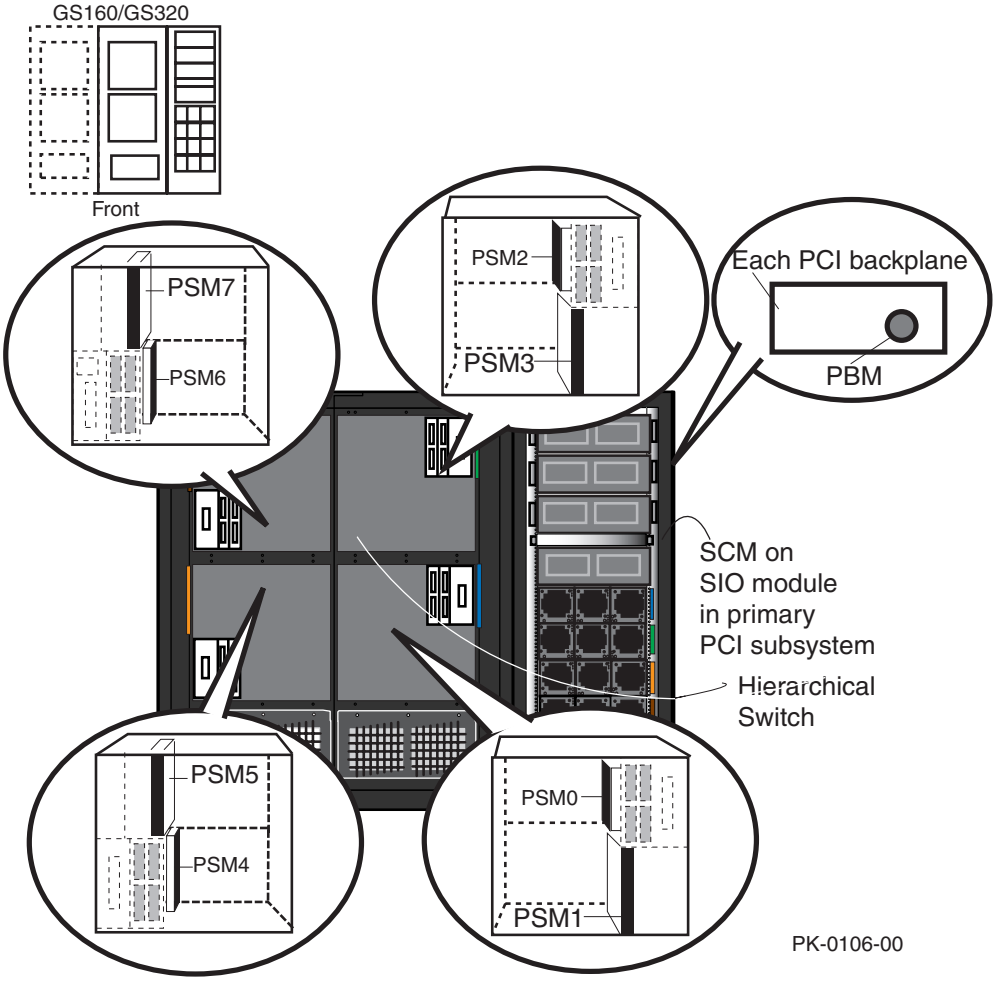


Figure 2–20 shows the physical location of nodes on the console serial bus. Each of these nodes plays a part in system power-up; individual CPU modules in QBBs called the *local CPU*s also play a part in power-up and are eventually replaced by one *primary CPU*.

The SCM assumes control when someone turns the keyswitch to the Secure position or when the keyswitch is in the On or 1 position or a console user issues a **power on** command. The SCM checks each PSM.

Each PSM powers up its backplane and initiates module self-tests. Each PSM then powers up each I/O riser, selects a *local CPU* (for that QBB) and tells that CPU to probe for standard I/O modules through the cables for that QBB's I/O risers. The local CPU performs initialization of various registers, and if a standard I/O module is found, returns that information to the PSM.

The PSM waits for all self-tests to complete and reports the good CPUs, memory, and all standard I/O modules found through that QBB's I/O risers.

The SCM updates the system map to include these results and selects the *primary CPU* and "soft" QBB0. The firmware and operating system software use physical memory addresses whose high-order three bits include the soft QBB number. The *Tru64 UNIX* operating system currently requires memory to start at physical address 0. Since it is possible that hard QBB0 might not contain a valid memory module, *soft QBB0* is assigned to ensure that what the firmware and software regards as QBB0 contains at least one valid memory module. Other soft QBB numbers are selected and this "soft system map" is passed to the PSM.

At this point, the *primary processor* assigns soft QBB IDs to all remote QBBs.

The primary processor then configures memory, loads the console firmware from flash EEPROM on a standard I/O module, and transfers control to the SRM console.

Chapter 3
Remote Console Operations

The system control manager (SCM) firmware provides a command-line interface that can be used at the local console terminal or at a remote terminal. Before a remote console can be used, however, you must initialize the modem, define the dial-out number for a remote pager, and perform other tasks.

This chapter describes tasks performed at the local console terminal to set up remote terminal operations, as well as operations that can be performed at either the local or remote terminal to display status or to supply or remove power from the system or a part of the system.

Sections in this chapter include:

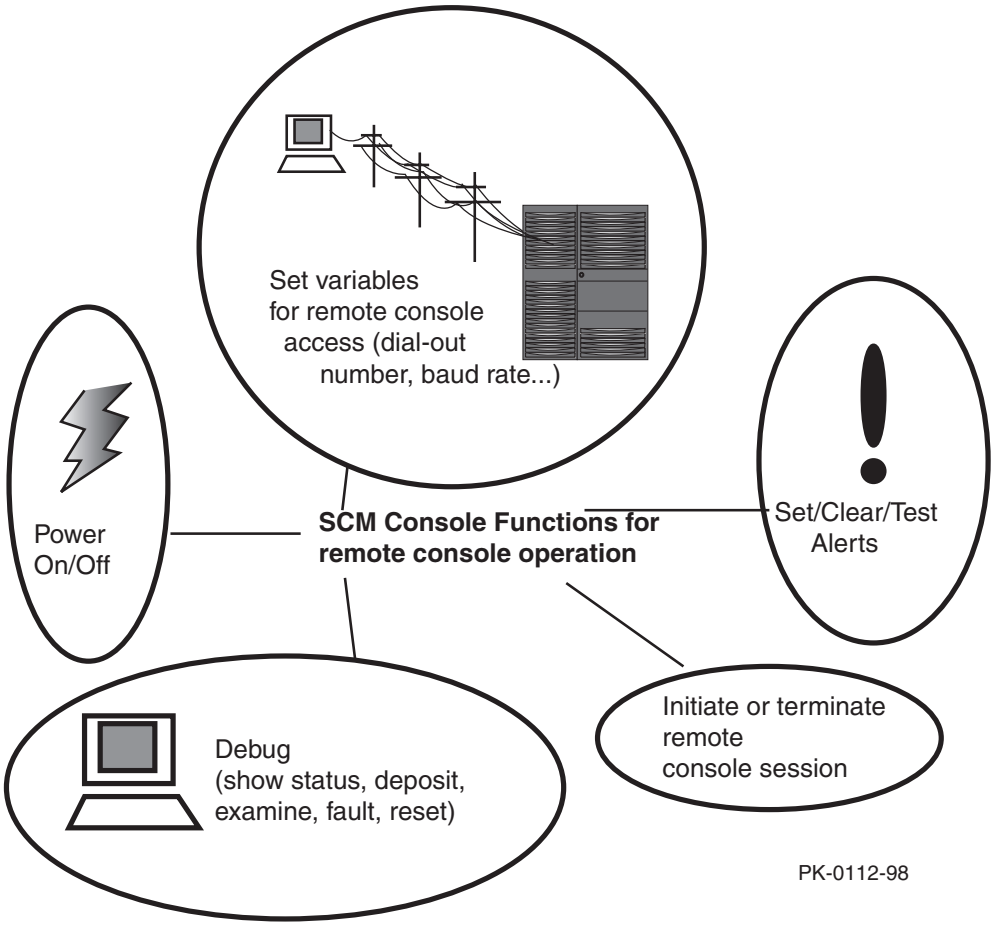
- Overview of Remote Setup and Operation
- Remote Operations Command Summary
- Displaying Help on SCM Console Commands
- Setting Up for a Remote Console
- Dialing In and Invoking SCM
- Transmission Modes
- Displaying System Status
- Controlling Power to the System
- Leaving the SCM Command-Line Interpreter
- Terminating the Remote Session

NOTE: *Chapter 4 tells how SCM commands define hard partitions. Chapter 6 describes the SCM commands in alphabetical order.*

3.1 Overview of Remote Setup and Operation

This section lists system management tasks to set up and use a remote console, the related SCM commands, and the place in this manual where the commands are described.

Figure 3–1 Overview of Remote Setup and Operation



One concept of SCM firmware design is that two different people will maintain *AlphaServer* GS80/160/320 systems. System monitoring is done by a *system manager*, who may be at a remote location in the next room, building, state, or even country. The system manager then may dispatch a *local service technician* to do on-site activities.

Remote Use of the SCM Monitor

The SCM firmware allows a remote user access to the system for general system monitoring and maintenance. The SCM firmware notifies the system manager of alert conditions by dialing out through a modem, usually to a pager held by the remote diagnostician on call.

To enter the SCM console remotely, the remote user dials in through a modem, enters a password, and then types a special escape sequence that invokes SCM command mode. (The system remains in whatever state it is in, including having the operating system running.) You must set up the remote modem before you can dial in remotely. (See Section 3.4.)

Local Use of the SCM Monitor

Although SCM commands can also be used from the local console terminal, there is no local notification when an alert condition occurs. A local user must initially define the dial-out number and set other parameters.

Also, the SCM provides the **power off** and **power on** commands that would be used by a service technician to power off or on the entire system, or a hard partition.

To enter the SCM locally, the user types the escape sequence on the local serial console terminal. (The default escape sequence is **<Esc><Esc>scm**.) The operating system state does not change upon entering SCM console mode.

Use of the SCM Monitor from the SRM Console

SCM commands can also be executed from a local or remote terminal from the SRM console by typing **scm** in front of the SCM command. For example:

```
P00>>> scm show csb
```

3.2 Remote Operations Command Summary

Table 3–1 lists the SCM commands used for setting up and using remote console terminals, and the sections where examples of the task can be found.

Table 3–1 SCM Commands Related to Remote Console Management

Task	Commands	Section
Get help on SCM commands	**help** or **?**	Section 3.3
Set up for remote console management	**set init** **set dial** **set alert** **set password** **enable remote** **enable alert** **test alert**	Section 3.4
Initialize the modem to prepare it to send alerts and receive incoming calls	**init** **set baud**	Section 3.4
Dial in and examine the state of the system	**clear alert** **show**	Section 3.5 Section 3.6
Low-order debugging	**examine** **deposit**	Chapter 6
Halt the system leaving error registers intact	**fault**	Chapter 6
Control power to the system	**power**	Section 3.8
Reset system or CSB component	**reset**	Chapter 6
Terminate the remote session	**hangup**	Section 3.10
Leave the SCM command-line interpreter	**quit**	Section 3.9

3.3 Displaying Help on SCM Console Commands

The help command displays information on the SCM commands. You can also type a question mark (?) to see this display.

Example 3–1 Help or ?

```
SCM_E0> help

               SCM CLI Version 1.0  2000/27/07

 build <fru> <pn> <sn> <mod> <ali> Build FRU Data(pn=2-5-2.4 sn=xxyzzabcde)
 build EEPROM                      Restore EEPROM RMC default values
 clear {alert, port}               Clear alert state, clear COM1 port comm
 deposit [-ipr,-spr(-iic,-offset),&mn, &p<csb_adr>,-next<n>,-q,-l,-w,-b] [data]
                                   Write data to a memory location or CSR/IPR
 disable {alert, remote, test<n>}  Disable remote dial-in, alert dial-out,test-n
 el                                Type out event log
 enable {alert, remote, test <n>}  Enable remote dial-in, alert dial-out, test-n
 erase                             Erase screen
 examine [-ipr, -spr, (-iic,-offset),&mn, &p<csb_adr>, -next <n>, -q, -l, -w, -b]
                                   Read a memory location or CSR/IPR
 fault [-all, -partition <n>]      Create a system wide FAULT condition
 halt {in, out} <-partition <n>,-all>Halt the system(virtual OCP halt button)
 hangup                            Terminate remote session
 help or ?                         Display list of SCM commands
 init                              Initialize the modem
 master <cli cmd>                  Slave SCM Command to master SCM
 power {on, off} [-all, -partition <n>]
                                   Power command
 quit                              Switch from SCM-CLI mode COM1 port
 reset [-all -partition <n>]       System level reset
 set flow {com1,local,modem} {hard,soft,both,none}
                                   Set a port's flow control characteristic
 set baud {com1,local,modem} <value> Set a port's baud rate
 set com1_mode {through,snoop,software_bypass,firm_bypass,local}
                                   Set COM1 operating mode
 show{csb,system,status,fru,nvr}   Show various system information
 test {alert, &pc<n> test_num}     Trigger an alert or run a CPU test
 update {-csb <node_address,…>     Update micro FLASH on CSB node(s)

SCM_E0>
```

The **help** command displays information about the commands available from the SCM firmware. You can also type a question mark (**?**) to display this information.

3.4 Setting Up for a Remote Console

A series of SCM commands must be issued from the local console terminal to prepare the system to automatically "call out" over a modem (to a pager, for example) when an alert condition occurs and to enable the remote user to dial in.

Example 3–2 Sample Setup for a Remote Console

```
P00>>> <Esc><Esc>scm ❶
SCM_E0> enable remote ❷
SCM_E0> set password newpassword ❸
SCM_E0> set init ate0v0&c1s0=2   ❹
SCM_E0> init   ❺
Querying the modem port…modem detected
Initializing modem…passed
SCM_E0> set dial atdt9,15551212    ❻
SCM_E0> set alert ,,,,5085551234#;   ❼
SCM_E0> enable alert  ❽
SCM_E0> test alert  ❾
```

To enable remote connections, a user at the local console must use the SCM command language to set initial parameters. Example 3–2 shows a sample sequence of SCM commands. A detailed description of each command is given in the SCM command reference chapter, Chapter 6.

❶ From the SRM console, type the escape sequence that returns you to the SCM console prompt. The default sequence is **<Esc><Esc>scm**.

❷ The **enable remote** command directs the SCM to accept incoming calls through the local modem. Note: the keyswitch on the operator control panel must not be in the Secure position, which prevents remote dial-in.

❸ The **set password** command requests that a remote user supply a password when dialing in. The **set password** command does not affect local access to the system. The default remote access password is **wffirmware**.

❹ The **set init** command gives the initialization string for the local modem. Currently supported modems and their initialization strings are:

Hayes Accura 336/56k	ate0v0&c1&k3s0=2
Compaq Microcom 510	ate0v0&c1s0=2

3Com USR Courier V.Everything	ate0v0&c1so=2
3Com USR 56K* (5676) (Japanese)	ate0v0&c1so=2
AT&T Dataport 14.4	ate0v0&c1so=2
MultiTech MT5600ZDS	ate0v0&c1so=2
Aiwa PV-BW5610 (Japanese)	ate0v0&c1so=2
Omrom ME5614D (Japanese)	ate0v0&c1so=2

❺ The **init** command initializes the modem. At this point, the modem is configured for remote dial-in and can be accessed through the modem.

❻ The **set dial** command specifies the dial-out string sent to the modem when an alert condition occurs. The dial string is case sensitive. The SCM automatically converts all alphabetic characters to uppercase. In the example, the following characters are used:

AT = Attention.
D = Dial
T = Tone (for touch-tone)
9 = Number for an outside line
, = Pause for 2 seconds
15551212 = Number to be dialed (usually a paging service)

❼ The **set alert** command defines the time to wait for the remote pager to respond, and a message passed to the pager (usually a telephone number for the remote user to dial in). In the example, the following values are used:

,,,, = Each comma (,) provides a 2-second delay. In this example, a delay of 8 seconds is set to allow the paging service to answer.

5085551234 The message displayed on the remote pager.

= Terminating character
; = Returns modem to command mode

❽ The **enable alert** command directs the SCM to monitor alert conditions and, when one occurs, to initiate the dial-out sequence defined by the **set init** and **set dial** commands issued previously.

❾ The **test alert** command directs the SCM to generate an alert condition and test the dial-out procedure. (The **show status** command reports on alert conditions detected by the SCM.)

3.5 Dialing In and Invoking SCM

To access a system from a remote console terminal, dial the modem, enter the modem password at the # prompt, and type the escape sequence. Use the hangup command to terminate the session.

Example 3–3 Sample Remote Dial-In Dialog

```
AT ❶
OK
ATDT15085551212
# password ❷
P00>>> <Esc><Esc>scm ❸
SCM_E0> show status ❹

    System Management Settings

RMC escape sequence      :    [ESC][ESC]scm
Local Baud/flow control  :    57600 / soft
COM1 Baud/flow control   :    57600 / hard
Modem Baud/flow control  :    57600 / hard
COM1 mode                :    Pass-through
OCP power switch         :    On
OCP halt                 :    Deasserted
OCP secure               :    Non-Secure
Remote Access            :    Enabled
Remote User              :    Connected
Remote status            :    Using SCM CLI
Alerts                   :    Enabled
Modem password           :    newpassword
Modem init string        :    ate0v0&c1s0=2
Modem dial string        :    atdt9,15551212
Modem alert string       :    ,,,,5085551234#
Alert pending            :    YES ❺
Most recent alert        :    User initiated…from CSB node e0
SCM_E0> clear alert ❻
Pending alert cleared
SCM_E0> hangup ❼
```

Example 3–3 shows a brief session from a remote console.

❶ AT = Attention. Standard modem response when ready.

❷ The remote user types the dial-in password set with the SCM command **set password**. The password is not displayed.

❸ After the password has been given, the SCM banner displays. The user is connected to the modem port, and is at whatever level the system or partition was operating at dial-in (the SCM console, the SRM console, or the operating system level.) You can type **<Esc><Esc>scm** to return to the SCM console.

❹ The **show status** command is a reasonable first command in the event you are dialing in to a system after receiving an alert on a pager. It shows current settings for remote access and the most recent alert.

❺ The most recent alert is shown to be user-initiated by a **test alert** command from CSB node e0.

❻ The remote user clears the alert, so that other alerts can be logged (such as overtemperature). The remote user would then log a call to the service technician to check on the condition causing the alert.

❼ The **hangup** command terminates the remote connection.

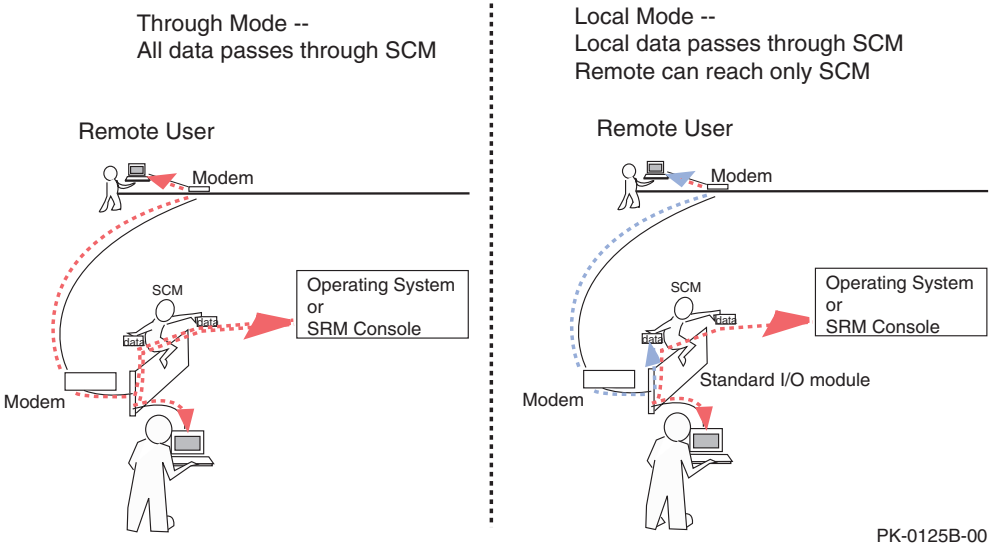
3.6 Transmission Modes

By default, all data transmissions between the local and remote console devices and the system pass through the SCM. You can select other modes of transmission.

3.6.1 Through Mode and Local Mode

In through mode, data passes between the local and remote console devices and the system through the SCM. In local mode, the remote console can only communicate with the SCM, not the system.

Figure 3–2 Through and Local Modes



Through Mode

Through mode is the default operating mode. The SCM routes every character of data between the system (the operating system or the SRM console) and the active external port, either the local port or the modem port. If a modem is connected, the data goes to the modem. The SCM filters the data for a specific escape sequence. If it detects the escape sequence, it enters the SCM CLI. This mode is illustrated by the left half of Figure 3–2.

If you wish to change from another transfer mode to local mode, you can use the SCM **set com1_mode through** command.

Example 3–4 Set com1_mode through Command

```
SCM_E0> set com1_mode through
```

Local Mode

In local mode, only the local port can communicate with the system; the modem is prevented from sending characters to the system, but a remote user can still enter the SCM from the modem. This mode is illustrated by the right half of Figure 3–2.

Example 3–5 Set com1_mode local Command

```
SCM_E0> set com1_mode local
```

3.6.2 Bypass Modes

The bypass modes determine how much part the SCM plays in data transfers between the system and the local or remote console device.

Figure 3–3 Bypass Modes

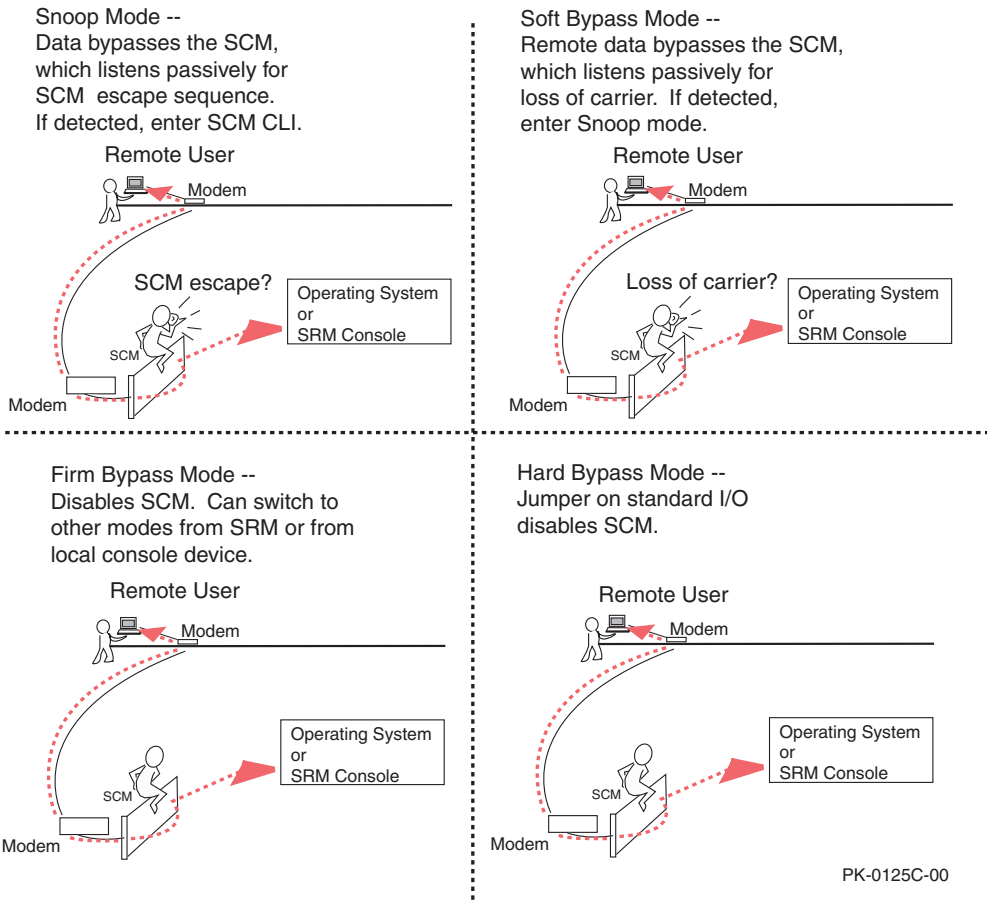


Figure 3–3 shows the data flow in the various bypass modes. Note that the local terminal is still connected to the SCM and can still enter the SRM to switch the **com1_mode** to alter the bypass mode, if necessary. In any of the bypass modes, when the system loses power, the default is snoop mode.

Snoop Mode

In snoop mode, all data bypasses the SCM as it passes between the system and the remote console device. The SCM taps into the data lines and listens passively for the SCM escape sequence. If it detects the escape sequence, it enters the SCM CLI. This mode is selected by the command **set com1_mode snoop.**

The escape sequence is also passed to the system. If you decide to change the default escape sequence, be sure to choose a unique sequence so that the system software does not interpret characters intended for the SCM.

In snoop mode, the SCM is responsible for configuring the modem for dial-in as well as dial-out alerts. It also responsible for monitoring modem connectivity.

Snoop mode is useful when you want to monitor the system but also ensure optimum transfer speed between the system and the remote console device.

Soft Bypass Mode

In soft bypass mode, the SCM is again bypassed, but it listens passively for loss of carrier. If the SCM detects loss of carrier, it switches automatically into snoop mode. If you have set up the dial-out alert feature, the SCM pages the operator if an alert is detected and the modem line is not in use. This mode is selected by the command **set com1_mode software_bypass.**

Soft bypass mode is useful if system management applications need to perform a remote binary download, because it ensures that the SCM does not accidentally interpret some binary data as the escape sequence. After downloading binary files, you can set the **com1_mode** environment variable from the SRM console to switch back to snoop mode or other modes, or you can hang up the current modem session and reconnect it.

Firm Bypass Mode

In firm bypass mode, the SCM is disabled. The SCM does not configure or monitor the modem, and the SCM dial-in and call-out features are disabled. You select this mode with the command **set com1_mode firmware_bypass.**

This mode is useful if you want the system, not the SCM, to control the modem port and you want to disable SCM remote management features. You can switch to other modes by resetting the SRM **com1_mode** environment variable.

Hard Bypass Mode

In this mode, the SCM is disabled by a jumper on the standard I/O module.

3.7 Displaying System Status

The show command displays the status of the requested components of the system. Examples of the three "systemwide" show commands are given in this section. See Chapters 6 and 7 for SCM and SRM commands that display system and/or partition status.

3.7.1 Show CSB

The show csb command displays a list of all known nodes on the console serial bus.

Example 3–6 Show CSB Command

```
SCM_E0> show csb    ❶

❷  ❸              ❹                        ❺           ❻          ❼
CSB Type          Firmware Revision        FSL Revision  Power State

10  PBM           X05.7   (06.12/02:16)    X5.7 (06.09)  ON
11  PBM           X05.7   (06.12/02:16)    X5.7 (06.09)  ON
12  PBM           X05.7   (06.12/02:16)    X5.7 (06.09)  ON
13  PBM           X05.7   (06.12/02:16)    X5.7 (06.09)  ON
15  PBM           X05.7   (06.12/02:16)    X5.7 (06.09)  ON
16  PBM           X05.7   (06.12/02:16)    X5.7 (06.09)  ON
30  PSM           X05.7   (06.12/02:16)    X5.7 (06.09)  ON          SrvSw: NORMAL
30    XSROM       X05.7   (06.12/02:16)
c0    CPU0/SROM   V6.0-7                                 ON
c1    CPU1/SROM   V6.0-7                                 ON
c2    CPU2/SROM   V6.0-7                                 ON
c3    CPU3/SROM   V6.0-7                                 ON
c0    IOR0                                               ON
c1    IOR1                                               ON
c2    IOR2                                               ON
c3    IOR3                                               ON
31  PSM           X05.7   (06.12/02:16)    X5.7 (06.09)  ON          SrvSw: NORMAL
31    XSROM       X05.7   (06.12/02:16)
c4    CPU0/SROM   V6.0-7                                 ON
c5    CPU1/SROM   V6.0-7                                 ON
c6    CPU2/SROM   V6.0-7                                 ON
c7    CPU3/SROM   V6.0-7                                 ON
c4    IOR0                                               ON
c5    IOR1                                               ON
c6    IOR2                                               ON
c7    IOR3                                               ON
```

```
32  PSM           X05.7   (06.12/02:16)  X5.7 (06.09)  ON           SrvSw: NORMAL
32    XSROM       X05.7   (06.12/02:16)
c8    CPU0/SROM   V6.0-7                               ON
c9    CPU1/SROM   V6.0-7                               ON
ca    CPU2/SROM   V6.0-7                               ON
cb    CPU3/SROM   V6.0-7                               ON
c8    IOR0                                             ON
c9    IOR1                                             ON
33  PSM           X05.7   (06.12/02:16)  X5.7 (06.09)  ON           SrvSw: NORMAL
33    XSROM       X05.7   (06.12/02:16)
cc    CPU0/SROM   V6.0-7                               ON
cd    CPU1/SROM   V6.0-7                               ON
ce    CPU2/SROM   V6.0-7                               ON
cf    CPU3/SROM   V6.0-7                               ON
cc    IOR0                                             ON
cd    IOR1                                             ON
40  HPM           X05.7   (11.03/01:11)  T4.2 (09.08)  ON
e0  SCM MASTER    X05.7   (11.03/01:13)  T4.2 (09.08)  ON
e1  SCM SLAVE     X05.7   (11.03/01:13)  T4.2 (09.08)  ON           Ineligible
e2  SCM SLAVE     X05.7   (11.03/01:13)  T4.2 (09.08)  ON           Ineligible
e6  SCM SLAVE     X05.7   (11.03/01:13)  T4.2 (09.08)  ON           Ineligible
```

❶ The user enters the **show csb** command.

❷ The possible addresses on the console serial bus run from 10-1F (PBM), 30-37 (PSM), 40 (HPM), and E0-EF (SCM). This column identifies the address (hexadecimal) of each node on the CSB. You can use this address to identify PCI subsystems (for example, in a **power off** command) to replace a module in such a subsystem without powering off the entire system.

❸ This column identifies the type of module at each address.

❹ This column identifies the firmware revision level of the firmware code residing at this address.

❺ This column identifies the firmware revision level of the fail-safe loader residing at this address.

❻ This column gives the power state of the CSB node. In the example, all nodes are On. The column would show Off for nodes whose power had been removed for a hot or warm swap operation.

❼ This column lists the service status of the node; NORMAL means that power is being supplied to the node. SERVICE means power is not being supplied to the node; it is ready for removal and replacement operations by field service. This column also lists whether an SCM node is eligible to serve as SCM master in a partitioned system. See Chapter 4 for more information on partitions and master and slave SCMs.

3.7.2 Show System Command

The show system command displays an overall system summary.

Example 3–7 Show System Command

```
SCM_E0> show system
System Primary QBB0 : 1                    ❶
System Primary CPU  : 1 on QBB1
❷    ❸      ❹    ❺         ❻                  ❼ ❽   ❾  ❿   ①

Par hrd/csb CPU  Mem   IOR3 IOR2 IOR1 IOR0  GP  QBB  Dir PS  Temp
     QBB#   3210 3210     (pci_box.rio)     Mod BP   Mod 321 (ºC)

(3) 0/30   --PP ---P  Px.x P2.0 Pf.1 Pf.0   P   P    P  PPP  28.0
(0) 1/31   --PP --PP  P4.0 Px.x Px.x Px.x   P   P    P  PPP  30.0
(2) 2/32   --PP ---P  Px.x P0.0 Px.x P3.0   P   P    P  PP-  26.5
(1) 3/33   --PP ---P  --.- --.- Px.x P5.0   P   P    P  PP-  28.1

HSwitch  Type    Cables 7 6 5 4 3 2 1 0    Temp(ºC)
                                ②          ③
HPM40    4-port          - - - - P P P P     26.0
④          ⑤                                ⑥   ⑦   ⑧
PCI Rise1-1  Rise1-0  Rise0-1  Rise0-0   RIO  PS  Temp
Cab 7 6 5 4   3 2 1   7 6 5 4   3 2 1    1 0  21  (ºC)

10  - - - -   - - -   - L - L   - - S    - P  -P  32.5
12  - - - -   - - -   - - - -   - L S    - P  P-  32.5
13  - - - -   - - -   - - L -   - - S    - P  P-  32.5
14  - - - -   - - -   - - L -   L - S    - P  PP  32.5
15  - - - -   - - -   L - - -   - - S    - P  PP  32.5
1f  - - - -   - - -   - - - H   L - S    P P  PP  32.5
```

Example 3–7 shows a **show system** command for a 4-QBB system.

❶ The display shows the system primary QBB and CPU.
❷ The hard partition number (if no hard partitions, a dash is displayed).
❸ The hard QBB number (defined by the location of the QBB; see Section 2.4) and the CSB address for the power system manager (PSM) for the QBB.
❹ CPU self-test results, identified by the slot number in the QBB. Note that not all QBBs are fully populated with four CPU modules.
❺ Memory self-test results, identified by the slot number in the QBB. Note that each QBB has two CPUs and one or two memory modules.

❻ Status of I/O risers in the QBB.
  P passed and powered on
  p passed and powered off
  m.n m= PCI box (0-f), n = remote I/O riser (0,1)
  x.x nothing is connected
❼ Global port self-test results for the QBB.
❽ QBB backplane self-test results.
❾ Directory module self-test results for the QBB.
❿ Status of the power supplies to the QBB. Note that QBBs 0 and 1 have redundant power supplies, while QBBs 2 and 3 do not.
① Temperature of the QBB.
② This section of the report gives the status of the hierarchical switch. Possible values are
  P passed
  V valid; present, but may not be turned on
  * present
  - not present
③ Temperature surrounding the H-switch.
④ The number of the PCI cabinet for which information is displayed. These numbers are defined by the node ID switch setting.
⑤ The status of the slots in the PCI card cages connected to the I/O risers.
  S standard I/O module
  * present
  - not present

  L Low power dissipation
  M Medium power dissipation
  H High power dissipation

⑥ Status of remote I/O riser P = present, - = not present.
⑦ Status of power supplies to the PCI box.
  P passed and powered on
  p passed and powered off
  F failed and powered on
  f failed and powered off
  * present but no status yet
  - not present

⑧ Temperature of the PCI box.

3.7.3 Show FRU Command

The show FRU command lists the field-replaceable units (FRUs) in the system.

Example 3–8 Show FRU Command

```
SLV_E2>  show fru

    ❶              ❷                ❸          ❹          ❺
FRUname          E  Part#            Serial#    Model/Other Alias/Misc
PBP0             02 00-00000-00.A00  NI00000000 FRU         REG
PBP0.SIO         00 54-B4190-BA.A01  NI91960002 OB5QBB1
PBP0.RIO0        00 B4171-AA.D02     NI93473157
PBP1             42 54-25027-01.A02  NI23234454 OB5QBB1
PBP1.SIO         00 B4190-BA.B01     NI93160206
PBP1.RIO0        00 B4171-AA.D02     NI93473157
PBP2             02 00-00000-00.A00  NI00000000 FRU         REG
PBP2.SIO         00 54-B4190-BA.A01  NI91960002 OB5QBB1
QBB5             00 54-25043-02.D01  NI92460907 OB5QBB1
QBB5.PSM         00 54-25074-01.E02  NI91860246 OB5QBB1
QBB5.PWR         00 54-25017-01.D01  NI90665951 OB5QBB1
QBB5.AUX         00 54-25123-01.D03  NI93060053 ..........  ..............
QBB5.CPU0        00 B4115-AA.B02     NI91360068 EV6         OB2QB3
QBB5.CPU1        00 54-B4115-AA.B02  NI91360077 OB5QBB1
QBB5.MEM1        00 54-B4150-AA.C01  NI90863268 OB5QBB1
QBB5.MEM1.DIM0   00 54-24941-BA. C02 AY305#J0
QBB5.MEM1.DIM1   00 54-24941-BA. C02 AY305&J0
QBB5.MEM1.DIM2   00 54-24941-BA. C02 AY305=K0
QBB5.MEM1.DIM3   00 54-24941-BA. C02 AY305<K0
QBB5.MEM1.DIM4   00 54-24941-BA. C02 AY305XI0
QBB5.MEM1.DIM5   00 54-24941-BA. C02 AY305DJ0
QBB5.MEM1.DIM6   00 54-24941-BA. C02 AY305CJ0
QBB5.MEM1.DIM7   00 54-24941-BA. C02 AY305BJ0
QBB5.IO01        00    -B4170-AA.D01 NI93871029 ..........  ..............
QBB5.IO23        00    -B4170-AA.D01 NI93870976 ..........  ..............
QBB5.GP          00 54-30330-02.A01  NI91360395 OB5-F       FAKEGP
SLV_E2>
```

Example 3–8 shows a **show fru** command issued from a local or remote console terminal connected to a particular hard partition. (See Chapter 4 for a discussion of partitions.)

❶ The name of the FRU:

PBP	PCI backplane
SIO	Standard I/O module
RIO	Remote I/O riser (located in PCI box)
QBB	Quad building block
PSM	Power system manager module
PWR	Main power module
AUX	Auxiliary power module
CPU	CPU module
MEM	Memory module
DIM	DIMM memory array
IO	Local I/O riser (located in QBB)
GP	Global port module

❷ Part number by which the FRU can be ordered.

❸ The serial number of the FRU.

❹ Optional manufacturing specifications; usually blank.

❺ Optional manufacturing specifications; usually blank.

3.8 Controlling Power to the System

Once you have determined the cause of a system problem, you may want to power off the system or a particular component of the system. This can be done remotely, but unless the situation is catastrophic, it is probably best to let the local service technician handle power on and power off.

Example 3–9 Power Off Command

```
SCM_E0> show system
System Primary QBB0 : 1
System Primary CPU  : 1 on QBB1

Par hrd/csb CPU  Mem   IOR3 IOR2 IOR1 IOR0  GP  QBB  Dir PS  Temp
     QBB#   3210 3210     (pci_box.rio)     Mod BP   Mod 321 (ºC)
                                                ❶
(3) 0/30    --PP ---P  Px.x P2.0 Pf.1 Pf.0   P   F    P  PPP  28.0
(0) 1/31    --PP --PP  P4.0 Px.x Px.x Px.x   P   P    P  PPP  30.0
(2) 2/32    --PP ---P  Px.x P0.0 Px.x P3.0   P   P    P  PP-  26.5
(1) 3/33    --PP ---P  --.- --.- Px.x P5.0   P   P    P  PP-  28.1

HSwitch  Type    Cables 7 6 5 4 3 2 1 0    Temp(ºC)
HPM40    4-port         - - - - P P P P     26.0

PCI Rise1-1  Rise1-0  Rise0-1  Rise0-0   RIO  PS  Temp
Cab 7 6 5 4   3 2 1   7 6 5 4   3 2 1    1 0  21  (ºC)

10  - - - -   - - -   - L - L   - - S    - *  -P  32.5
12  - - - -   - - -   - - - -   - L S    - *  P-  32.5
13  - - - -   - - -   - - L -   - - S    - *  P-  32.5
14  - - - -   - - -   - - L -   L - S    - *  PP  32.5
15  - - - -   - - -   L - - -   - - S    - *  PP  32.5
1f  - - - -   - - -   - - - H   L - S    * *  PP  32.5

SCM_E0> power off -par 0,3  ❷
```

❶ The **show system** command reveals a QBB backplane as failing.

❷ The user removes power from hard partitions 0 and 3 (hard QBBs 0 and 1) in preparation for replacing a system box, leaving the remaining partitions up and running.

3.9 Leaving the SCM Command-Line Interpreter

It is often useful in debugging to leave the SCM command-line interpreter and return to the SRM interpreter. The quit command exits the SCM and returns control to the COM1 port, through which the SRM firmware communicates.

Example 3–10 Using Quit to Return to SRM

```
SCM_E0> quit    ❶
Returning to COM1 port.
P00>>> boot  ❷

  [system boots and displays prompt]
```

❶ The **quit** command does not terminate the modem connection; rather, it transfers I/O operations from the SCM to the COM1 port, where the user can issue SRM commands.

❷ The remote user boots the operating system to return the system to work.

3.10 Terminating the Remote Session

The hangup command terminates the remote session, disconnecting the remote modem and returning I/O to the COM1 port.

Example 3–11 Using the Hangup Command

```
SCM_E0>  hangup    ❶
SCM_E0>    ❷
```

❶ The **hangup** command terminates the remote connection.

❷ In this case, the **hangup** command was issued by a user at the local console, so the SCM prompt appears. The **hangup** command can be issued either by the remote user or at the local console device.

Chapter 4
Partitions

Partitions are defined as individual subsets of a system's computing resources, each of which is capable of running a copy, or *instance*, of an operating system. This chapter describes hard and soft partitions and the firmware used to define them. Sections include:

- Maintaining Partitions Locally
- Hard and Soft Partitions
  - Hard Partitions
  - Soft Partitions
- Hardware Requirements
- Defining Hard Partitions
  - SCM Commands That Define Hard Partitions
  - Detailed Directions for **hp_qbb_mask**
- Defining Soft Partitions
  - SRM Commands That Define Soft Partitions
  - Detailed Directions for **lp_io_mask**
  - Detailed Directions for **lp_cpu_mask**
  - Detailed Directions for **lp_mem_size**
  - Detailed Directions for **lp_shared_mem_size**
- **Lpinit** Command
- Soft Partitions Within Hard Partitions
- Master and Slave SCMs
- SCM Functionality from Master and Slave SCMs

4.1 Maintaining Partitions Locally

Partitions are monitored at the local site using the system management console (SMC).

Figure 4–1 Maintaining Multiple OS Instances at the SMC

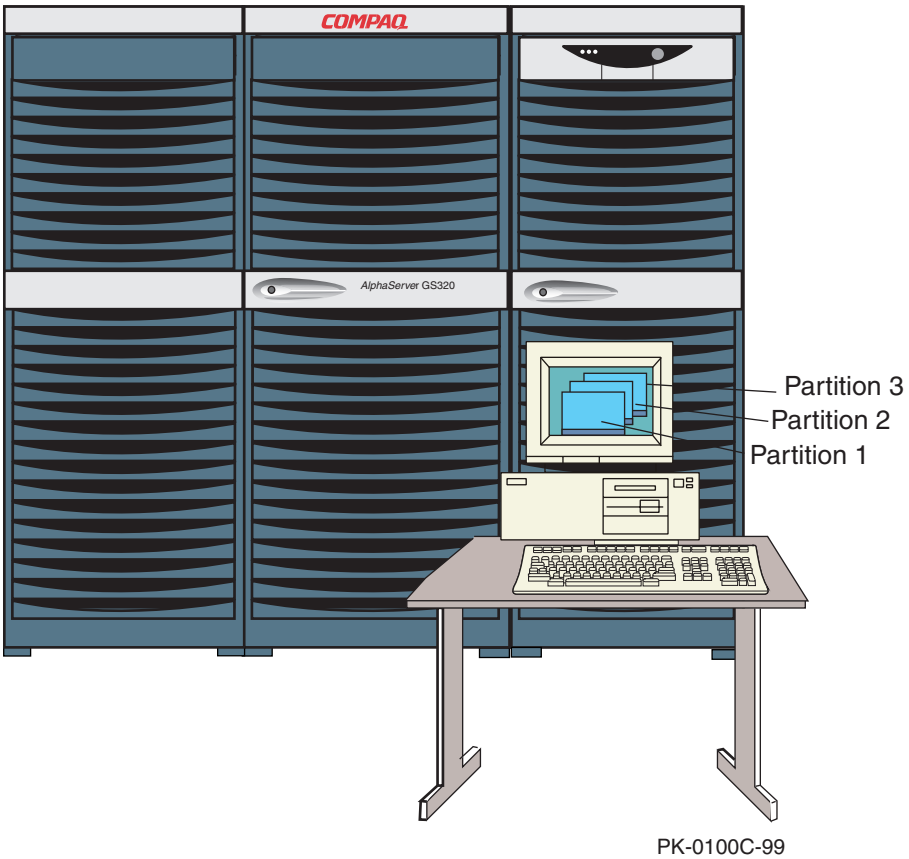


Figure 4–1 shows a GS320 system with three partitions, each maintained at the system management console by a separate invocation of a terminal emulator.

The system management console may also be used from remote locations, or the system administrator can dial in from a remote serial terminal or from a system using terminal emulator software. See the *AlphaServer GS80/160/320 System Management Console Installation and User's Guide* for information about this console device.

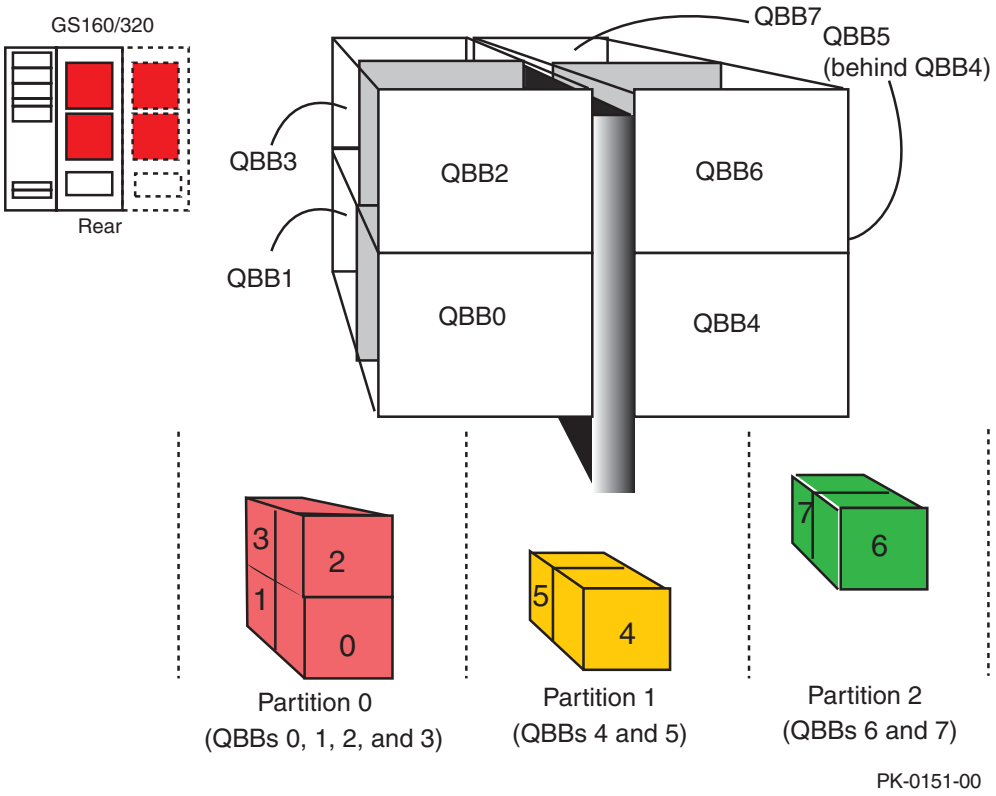
4.2 Hard and Soft Partitions

The firmware supports two types of partitioning: hard and soft.

4.2.1 Hard Partitions

Hard partitions are defined by naming the QBBs in each partition.

Figure 4–2 Hard Partitions



As shown in Figure 4–2, hard partitions are defined along QBB boundaries. The figure shows a GS320 system with eight QBBs divided into three hard partitions.

There is no sharing of CPU, memory, or I/O resource between hard partitions. The boundaries of these partitions are considered "hard." An instance of an operating system may run in each hard partition; these instances run completely independently of each other.

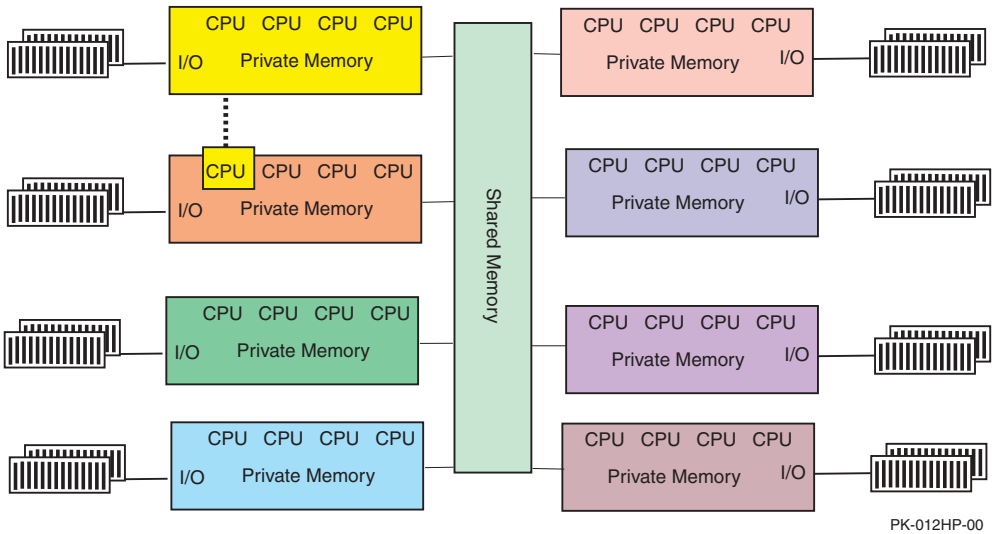
Tru64 UNIX instances do support individual applications using a specified percentage of a given resource, such as processor sets. This type of resource partitioning is described in detail in the *Tru64 UNIX System Management* manual.

OpenVMS implements resource sharing differently, building on SRM firmware-based entities called *soft partitions*, discussed next.

4.2.2 Soft Partitions

Soft partitions are defined by naming the specific CPUs, memory allocation, and I/O modules comprising each partition. CPUs can be moved from one instance to another as computation load changes. Also, soft partitions allow instances to share memory. Only *OpenVMS* Galaxy is an implementation of soft partitions.

Figure 4–3 Soft Partitions



Soft partitions allow a portion of memory to be defined as shared memory; they also allow CPUs to be reassigned by agreement of the operating system instances running in the partitions. The boundaries defined by the firmware are thus considered "soft."

Figure 4–3 shows a GS320 system with eight soft partitions set up on QBB boundaries. Each partition has private memory. In addition, you can allocate shared memory that all instances can use, as shown in the center of the diagram.

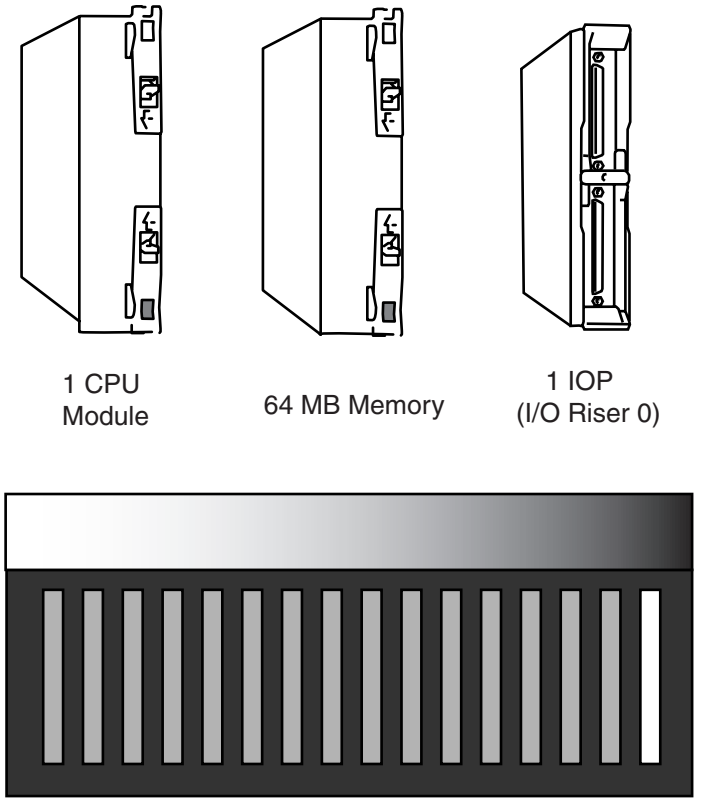
In addition to shared memory, soft partitions differ from hard partitions in that, with soft partitions, operating system instances may agree to share CPU resources. CPUs can be reassigned to different instances as load varies.

Soft partitions are defined using the SRM firmware. The division of resources defined can be thought of as a starting point. Once an operating system instance is running in each soft partition, resources can be reassigned as needed at the operating system level. See the *OpenVMS Alpha Galaxy and Partitioning Guide* for more information on this type of resource sharing.

4.3 Hardware Requirements

A partition requires at least one CPU, one memory, an I/O module, and a master PCI box with a standard I/O module.

Figure 4–4 Hardware Requirements for Each Partition



A hard or soft partition requires at least one CPU (resident in the QBB with I/O) and 64 Mbytes of memory. I/O requirements are at least one local I/O riser (IOR0) be present per partition connected to a master PCI box. A master PCI box is one with a DVD/CD-ROM drive and keyboard and mouse ports, etc., and contains a standard I/O module with local and modem ports for connection to a console device. (Two I/O risers are supported by one QBB; if both are present in a QBB, they cannot be split between partitions.)

With hard partitions, at least one QBB is required per partition. Soft partitions allow more flexible assignment of resources; however, the two I/O riser modules in a QBB cannot be split between partitions.

The maximum number of partitions varies according to the configuration. With hard partitions, a two-drawer GS80 supports two hard partitions. A partition may include more than one QBB. For example, an 8-QBB GS320 system could have two partitions, one with five QBBs and one with three QBBs.

With soft partitions, a 2-QBB system is still limited to only two soft partitions while an 8-QBB GS320 system supports up to eight soft partitions. The number of soft partitions is limited by the constraint that the two I/O riser modules in a QBB cannot be split between partitions.

Each partition must include a master PCI box with a standard I/O module. The standard I/O module contains both SRM firmware and SCM firmware. Hard partitions are defined using the SCM firmware. Soft partitions are defined using the SRM firmware.

4.4 Defining Hard Partitions

4.4.1 SCM Commands That Define Hard Partitions

Use the SCM set command for the hp_count and hp_qbb_maskn environment variables, and the power on command for initialization.

Figure 4–5 SCM Commands for Hard Partitions

SCM environment variables

define:

1. Number of partitions:
   hp_count (0, 2, 3, ... 8)

```
SCM_E0> set hp_count 3
```

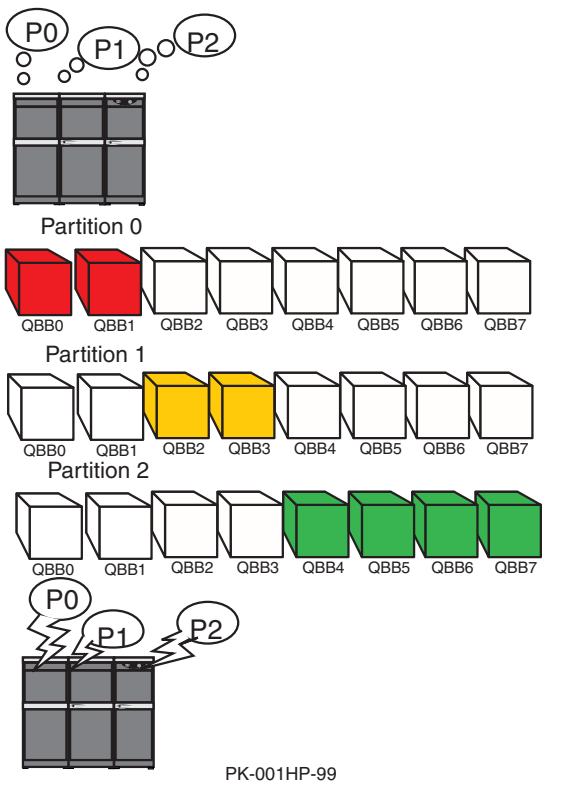
2. QBBs in each partition:
   hp_qbb_mask* (bit mask)

```
SCM_E0> set hp_qbb_mask0 = 3
SCM_E0> set hp_qbb_mask1 = C
SCM_E0> set hp_qbb_mask2 = F0
```

Power on commands

initialize partitions defined:

```
SCM_E0> power on -all
```



At the SCM prompt, you set a value for the **hp_count** environment variable to define the number of hard partitions on your system, and values for **hp_qbb_mask*n*** environment variables to define the QBBs in each partition. Figure 4–5 and Table 4–1 give an overview of the environment variables and what they do. Environment variable settings for the **hp_qbb_mask*n*** are discussed in the next section.

Table 4–1 SCM Environment Variables for Hard Partitions

Environment Variable	Definition
hp_count *n*	The number of hard partitions you wish to create on the system. Possible values are 0 (partitions disabled), 1 (for one partitions), 2 (for two partitions), 3 (for three partitions), and so forth, to a maximum of 8.
hp_qbb_mask* *x*	For the *, you supply the number of the partition, which may be 0, 1, 2, up to 7. Partition numbers must be in sequence; for example, you could not have partitions 0, 3, and 8 only.)The value *x* gives a binary mask indicating which QBBs you want included in the partition. See Section 4.4.2 for detailed examples.

A **power on –par *n*** command supplies power to each partition, which goes through its own power-up process, displayed at the partition's console window. You can also do a **power on –all** command to initialize all partitions as shown in Figure 4–5.

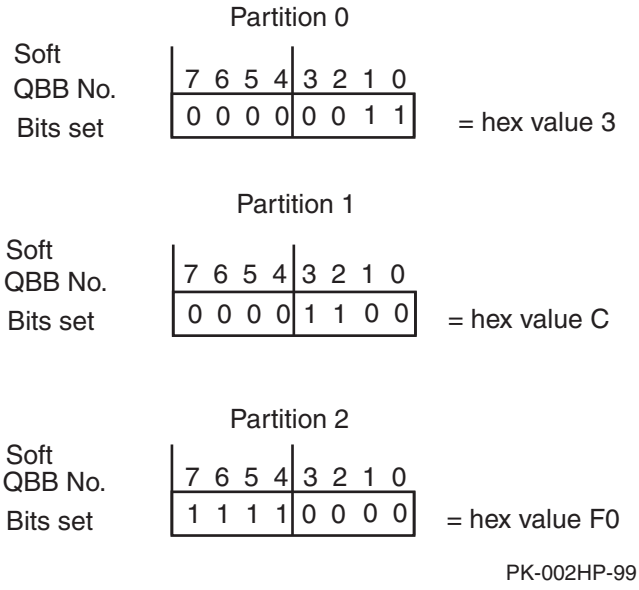
If you are defining partitions on a system that is already powered on, issue a **reset –all** command to initialize the system with the new partition definitions.

Once a partition is powered on or reset, you will notice that **show** displays reflect CPUs, CSB nodes, adapters, and devices that are numbered beginning with 0 for each hard partition. The only exception is the hard QBB number, which remains as though the system were not partitioned. Any commands that use the CPU number, CSB node address, adapter number, or device number must reflect the soft numbering given in the **show** commands for the individual partition.

4.4.2 Detailed Directions for hp_qbb_mask

The hp_qbb_mask environment variable is set to a value that creates a binary mask in which a bit set to 1 indicates that an individual QBB belongs to a partition.

Figure 4–6 Sample QBB Bit Masks



QBB modules can be assigned to partitions in any order, although there are considerations for maximum efficiency and reliability, as described in the manual *Getting Started with Partitions*. Bit values for one nibble of data are as follows:

Hex	Binary	Hex	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Example 4–1 shows the SCM commands used to create hard partitions for the configuration shown in Figure 4–5.

Example 4–1 Sample Hard Partition Setup

```
SCM_E0> set hp_count 3
SCM_E0> set hp_qbb_mask0 3
SCM_E0> set hp_qbb_mask2 C
SCM_E0> set hp_qbb_mask1 F0
SCM_E0> power on -all
Powering on Hard_partition_0 consisting of
      QBB0
      QBB1
 Powering on PCI Box 0
 Powering on PCI Box 1
     .
     .
     .
```

4.5 Defining Soft Partitions

4.5.1 SRM Commands That Define Soft Partitions

Soft partitions are defined by setting environment variables that define the number of partitions, as well as the CPUs, I/O risers, and memory sizes of each partition, the size of shared memory, and the partition to receive error interrupts. Soft partitions are initialized with the lpinit command.

Figure 4–7 Soft Partitions

SRM environment variables

define:

1. Number of partitions:
   lp_count (0, 1, 2, 3, ... 8)

```
P00>>>> set lp_count 3
```

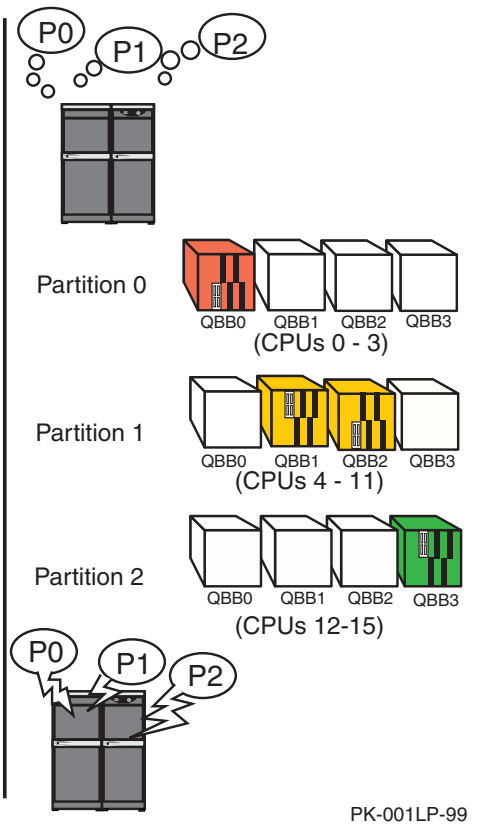
2. Resources in each partition:

```
P00>>> set lp_io_mask0 1
P00>>> set lp_cpu_mask0 F
P00>>> set lp_mem_size0 6GB

P00>>> set lp_io_mask1 6
P00>>> set lp_cpu_mask1 FF0
P00>>> set lp_mem_size1 12GB

P00>>> set lp_io_mask2 8
P00>>> set lp_cpu_mask2 F000
P00>>> set_mem_size2  10GB

P00>>> set lp_shared_mem_size 4GB
P00>>> set lp_error_target 0
```

Lpinit command

initialize partitions defined:

```
P00>>> lpinit
```



You set values for environment variables to define the number of partitions, the CPU, memory, and I/O resources belonging to each partition, the amount of shared memory, and the partition to receive error interrupts. Figure 4–7 and Table 4–2 give an overview of the environment variables and what they do. Succeeding sections describe the more complex environment variables and the **lpinit** command, which initializes soft partitions.

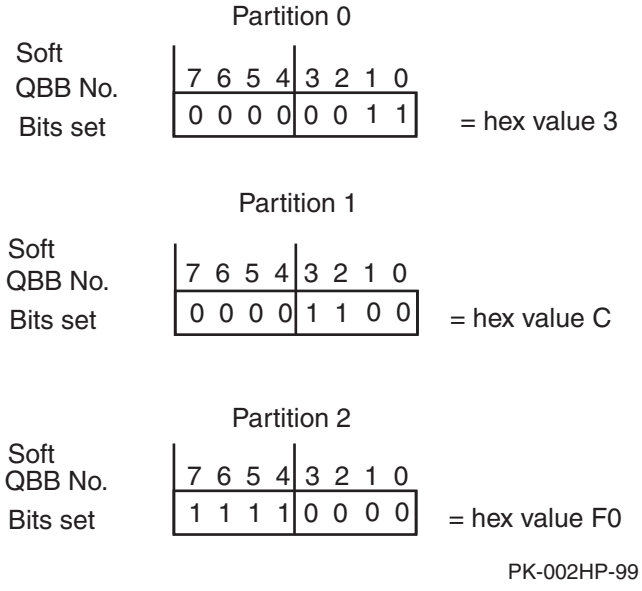
Table 4–2 SRM Environment Variables for Soft Partitions

Environment Variable	Definition
lp_count *n*	The number of soft partitions to create. Possible values: 0 Default. All IOPs, CPUs, and memory are assigned to one soft partition. No shared memory is defined. 1 One soft partition is created (partition 0). 2-8 From two to eight soft partitions can be defined.
lp_io_mask* *x*	For the *, supply the partition number (0 – 7). The value *x* gives a binary mask indicating the soft QBB numbers (as displayed by **show config**) for QBBs (containing I/O risers) to be included in the partition. See Section 4.5.2 for details.
lp_cpu_mask* *x*	For the *, supply the partition number. The value *x* gives a binary mask indicating which CPUs (using soft CPU numbers as displayed by **show config**) are part of the partition. See Section 4.5.3 for details.
lp_mem_size* *n*	For the *, you supply the number of the partition. The value *n* can be a size or list of sizes. See Section 4.5.4 for details.
lp_shared_mem_size *n*	Size of memory shared by all partitions. See Section 4.5.5 for details.
lp_error_target	The number of the soft partition that is to receive error interrupts. The primary CPU of this soft partition will receive the interrupts. If **lp_count** is 0, **lp_error_target** is ignored.

4.5.2 Detailed Directions for lp_io_mask

The lp_io_mask environment variable is set to a value that creates a binary mask in which a bit set to 1 indicates that an individual IOP (same as soft QBB number) belongs to a soft partition.

Figure 4–8 Sample lp_io_mask Bit Masks



I/O ports (IOPs), defined by the soft QBB number for the QBB in which they reside, can be assigned to soft partitions in any order. Soft QBB numbers are displayed by a **show config** command. Each soft partition must contain at least one IOP with an I/O riser connected to a master PCI box with a standard I/O module.

Bit values for one nibble of data are as follows:

Hex	Binary	Hex	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Example 4–2 Sample lp_io_mask Setup

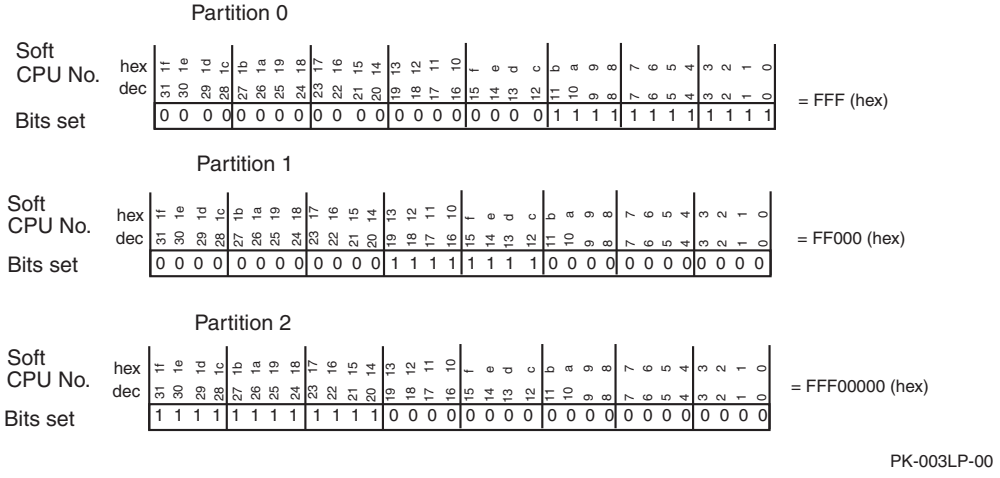
```
P00>>> set lp_io_mask0 3
P00>>> set lp_io_mask1 2C
P00>>> set lp_io_mask2 F0
```

In Example 4–2, the first command defines the IOPs in soft QBBs 0 and 1 as part of soft partition 0. The second command defines the IOPs in soft QBBs 2 and 3 as part of soft partition 1. The third command defines the IOPs in soft QBBs 4, 5, 6, and 7 as part of soft partition 2.

4.5.3 Detailed Directions for lp_cpu_mask

The lp_cpu_mask environment variable is set to a value that creates a binary mask in which a bit set to 1 indicates that an individual CPU belongs to a soft partition.

Figure 4–9 Sample lp_cpu_mask Bit Masks for 32-Processor System with No Hard Partitions



PK-003LP-00

Figure 4–9 shows some examples of bit masks for the **lp_cpu_mask** environment variable on a fully populated GS320 system (up to 32 processors) with no hard partitions. Both hexadecimal and decimal are given for bit place values.

Note that if you are defining a soft partition within a hard partition, the SCM firmware, when the hard partition is powered on, supplies new "soft" CPU numbers from 0 at the lowest-numbered hard QBB within the partition and proceeding upward. An SRM **show config** display for the partition will show the soft CPU numbers. An example of how to construct CPU numbers given the soft QBB numbers in the partition is given in Section 4.6.

You can use the table on page 4-17 to construct values for as many hexadecimal digits as you need to form a bit mask.

Example 4–3 shows the SRM commands used to create **lp_cpu_mask** environment variables for the configuration shown in Figure 4–9.

Example 4–3 Sample lp_cpu_mask Setup

```
P00>>> set lp_cpu_mask0 FFF
P00>>> set lp_cpu_mask1 FF000
P00>>> set lp_cpu_mask2 FFF00000
```

4.5.4 Detailed Directions for lp_mem_size

The lp_mem_size environment variable is set to a value, or a list of values, that define the amount of memory allocated to a soft partition.

Example 4–4 Example for Constructing lp_mem_size

```
P00>>> show config ❶

                  Compaq AlphaServer GS320 6/731

QBB 0         Hard QBB 0
 Quad Switch                              QSA rev 2, QSD revs 0/0/0/0
   .
   .
   .
QBB     Size    Interleave                System Memory 64 GB  ❷
  0      16 GB     32-Way  ❸
  1      16 GB     32-Way
  2      16 GB     32-Way
  3      16 GB     32-Way
```

As shown in Example 4–4, the SRM command **show config** (see ❶) displays the system memory size (see ❷). From this number, you assign the amount of memory in each soft partition. The total amount of memory for all partitions and that allocated for shared memory (see next section) cannot exceed this total. Note that if *less* than the total amount of memory is allocated for partitions and shared memory, the unallocated memory is unusable by any partition.

The format of the command is:

set lp_mem_size* [*size* | *list-of-sizes*]

where:

*	Partition number (0 – 7).
size	Can be specified in hexadecimal bytes, decimal megabytes (MB) or decimal gigabytes (GB). The size must be on a 64 MB boundary. The size must also be at least as much memory required by the operating system that will be booted in soft partition *n*. For example, *OpenVMS* requires 64 MB.

Can be bound to a QBB by using the form

n = *size*

where *n* is the QBB number and *size* is as described previously.

Only one unbound (not attached to a QBB) *size* may be given for any partition. Note that if **lp_shared_mem_size** allocates memory that is bound to specific QBBs, so must the **lp_mem_size** allocations be bound to specific QBBs.

list-of-sizes A comma-separated list of sizes from particular QBBs (see ❸ in Example 4–4) for soft partition *n*. One element in the list of sizes must be at least the minimum amount of memory required by the operating system that will be booted in soft partition *n*. Each size can be specified in hexadecimal bytes, decimal megabytes (MB), or decimal gigabytes (GB). Each size must be on a 64 MB boundary..

No more than one size can be bound to an individual QBB. No more than one unbound size can be given. Note that if **lp_shared_mem_size** allocates memory that is bound to specific QBBs, so must the **lp_mem_size** allocations be bound to specific QBBs.

Example 4–5 Examples of lp_mem_size

```
P00>>> set lp_mem_size0 100000000   ❶
P00>>> set lp_mem_size1 1024 MB   ❷
P00>>> set lp_mem_size2 0 = 8 MB, 2 = 2 GB   ❸
```

❶ Soft partition 0 is assigned 100000000 hex bytes from any QBBs in the system (or hard partition).

❷ Soft partition 1 is assigned 1024 (decimal) Mbytes from any QBBs in the system (or hard partition).

❸ Soft partition 2 is assigned 8 Mbytes from QBB0 and 2 Gbytes from QBB2 in the system (or hard partition).

Example 4–6 Illegal Use of lp_mem_size

```
P00>>> set lp_mem_size_0 0=64 MB, 0=128MB  ❶
P00>>> set lp_mem_size1 1=100 MB  ❷
```

❶ Two allocations are bound to QBB0.

❷ Allocation is not on 64 MB boundary.

4.5.5 Detailed Directions for lp_shared_mem_size

The lp_shared_mem_size environment variable defines the amount of memory to be shared by the defined soft partitions.

Example 4–7 Example of Constructing lp_shared_mem_size

```
P00>>> show config ❶

                  Compaq AlphaServer GS320 6/731

QBB 0         Hard QBB 0
 Quad Switch                              QSA rev 2, QSD revs 0/0/0/0
 Duplicate Tag                 Up To 4 MB Caches      DTag revs 1/1/1/1
 CPU 0         CPU 0      4 MB Cache              EV67 pass 2.2.3
 CPU 1         CPU 1      4 MB Cache              EV67 pass 2.2.3
 CPU 2         CPU 2      4 MB Cache              EV67 pass 2.2.3
 CPU 3         CPU 3      4 MB Cache              EV67 pass 2.2.3
      .
      .
      .
Hierarchical Switch                       HSA revs 1/1, HSD revs 1/1/1/1
      .
      .
      .
 QBB      Size    Interleave                      System Memory 64 GB  ❷
  0       16 GB     32-Way  ❸
  1       16 GB     32-Way
  2       16 GB     32-Way
  3       16 GB     32-Way
```

As shown in Example 4–7, the SRM command **show config** (❶) displays the system memory size (see ❷). From this number, you assign the amount of memory to be shared. The total amount of memory for all partitions (see previous section) and that allocated for shared memory cannot exceed this total. Note that if *less* than the total amount of memory is allocated for partitions and shared memory, the unallocated memory is unusable by any partition.

The format of the command is:

set lp_shared_mem_size [*size* | *list-of-sizes*]

where:

size Can be specified in hexadecimal bytes, decimal megabytes (MB), or decimal gigabytes (GB). The size must be on a 64-MB boundary.

Can be bound to a QBB by using the form

n= size

where *n* is the QBB number and *size* is as described previously.

list-of -sizes A comma-separated list of sizes from particular QBBs (see ❸ in Example 4–7) for shared memory. Each size can be specified in hexadecimal bytes, decimal megabytes (MB), or decimal gigabytes (GB). Each size must be on a 64-MB boundary. No more than one size can be bound to an individual QBB. No more than one unbound size may be given. Also, if **lp_shared_mem_size** is allocated from specific QBBs, so must the **lp_mem_size** allocations.

Example 4–8 Examples of lp_shared_mem_size

```
P00>>> set lp_shared_mem_size 100000000    ❶
P00>>> set lp_shared_mem_size 1024 MB    ❷
P00>>> set lp_shared_mem_size 0 = 64 MB, 2 = 2 GB    ❸
```

❶ 100000000 hex bytes from any QBBs in the system is allocated for memory to be shared by all soft partitions.

❷ 1024 (decimal) Mbytes from any QBBs in the system is allocated for memory to be shared by all soft partitions.

❸ 64 Mbytes from QBB0 and 2 Gbytes from QBB2 is allocated for memory to be shared by all soft partitions.

4.5.6 Lpinit Command

An lpinit command initializes the soft partitions defined, either for the system (no hard partitions) or within a hard partition, at the system (or partition) console device.

Example 4–9 Show Config Command for a Hard Partition

```
P00>>> show config
                          Compaq Computer Corporation
                          Compaq AlphaServer GS320 6/731

 SRM Console V5.8-1, built on May 26, 2000 at 12:15:01
 PALcode          OpenVMS PALcode V1.81-1, Tru64 UNIX PALcode V1.75-1
 Micro Firmware   X5.7

 QBB 0            Hard QBB 1     ❶
  Quad Switch                                        QSA rev 2, QSD revs 0/0/0/0
  Duplicate Tag                  Up To 4 MB Caches   DTag revs 1/1/1/1
     ❷              ❸
  CPU 0            CPU 0         4 MB Cache          EV67 pass 2.4, 731 MHz
  CPU 1            CPU 1         4 MB Cache          EV67 pass 2.4, 731 MHz
  CPU 2            CPU 2         4 MB Cache          EV67 pass 2.4, 731 MHz
  CPU 3            CPU 3         4 MB Cache          EV67 pass 2.4, 731 MHz
  Memory 0                                           MPA rev 2, MPD revs 1/1
  Memory 1                                           MPA rev 2, MPD revs 1/1
  Memory 2                                           MPA rev 2, MPD revs 1/1
  Memory 3                                           MPA rev 2, MPD revs 1/1
    .
    .
    .

QBB 1            Hard QBB 2     ❶
  Quad Switch                                        QSA rev 2, QSD revs 0/0/0/0
  Duplicate Tag                  Up To 4 MB Caches   DTag revs 1/1/1/1
     ❷              ❸
  CPU 0            CPU 4         4 MB Cache          EV67 pass 2.4, 731 MHz
  CPU 1            CPU 5         4 MB Cache          EV67 pass 2.4, 731 MHz
  CPU 2            CPU 6         4 MB Cache          EV67 pass 2.4, 731 MHz
  CPU 3            CPU 7         4 MB Cache          EV67 pass 2.4, 731 MHz
  Memory 0                                           MPA rev 2, MPD revs 1/1
  Memory 1                                           MPA rev 2, MPD revs 1/1
  Memory 2                                           MPA rev 2, MPD revs 1/1
  Memory 3                                           MPA rev 2, MPD revs 1/1
  .
  .
  .
 QBB     Size    Interleave                  System Memory 8 GB
  0       4 GB      8-Way
  1       4 GB      8-Way
```

Soft partitions within hard partitions are discussed in the next section. For convenience, simply assume for now that the **show config** command in Example 4–9 was issued from the console device for the second hard partition of a 3-hard-partition system.

Note the QBB numbering. A "soft QBB number" is given on the left, and the hard QBB number is given on the right (see ❶).

Note the processor numbering: two columns are given. The first column gives the processor numbers shown on the QBB (see ❷). The second column gives what are called the "soft CPU numbers," running from 0 through 7 in this example, so that there is a unique CPU number for each CPU in the partition (see ❸).

Example 4–10 shows how two soft partitions might be set up within this hard partition. Note that soft CPU numbers 0, 1, 2, and 3 are allocated to soft partition 0, and soft CPU numbers 4, 5, 6, and 7 are allocated to soft partition 1 (see ❶). Likewise, soft QBB numbers are used in setting the **lp_io_mask** environment variables.

Shared memory is allocated from both of the two QBBs.

The **lpinit** command initializes the soft partitions defined, and the soft partitions are started up with displays at their respective console devices.

Example 4–10 Setting Up Soft Partitions

```
P00>>> set lp_count 2
P00>>> set lp_cpu_mask0 f    ❶
P00>>> set lp_cpu_mask1 f0
P00>>> set lp_error_target 0
P00>>> set lp_io_mask0 1      ❷
P00>>> set lp_io_mask1 2
P00>>> set lp_mem_size0  0 = 3GB
P00>>> set lp_mem_size1  1 = 3GB
P00>>> set lp_shared_mem_size 0 = 1GB, 1 = 1GB
P00>>> lpinit
```

4.6 Soft Partitions Within Hard Partitions

It is possible to define soft partitions within hard partitions in a system.

Table 4–3 Example of Soft Partitions Within Hard Partitions

	Hard Part. 0	Hard Part. 1	Hard Partition 2		
			Soft Part. 0	Soft Part. 1	Soft Part. 2
Hard QBB Number	0, 1	2, 3	4	5, 6	6, 7
Soft QBB Number	0, 1	0, 1	0	1, 2	2, 3
Systemwide CPU Number	0–7	8–15	16–19	20, 21, 24, 25	22, 23, 26–31
Soft CPU Number	0–7	0–7	0–3	4, 5, 8, 9	6, 7, 10–15
Number of PCI Boxes	4; 2 with SIO	4; 1 with SIO	2; 1 with SIO (from soft QBB0)	2 with SIO (from sofy QBB1)	4; 2 with SIO (from soft QBB2 and 3)
Memory	From QBB0 and 1	From QBB2 and 3	1 GB	2 GB	2 GB
			Shared memory: 5 GB		

Table 4–3 shows how a system might be divided into hard partitions and soft partitions. The table shows three hard partitions. The third hard partition is then divided into three soft partitions. An example of the commands that would define such partitioning is given below.

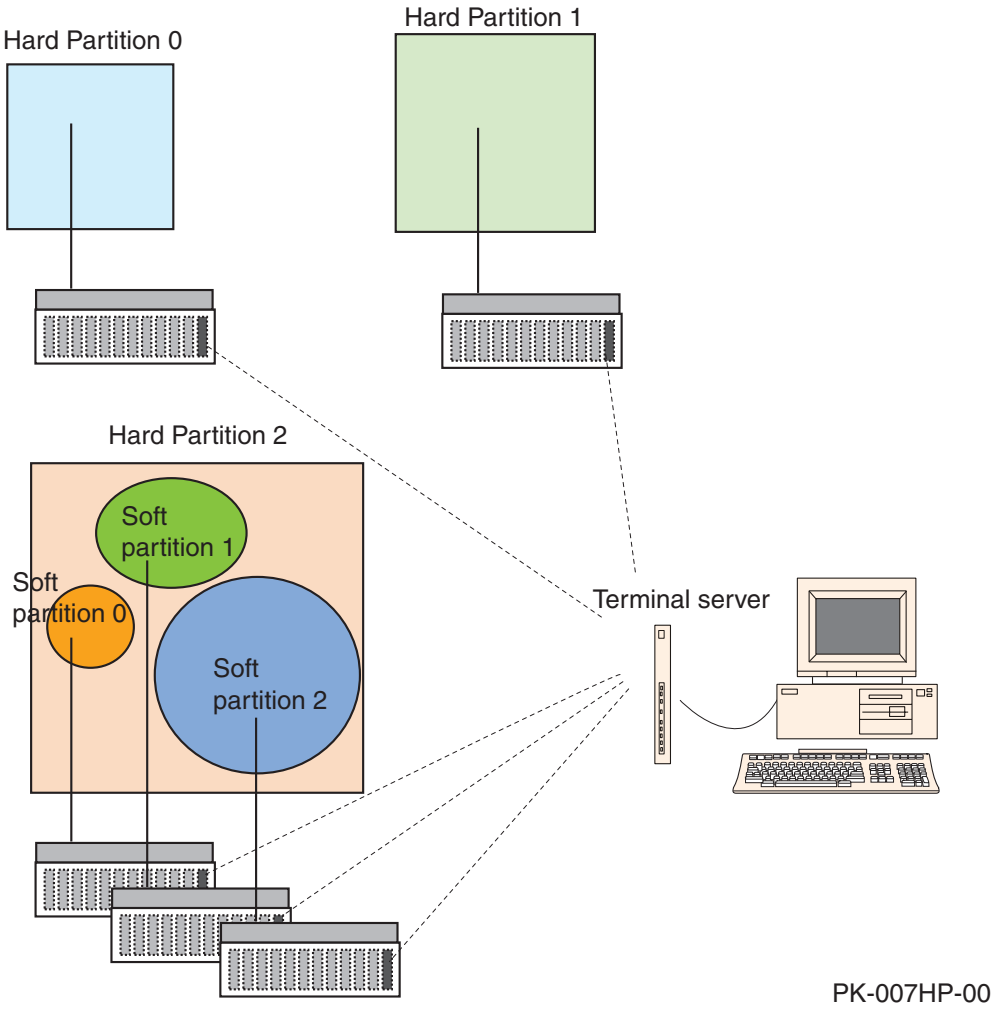
Example 4–11 Soft Partitions Within Hard Partitions

```
SCM_E0> set hp_count 3
SCM_E0> set hp_qbb_mask0 3
SCM_E0> set hp_qbb_mask1 c
SCM_E0> set hp_qbb_mask2 f0
SCM_E0> power on -par 0
SCM_E0> power on -par 1
SCM_E0> power on -par 2
SCM_E0> quit
    .
    .
    .
  [go to the console device or terminal emulator screen
  that controls hard partition 2]


P00>>> set lp_count 3
P00>>> set lp_cpu_mask0 f
P00>>> set lp_cpu_mask1 330
P00>>> set lp_cpu_mask2 fcc0
P00>>> set lp_error_target 0
P00>>> set lp_io_mask0 1
P00>>> set lp_io_mask1 4
P00>>> set lp_io_mask2 c
P00>>> set lp_mem_size0 1GB
P00>>> set lp_mem_size1 2GB
P00>>> set lp_mem_size2 2GB
P00>>> set lp_shared_mem_size 0 = 5GB, 1 = 5GB,2 = 5GB
P00>>> lpinit
```

4.7 Master and Slave SCMs

The multiple standard I/O modules needed for hard or soft partitions introduce the need for "master" and "secondary" SCMs. The master SCM, controls the console serial bus (CSB). Other SCMs are secondary.

Figure 4–10 Multiple SCMs in a Partitioned System



A user needs access to an individual partition for maintenance and debugging. This access is provided through the requirement that each partition, hard or soft, must be attached to a master PCI box with a standard I/O module.

The standard I/O module contains both SCM and SRM firmware. Only one SCM can control the console serial bus (CSB) at any given time. This SCM is called the *master SCM*. All other SCMs are called *secondary*, or *slave*, SCMs.

A master SCM must be:

- Attached to the operator control panel (OCP).
- Have its **scm_csb_master_eligible** environment variable set to 1. (By default, each SCM's **scm_csb_master_eligible** environment variable is set to 1. You can override a particular SCM's eligibility by setting this environment variable to 0 set from the console connected to that SCM.)

Figure 4–10 illustrates a system divided into three hard partitions. One hard partition is divided into three soft partitions. Each of these partitions has a master PCI box with a standard I/O module containing SCM and SRM firmware. Note that hard partition 2 is connected to three master PCI boxes. The SRM that controls the definition of soft partitions within hard partition 2 is the SRM residing on the standard I/O module in the master PCI box with the lowest-numbered hose connection within the hard partition.

At VAUX power-up on a system with multiple SCMs, the master SCM is selected as the one with the lowest-numbered CSB address (determined by the node ID switch setting on the PCI box) that is also connected to the OCP and has the **scm_csb_master_eligible** environment variable set to 1. The standby SCM is selected as the one with the next-lowest CSB adress that is also connected to the OCP and has the **scm_csb_master_eligible** environment variable set to 1.

4.8 SCM Functionality from Master and Slave SCMs

Systemwide functionality is provided by the master SCM; partition-dependent functionality is provided at the local (master or slave) level.

Figure 4–11 Systemwide Functionality

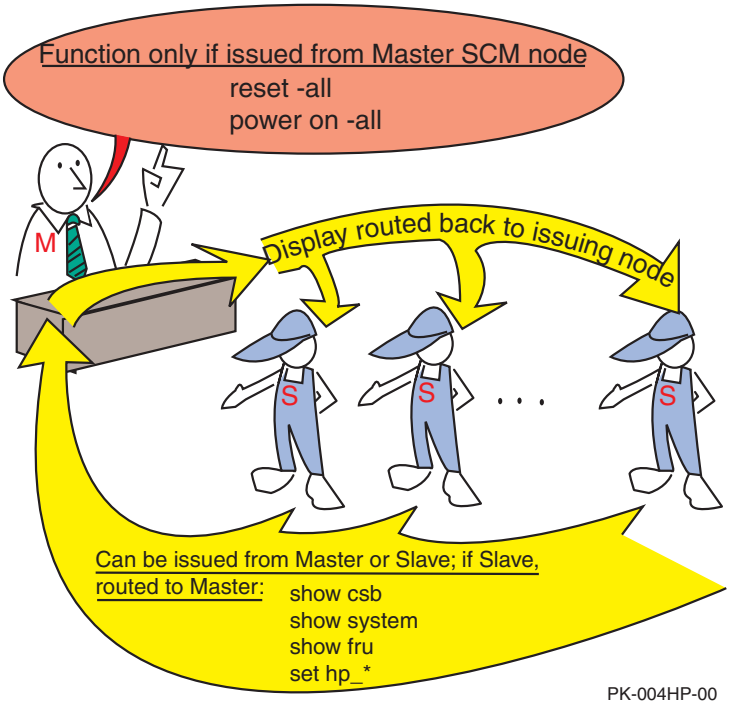


Figure 4–11 shows which SCM commands affect the whole system (systemwide functionality). Commands that affect all nodes on the console serial bus can be executed only from the master SCM. For example, commands that display information for all CSB nodes are routed through the master SCM for execution.

Figure 4–12 shows those SCM commands providing partition-specific functionality. As shown, you can set up and use remote access for each hard or soft partition (as long as they have a PCI box with a standard I/O module attached). The **halt**, **reset**, and **power** commands affect only the hard partition in which they are executed; however, they do have **–partition** options that can affect other partitions if executed from the master SCM.

Figure 4–12 Partition-Dependent Functionality

Take action from local SCM (Master or Slave):

build EEPROM	el	help	show status
clear alert	erase	init	show nvr
deposit	examine	quit	test alert
disable	hangup	set*	

(*Set hp_* routed to master SCM)

Routed to Master SCM; affects only the hard partition from which it is issued (hp_count≠0):

halt [in, out]
power [off, on]
reset
fault

PK-005HP-00

Chapter 5
Booting an Operating System

This chapter briefly describes how to boot the *OpenVMS* and *Tru64 UNIX* operating systems on *AlphaServer* GS80/160/320 systems. More complete information on booting and installing an operating system is given in the *AlphaServer GS80/160/320 User's Guide.*

Sections include:

- Preparation
- Selecting a Boot Device
- Booting an Operating System

5.1 Preparation

There are some steps you must take preparatory to booting and other steps that can make booting thereafter easier.

5.1.1 Set os_type Environment Variable

For factory-installed software (FIS), a default operating system is defined. If you do not have factory-installed software, or you wish to change the default operating system, you use the set os_type command.

Example 5–1 Setting os_type for OpenVMS

```
P00>>> set os_type vms
```

Example 5–2 Setting os_type for Tru64 UNIX

```
P00>>> set os_type unix
```

***NOTE**: If your system is partitioned, you must set these and any other relevant environment variables for each partition.*

5.1.2 Set Console Environment Variable

For *OpenVMS* systems, the console environment variable should be set to serial (the default). For *Tru64 UNIX* systems, the console environment variable may be either serial or graphics (if a graphics device is attached).

Example 5–3 Set Console to Serial

```
P00>>> set console serial
```

Example 5–4 Set Console to Graphics

```
P00>>> set console graphics
```

All operating systems require a serial console terminal. Some operating systems support a graphics monitor as well. The **set console graphics** command is needed for systems having a graphics monitor as part of their console interface. The default setting for the **console** environment variable is **serial.**

By specifying **set console graphics**, the SRM console will build the proper parameters in a data structure called the HWRPB, or hardware restart parameter block, so that an operating system will build with graphics enabled.

5.1.3 Set auto_action Environment Variable

You can direct the SRM console to automatically boot the operating system upon system power-up, restart, or init. The default is to halt the system after system self-test and leave the console device in SRM console mode.

Example 5–5 Setting the auto_action Environment Variable

```
P00>>> show auto_action
auto_action   halt
P00>>> set auto_action boot
```

The **auto_action** environment variable is set to **halt** by default. In this case, at system restart (power-up, reset, or SRM **init** command), the SRM console software will display the system self-test and display the SRM console prompt to accept commands.

If you set the **auto_action** environment variable to **boot**, at system restart, the SRM console software will display the system self-test and then automatically boot the operating system according to the default boot parameters you defined.

5.2 Selecting a Boot Device

An operating system can be booted from a number of devices: the DVD/CD-ROM drive, a local system disk, a disk connected to the system through a CIPCA adapter, or by Ethernet from a remote disk on another system. Certain operating systems use a subset of these devices. Note that factory-installed software has a default boot device selected.

Table 5–1 Boot Devices

Device	Location
DVD/CD-ROM	In-cabinet compact disk drive, used for booting the Loadable Firmware Update (LFU) Utility. See Appendix B.
Local device	Disk connected to the system through an adapter on an I/O bus.
CI disk	Disk located on the system's HSC controller connected to the system by a CIPCA adapter on the PCI bus.
Remote disk	Disk connected to another system on the Ethernet, through the Ethernet port interface or the adapter.

Table 5–1 lists the types of devices available for your operating system for booting. For systems with factory-installed software (FIS), a default boot device has already been selected for you. If you want to change this default, follow the instructions here.

NOTE: *The default boot device selected applies to the operating system. If you wish to boot the Loadable Firmware Update (LFU) utility, you must specify the boot device where LFU is located on your system, since LFU is not on the FIS boot device.*

You can set or change the default boot device with the **set bootdef_dev** command, as described in Sections 5.2.1 and 5.2.2. You can also specify a particular device with the SRM **boot** command.

5.2.1 Show Device Command

If you do not wish to use the default boot device selected for factory-installed *OpenVMS* or *Tru64 UNIX* systems, the show device SRM console command displays your system I/O configuration, allowing you to identify and select an appropriate boot device.

Example 5–6 Show Device Command

```
P00>>> sho dev
dka0.0.0.1.0               DKA0                       RZ1DF-BF  1614
dkb0.0.0.7.1               DKB0                COMPAQ BB00911CA0  3B05
dkb100.1.0.7.1             DKB100              COMPAQ BB00911CA0  3B05
dkb200.2.0.7.1             DKB200              COMPAQ BB00911CA0  3B05
dkb300.3.0.7.1             DKB300              COMPAQ BB00911CA0  3B05
dkc0.0.0.1.2               DKC0                COMPAQ BB00911CA0  3B05
    .
    .
    .
dkn0.0.0.1.8               DKN0                       RZ1DF-BF  1614
dqa0.0.0.15.0              DQA0          TOSHIBA CD-ROM XM-6302B  1017  ❶
dqc0.0.0.15.6              DQC0          TOSHIBA CD-ROM XM-6302B  1017
dqe0.0.0.15.8              DQE0          TOSHIBA CD-ROM XM-6302B  1017
ewa0.0.0.3.8               EWA0               08-00-2B-C3-BD-00
fwa0.0.0.4.1               FWA0               00-00-F8-CD-1F-39
fwb0.0.0.5.7               FWB0               00-00-F8-CD-22-E0
fwc0.0.0.1.10              FWC0               00-00-F8-CD-22-FB
pga0.0.0.7.7               PGA0          WWN 1000-0000-c920-da1e
pka0.7.0.1.0               PKA0                    SCSI Bus ID 7  5.57
pkb0.7.0.7.1               PKB0                    SCSI Bus ID 7  5.57
pkc0.7.0.1.2               PKC0                    SCSI Bus ID 7  5.57
pkd0.7.0.2.2               PKD0                    SCSI Bus ID 7  5.57
pke0.7.0.3.2               PKE0                    SCSI Bus ID 7  5.57
pkf0.7.0.4.3               PKF0                    SCSI Bus ID 7  5.57
pkg0.7.0.5.3               PKG0                    SCSI Bus ID 7  5.57
pkh0.7.0.6.3               PKH0                    SCSI Bus ID 7  5.57
pki0.7.0.7.3               PKI0                    SCSI Bus ID 7  5.57
pkj0.7.0.2.4               PKJ0                    SCSI Bus ID 7  5.57
pkk0.7.0.6.5               PKK0                    SCSI Bus ID 7  5.57
pkl0.7.0.1.6               PKL0                    SCSI Bus ID 7  5.57
pkm0.7.0.2.6               PKM0                    SCSI Bus ID 7  5.57
pkn0.7.0.1.8               PKN0                    SCSI Bus ID 7  5.57
pko0.7.0.2.8               PKO0                    SCSI Bus ID 7  5.57
pkp0.7.0.1.16              PKP0                    SCSI Bus ID 7  5.57
P00>>>
```

❶ The example shows a CD-ROM disk drive, where you can load firmware update CD-ROM disks. You can use the **boot** command to boot LFU from

the disk. In this example, the command would be:

```
P00>>> boot dqa0
```

If you are not planning to use the default boot device selected for your factory-installed software, or if your operating system is not factory installed, select an appropriate boot device for your system. Use the **show device** command to display all the devices on your system. The format for device names is:

ddau.n.c.s.h

where the fields are as defined in Table 5–2.

Table 5–2 SRM Console Device Naming Conventions

Field	Meaning	Description
dd	Device driver	Two-letter designator for a port or class device driver. Usually one of: dk SCSI drive or CD dq IDE CD-ROM dr RAID set device ei Ethernet port fw FDDI device mk SCSI tape pk SCSI port
a	Adapter ID	Specifies the one-letter designator for adapter ID. One of a, b, c,…
u	Unit number	Specifies the device unit number. For MSCP devices, this is a unique increasing number. For SCSI devices, this is 100 times the bus node no.
n	Node number	Specifies the bus node ID.
c	Channel number	Specifies the channel or PCI function number. Only meaningful for multi-channel or multi-function devices.
s	Slot number	Specifies the device's logical slot number. The output of a **show config** command indicates the logical slot numbers for devices in a system.
h	Hose number	Specifies the device's hose number.

Ideally, your system manager labeled the I/O devices on your system, so that you can identify a particular unit by its name. If not, you will have to follow

cables back to the device adapter and then back to the hose to identify a particular unit.

In either case, select a device on your system that is of the type appropriate for the media your operating system software is on. It is useful to reserve one device on your system for this use, for consistency. This is done by setting the **bootdef_dev** environment variable, as described next.

5.2.2 Boot Environment Variables

You can change the factory-installed default boot device, or, if your operating system is not factory-installed, define a default boot device, to be used automatically if no device is specified in the boot command itself.

Example 5–7 Viewing and Setting Boot Environment Variables

```
P00>>> show boot*    ❶
boot_dev
boot_file
boot_osflags            0
boot_reset              OFF
bootdef_dev
booted_dev
booted_file
booted_osflags
P00>>> set bootdef_dev dkb100    ❷
P00>>> show boot*
boot_dev
boot_file
boot_osflags            0
boot_reset              OFF
bootdef_dev             dkb100.1.0.6.1
booted_dev
booted_file
booted_osflags
```

Some boot environment variables can be set by the user using the **set** SRM console command. Others are set by values you may specify in a **boot** command. See the *AlphaServer GS80/160/320 User's Guide* for a discussion of boot environment variables.

❶ The **show boot*** command displays the current settings of the boot environment variables.

❷ In Example 5–7, the user sets the default boot device environment variable **bootdef_dev** to device dkb100.

5.3 Booting an Operating System

Use the show boot* command to ensure that the device from which to boot the operating system, and other boot parameters, are correct. Issue the boot command.

Example 5–8 Tru64 UNIX Boot

```
P00>>> show boot*    ❶
boot_dev                dka0.0.0.1.0
boot_file
boot_osflags            a
boot_reset              OFF
bootdef_dev             dka0.0.0.1.0
booted_dev
booted_file
booted_osflags
P00>>> boot    ❷
 (boot dka0.0.0.1.0 -flags a)
block 0 of dka0.0.0.1.0 is a valid boot block
reading 13 blocks from dka0.0.0.1.0
bootstrap code read in
base = 2e6000, image_start = 0, image_bytes = 1a00
initializing HWRPB at 2000
initializing page table at 17fbda000
initializing machine state
setting affinity to the primary CPU
jumping to bootstrap code

Tru64 UNIX boot - Wed Sep 8 20:59:58 EDT 1999

Loading vmunix ...
Loading at 0xfffffc0000430000

Sizes:
text = 4800560
data = 903872
bss  = 1412480
Starting at 0xfffffc00007790d0

Loading vmunix symbol table ... [1330800 bytes]
Alpha boot: available memory from 0x93de000 to 0x3180000000
Digital Tru64 UNIX V4.0G-1 (Rev. 1398); Thu Oct 21 13:34:00 EDT
1999
physical memory = 24572.00 megabytes.
available memory = 24140.65 megabytes.
```

Example 5–8 Tru64 UNIX Boot (Continued)

```
using 94344 buffers containing 737.06 megabytes of memory
Master cpu at slot 0.
Firmware revision: 5.6-6930
PALcode: Digital Tru64 UNIX version 1.60-1
Compaq AlphaServer GS320 6/731
.
.
.
Digital Tru64 UNIX Version V4.0

login:    ❸
```

Example 5–8 shows an example of booting a *Tru64 UNIX* system.

❶ The **show boot** command shows the values of all the environment variables beginning with the characters "boot."

❷ The user issues the **boot** command.

❸ The operating system boots and displays its login prompt.

Chapter 6
SCM Command Reference

The format of the commands recognized by the SCM command parser is described in full in this chapter. The sections are:

- Language Overview
- SCM Command Summary
- SCM Command Description Conventions
- Clear Command
- Deposit Command
- Disable Command
- El Command
- Enable Command
- Erase Command
- Examine Command
- Fault Command
- Halt Command
- Hangup Command
- Help Command
- Init Command
- Power Command
- Quit Command
- Reset Command
- Set Commands
- Show Commands
- Test Alert Command
- Test &pc*n* Command

When the operator enters SCM mode for the system or partition, the operating system state is unchanged. The operator communicates with the firmware through the console terminal, which displays one of the following prompts:

`SCM_`*`nn`*`>` or

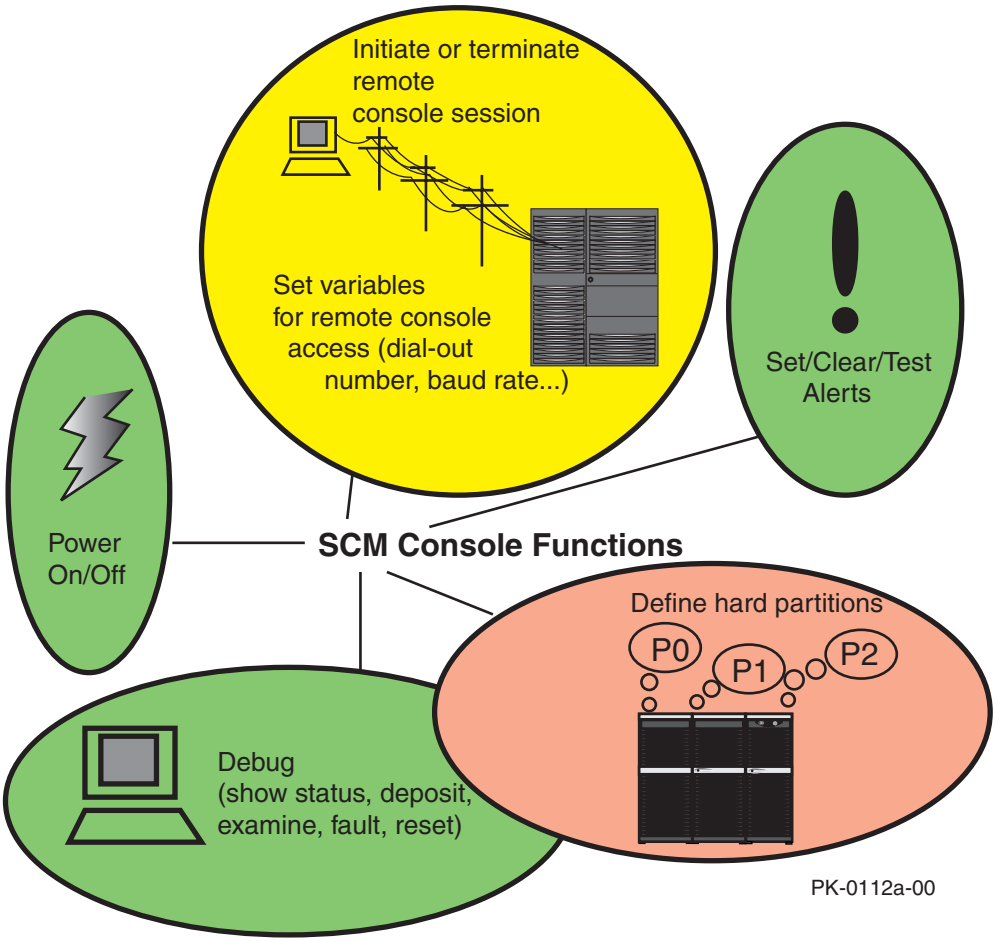
`SLV_`*`nn`*`>`

Where SCM is given for a console device attached to the master CSB. SLV indicates a console device attached to a slave CSB node, and *nn* may range from E0 to EF (the number shown by a **show csb** display for the CSB node).

6.1 Language Overview

This section lists system management tasks, the related SCM commands, and the place in this manual where a description of the commands can be found.

Figure 6–1 Overview of SCM Command Language Functionality



Remote Access

As described in Section 3.1, the SCM allows remote maintenance of *AlphaServer* GS80/160/320 systems.

System monitoring and troubleshooting can be done by a system manager at a remote location; either in the next room, building, state, or even country. Once a specific problem has been identified, the system manager can dispatch a local service technician to handle the problem.

A subset of SCM commands must be issued from the local console to define parameters needed for remote access. Another subset of commands can be used from either the local or remote console to display system status and troubleshoot the system as a whole.

Hard Partitions

As described in Chapter 4, you can use SCM commands to divide the system into *hard partitions*. This capability allows separate copies, or *instances*, of an operating system to run in each partition.

Entering and Leaving the SCM Console

To enter the SCM command-line interpreter, the user types the escape sequence on the local or remote serial console terminal. (The default escape sequence is **<Esc><Esc>scm**.) The operating system state does not change when the user enters SCM command mode.

The **quit** command terminates SCM console usage and returns the system to the state it was in when the escape sequence was entered.

6.2 SCM Command Summary

Table 6–1 lists the commands used for various tasks, and the sections or chapters where examples of the task can be found. (Detailed information on SCM command format is given in Section 6.3.)

Table 6–1 Management Tasks and Related SCM Commands

Task	Commands	Section
Get help on SCM commands	**help** or **?**	Section 3.3
Erase the current screen	**erase**	Section 6.9
Set up for remote console management features	**set init** **set dial** **set alert** **set password** **enable remote** **enable alert** **test alert** **set com1_mode**	Section 3.4 Section 3.6
Initialize the modem to prepare it to send alerts and receive incoming calls; clear alerts.	**init** **set baud** **clear alert**	Section 3.4 Section 6.19.2 Section 6.4
Examine the state of the system	**show** **el**	Section 3.7 Section 6.7
Low-order debugging	**deposit** **examine**	Section 6.5 and 6.10
Define hard partitions	**set hp_count** **set hp_qbb_mask***	Section 4.4
Control power to the system, initialize hard partitions	**power**	Sections 3.8, 4.4
Reset system or QBB	**reset**	Section 6.18
Terminate the remote session	**hangup**	Section 3.10
Leave the SCM command interpreter	**quit**	Section 3.9

6.3 SCM Command Description Conventions

The following conventions are used in the descriptions of the SCM command-line interface.

Table 6–2 SCM Command Description Conventions

Convention	Meaning
`fixed-font`	SCM command examples are shown in a fixed-width font.
bold	Command and option keywords are presented in bold type.
item	Italics indicate a placeholder for an item that the user supplies.
[*item*]	Square brackets are used to enclose optional parameters, qualifiers, and values. For example, **`help`** `[topic]`.
{*a*, *b*, *c*}	Braces containing items separated by commas imply mutually exclusive values. For example {*a*, *b*, *c*} indicates that you can choose one of *a*, *b*, or *c*.
{*a* | *b* | *c*}	Braces containing items separated by the vertical bar indicate that you can choose any combination of *a*, *b*, and *c*.

6.4 Clear Command

The clear command clears the current alert condition.

Syntax

clear {alert, port}

Arguments

alert Clears the current alert condition and causes the SCM to stop paging the remote operator. If the alert is not cleared, the SCM will continue to page the remote operator every 30 minutes. The first alert to occur is logged as the alert type. Subsequent alerts will not be seen by the SCM until the current alert condition is cleared. Only one alert can be pending at any given time.

port Clears communication of terminal emulator when the port is "wedged" in COM1 mode. Escape back to the SCM and clear the port to COM1.

Example

```
SCM_E0> clear alert
```

6.5 Deposit Command

The deposit command stores data in the contents of memory or in IPRs and GPRs in specific processors, including those processors on the CSB.

Syntax

deposit [- {**b**, **w**, **l**, **q**, **o**}] [-{**gpr** *n*, **ipr** *n*, **p**}] [**-n** *count*] [**&**{**p**, **m**}*n*]
[*, *address*] [*data*]

Options

-b The *data* deposited is a byte (8 bits).

-w The *data* deposited is a word (16 bits).

-l The *data* deposited is a longword (32 bits).

-q The *data* deposited is a quadword (64 bits). This is the default.

-o The *data* deposited is an octaword (128 bits).

-gpr *n* The *data* is deposited in general-processor register (GPR) *n* (hex).

-ipr *n* The *data* is deposited in internal-processor register (IPR) *n* (hex).

-p The *data* is deposited in a physical memory address (default).

-n *count* The address will be incremented *count* (hex) times. The address increment is determined by the address size option (**-b, -w, -l, -q, -o**).

&p*n* Specifies the CPU on which the **deposit** is to be made. The argument *n* is the soft CPU number displayed in the **show csb** command.

&m*n* Specifies the number of the CSB node (AM186 processor) on which the **deposit** is to be performed. The argument *n* is the CSB address shown in the **show csb** command.

Arguments

* The set of arguments (*address*, *data*) used in the latest **deposit** or **examine** command are to be used here.

address A physical address in system memory.

data The data (hex) to be written to the specified address or register.

Example

```
SCM_EF> ex &mef 80000     ❶
CSB_EF/00080000: 00
SCM_EF> dep &mef 80000 1b   ❷
SCM_EF> ex &mef 80000     ❸
CSB_EF/00080000: 1B
```

❶ Examines location 80000 in CSB node EF. The current value is 00.

❷ Deposits the hexadecimal value 1b into location 80000 of CSB node EF.

❸ Examines location 80000 in CSB node EF. It has been changed to 1B.

6.6 Disable Command

The disable command disables system alerts and remote console operations.

Syntax

disable {**alert**, **remote**}

Arguments

alert Stops the SCM from monitoring alert conditions. **Using this command is strongly discouraged.** Alert monitoring is essential to ensuring that environmental conditions do not damage the hardware.

remote Disables the capability for a remote user to dial in to the system and execute commands.

Examples

```
SCM_E0> disable alert
SCM_E0> disable remote
```

6.7 El Command

The el command displays the current event log on the console screen. The event log contains the most immediate (up to 4 Kbytes) input and output that has passed through the SCM. The el command is useful for reviewing recent command history.

Syntax

el

Example

```
SCM_E0> el    ❶
fff

PSM33 SysEvent: QBB_POWER_OFF    ❷
SysEvent Reg0: 1cbc
SysEvent Reg1: cfff

SCM_E0>
SCM_E0> p on    ❸
Powering on PCI Box 0
Powering on PCI Box 1
   .
   .
   .
Qbb0 unloading console across hose2 from PCI Box-0
Console COM1 from master PCI Box-0
Retrieve FRU information for Shared RAM
SCM_E0> .
QBB3 now Testing Step-e.
QBB0 now Testing Step-e
QBB1 now Testing Step-e
QBB2 now Testing Step-e...
Power On Complete
SCM_E0> el    ❹
```

❶ The user types the **el** command.

❷ The end of a display produced by a **power off** command is the most recent event in the event log history.

❸ A **power on** command was issued; the display contains the entire output of the power-up, abbreviated here.

❹ The most recent command, the **el** command itself, is displayed.

6.8 Enable Command

The enable command enables system alerts or remote access.

Syntax

enable {**alert**, **remote**}

Arguments

alert Enables the SCM's monitoring of all alert conditions:

- Blower failure from a PSM
- Vital power failure from a PSM
- Overtemperature failure from a PSM
- Overtemperature failure from an HPM
- Vital power failure from an HPM
- System clock failure from an HPM
- Fan failure from a PBM
- Overtemperature failure from a PBM
- Power supply failure from a PBM
- User-initiated alert

When an alert occurs, the dial-out sequence is sent. The alert is repeated every 30 minutes until a **clear alert** or **disable alert**, or until the OCP keyswitch is turned to Secure.

remote Enables remote dial-in. (See Section 3.4 for a discussion of values that must be set for the **init** of the modem to succeed.)

Examples

```
SCM_E0> enable alert  ❶
SCM_E0> enable remote  ❷
```

❶ Turns on alert condition monitoring in the SCM.

❷ Initializes the modem (allows remote access).

6.9 Erase Command

The erase command erases the current display screen on the console terminal.

Syntax

erase

Example

```
SCM_E0> erase
```

6.10 Examine Command

The examine command displays the contents of memory or IPRs and GPRs in specific processors, including those processors on the CSB.

Syntax

examine [- {**b**, **w**, **l**, **q**}] [-{**gpr** *n*, **ipr** *n*, **p**}] [**-n** *count*] [**&**{**p**, **m**}*n*] [*, *address*]

Options

-b The data to be displayed is a byte (8 bits).

-w The data to be displayed is a word (16 bits).

-l The data to be displayed is a longword (32 bits).

-q The data to be displayed is a quadword (64 bits). This is the default.

-gpr *n* The data to be displayed is in general-processor register (GPR) *n* (hex).

-ipr *n* The data to be displayed is in internal-processor register (IPR) *n* (hex).

-n *count* The *address* will be incremented *count* (hex) times. The address increment is determined by the address size option (**-b, -w, -l, -q, -o**).

&p*n* Specifies the CPU module on which the **examine** is to be performed. The argument *n* is the CPU number shown under the heading CSB in the **show csb** command.

&m*n* Specifies the number of the CSB node (AM186 processor) on which the **examine** is to be performed. The argument *n* is the CSB address shown in the first column in the **show csb** command.

Arguments

*	Indicates that the set of arguments (*address*, *data*) used in the last **deposit** or **examine** command are to be used here.
address	A physical address in system memory.

Example

```
SCM_EF> ex &mef 80000      ❶
CSB_EF/00080000: 00
SCM_EF> dep &mef 80000 1b    ❷
SCM_EF> ex &mef 80000      ❸
CSB_EF/00080000: 1B
```

❶ Examines location 80000 in CSB node EF. The current value is 00.

❷ Deposits the hexadecimal value 1b into location 80000 of CSB node EF.

❸ Examines location 80000 in CSB node EF. It has been changed to the value 1b.

6.11 Fault Command

The fault command emulates the control panel Fault button. It is similar to the reset operation, except that the contents of error registers in the system are "latched." That is, they retain their values.

Syntax

fault [**-all**, **-par**[**tition**] *n*]

Options

-all	Asserts the fault condition for all hard partitions (the default).
-par[tition] *n*	Asserts the fault condition for hard partition *n*.

Example

```
SCM_E0> fault
Returning to system COM1 port
P00>>>
```

6.12 Halt Command

The halt command emulates the control panel Halt button.

Syntax

halt { **in**, **out**} [**-par**[**tition** *n*], **-all**]

Arguments

in Asserts system halt. Causes the operating system to perform a halt, and prevents the console from booting the operating system. This halt stays in effect until an SRM **continue** command is given.

out Deasserts system halt.

Options (only effective from Master SCM)

-par[**tition**] *n* Asserts or deasserts halt on the hard partition identified by *n*.

-all Asserts or deasserts halt on all hard partitions (the default).

Example

```
SCM_E0> halt in
Returning to system COM1 port
P00>>>
```

6.13 Hangup Command

The hangup command terminates a remote access session, disconnecting the modem. The command can be issued from a remote terminal or from the local console.

Syntax

hangup

Example

```
SCM_E0> hangup
```

6.14 Help Command

The help (or ?) command displays information about SCM command format and function.

Syntax

help

Example

```
SCM_E0> help


               SCM CLI Version 1.0  2000/27/07

 build <fru> <pn> <sn> <mod> <ali> Build FRU Data(pn=2-5-2.4 sn=xxyzzabcde)
 build EEPROM                      Restore EEPROM RMC default values
 clear {alert, port}               Clear alert state, clear COM1 port comm
 deposit [-ipr,-spr(-iic,-offset),&mn, &p<csb_adr>,-next<n>,-q,-l,-w,-b] [data]
                                   Write data to a memory location or CSR/IPR
 disable {alert, remote, test<n>}  Disable remote dial-in, alert dial-out,test-n
 el                                Type out event log
 enable {alert, remote, test <n>}  Enable remote dial-in, alert dial-out, test-n
 erase                             Erase screen
 examine [-ipr, -spr, (-iic,-offset),&mn, &p<csb_adr>, -next <n>, -q, -l, -w, -b]
                                   Read a memory location or CSR/IPR
 fault [-all, -partition <n>]      Create a system wide FAULT condition
 halt {in, out} <-partition <n>,-all>Halt the system(virtual OCP halt button)
 hangup                            Terminate remote session
 help or ?                         Display list of SCM commands
 init                              Initialize the modem
 master <cli cmd>                  Slave SCM Command to master SCM
 power {on, off} [-all, -partition <n>]
                                   Power command
 quit                              Switch from SCM-CLI mode COM1 port
 reset [-all -partition <n>]       System level reset
 set flow {com1,local,modem} {hard,soft,both,none}
                                   Set a port's flow control characteristic
 set baud {com1,local,modem} <value> Set a port's baud rate
 set com1_mode {through,snoop,software_bypass,firm_bypass,local}
                                   Set COM1 operating mode
 show{csb,system,status,fru,nvr}   Show various system information
 test {alert, &pc<n> test_num}     Trigger an alert or run a CPU test
 update {-csb <node_address,…>     Update micro FLASH on CSB node(s)

SCM_E0>
```

6.15 Init Command

The init command initializes the modem; used in conjunction with other commands to enable remote access (see Chapter 3).

Syntax

init

Example

```
SCM_E0> init
Querying the modem port. . . non-modem detected  ❶
Modem port initialized

SCM_E0> init
Querying the modem port. . . no device detected   ❷

SCM_E0> init
Querying the modem port. . . modem detected      ❸
Modem port initialized
```

❶ In this case, a terminal is connected to the modem port.

❷ Here, nothing is connected to the modem port.

❸ In the final example, a modem is connected to the modem port.

6.16 Power Command

The power on command directs the SCM to supply power to the system or the specified components. Likewise, the power off command directs the SCM to remove power from the system or specified components.

Syntax

power [**on, off**] [*none,* **-all**, **-par[tition]** *n*]

Arguments

on Supplies power to the entire system or the specified component(s).

off Removes power from the entire system or the specified component(s).

Options

none
On partitioned systems: Powers on or off only the QBBs that comprise the hard partition from which the command is issued.
On nonpartitioned systems: Powers on or off the entire system.

-all
On partitioned systems: Can be issued from the master SCM to power on or off all hard partitions, including PCI boxes.
On nonpartitioned systems: Functions the same as supplying no arguments (powers on or off the entire system).

-par *n*
Can be issued from the master SCM; supplies power to and removes power from the hard partition specified by *n*. See Chapter 4.

Description

Power is supplied to the nodes on the console serial bus (CSB) whenever the system is plugged in and the main breaker switch is on. The **power on** command causes the SCM to supply power to the rest of the system or to the specified partition. The **power on** display is lengthy, detailing the process of powering on the system in each of its five phases, and can fill many screens. (The **el** command is useful for examining the steps of a power-up after the fact.) As a synopsis, the **power on** display terminates with the same display produced by the **show system** command.

The **power off** command removes power from the entire system or from a particular partition in an orderly fashion.

WARNING: *The keyswitch on the control panel must be set to Secure before any removal or replacement operation. Otherwise, a remote operator could dial in and inadvertently issue a* ***power on*** *command while the local service technician is working with a component. Injury to the service technician and/or damage to the component could occur.*

Example

```
SCM_E0> p on
Powering on PCI Box 0
Powering on PCI Box 1
Powering on PCI Box 2
Powering on PCI Box 3
Powering on PCI Box 5
Powering on PCI Box 6
Powering ON on QBB-0
Power ON Phase INIT
Powering ON on QBB-1
Powering ON on QBB-2
Powering ON on QBB-3
SCM powered via PBM
SCM_E0>
QBB0 now Testing Step-0
QBB1 now Testing Step-0
     .
     .
     .
```

6.17 Quit Command

The quit command changes the user's input focus from the SCM command language interface to the system COM1 port.

Syntax

quit

Example

```
SCM_E0> quit

Returning to system COM1 port
```

6.18 Reset Command

The reset command resets the entire system or a specified component.

Syntax

reset [*none*, {**-partition** [*n*], **-qbb** [*n*], **-cpu** [*n*] , **–pci** [*n*], **–hswitch**}]

Arguments

none	Resets the entire system.
-partition	Resets the partition specified by *n*.
-qbb	Resets the QBB specified by the soft QBB number *n*.
-cpu	Resets the CPU specified by the soft CPU number *n*.
-pci	Resets the PCI box specified by the PCI ID number *n*.
-hswitch	Resets the hierarchical switch.

6.19 Set Commands

The set commands are used to set system parameters necessary for remote connections to the SCM command-language interpreter.

6.19.1 Set Alert Command

The set alert command specifies a string that defines the time to wait for the remote pager to respond to a dial-out, and a message to be sent to the remote pager. Two terminating characters (#;) are necessary.

Syntax

set alert *string*	Defines the wait time for the remote pager to respond, as well as a message to be displayed on the pager, usually the telephone number the remote user uses to reach the local system.

Table 6–3 Alert String Elements

String Elements	Description
,,,,,,	A series of commas, each specifying a 2-second wait for the remote pager to respond. The six commas shown would call for a wait of 12 seconds.
message	A message to be displayed at the remote pager; usually the number for the remote user to dial in to initiate remote access to the system. Must be terminated with the characters "#;".

Example

```
SCM_E0> set alert ,,,,,,9785551212#;
```

6.19.2 Set Baud Comand

The set baud command changes the baud rate between the system and the modem or between the system and the local terminal port.

Syntax

set baud [modem, local, com1] [2400, 3600, 4800, 7200, 9600, 19200, 38400, 57600]

or

set [modem_baud, local_baud, com1_baud] [2400, 3600, 4800, 7200, 9600, 19200, 38400, 57600]

Arguments

modem or **modem_baud**	Changes the baud rate of the SCM-to-modem UART.
local or **local_baud**	Changes the baud rate of the SCM-to-local-console-device UART.
com1 or **com1_baud**	Changes the baud rate to the SCM-to-system UART.

Options

2400 **3600** **4800** **7200** **9600** **19200** **38400** **57600**	The actual baud rate for the specified channel. The default for the modem port is 57600. The default for the local port is 9600.

Example

```
SCM_E0> set baud modem 38400
```

6.19.3 Set Com1_Mode Command

The set com1_mode command defines routes for information flow between the system and the local or remote console device.

Syntax

set com1_mode *mode*

Arguments

mode Defines the path of data flow between the local or modem port and the system. The values of *mode* are:

through	All characters routed through SCM. Default.
local	Only local port can communicate, but you can enter the SCM from the modem.
snoop	Data between remote and system bypasses the SCM, except SCM listens passively for SCM escape sequence, and if detected, enters SCM command-language interface. Useful for optimizing data transfer by bypassing the SCM except where specifically requested.
software_ bypass	Data flow between remote and system bypasses the SCM, except SCM listens for loss of carrier, then switches to snoop mode. Useful for binary download, as binary data may contain escape characters that the SCM would respond to.
firmware_bypass	Data flow between remote and system completely bypasses the SCM. Useful if you want the system, not the SCM to control the modem port and you want to disable SCM remote management features. You can switch to other modes by resetting the **com1_mode** environment variable from the SRM console.

6.19.4 Set Dial Command

The set dial command defines the number to be dialed out to when an alert occurs; usually the number of a pager.

Syntax

set dial *string* Defines the string sent to the system modem. See Table 6–4.

Table 6–4 Dial String Elements

String Elements	Description
ATDT	Enter characters either in all upper case or all lower case: AT = Attention D = Dial T = Tone (for touch-tone)
outside-line,	In the example below, "9" gets an outside line. Enter the number for an outside line if your system requires it. The comma causes a pause of 2 seconds to wait for the outside line.
pager-number	The telephone number of the pager to be activated.

Example

```
SCM_E0> set dial ATDT9,16035551212
```

See Section 3.4 for an overview of how this command is used in conjunction with others to prepare for remote access.

6.19.5 Set Escape Command

The set escape command allows you to change the escape sequence used for entering SCM mode. The escape sequence can be any printable character string, up to six characters.

Syntax

set escape *string*

Argument

string A string of up to six printable characters to be used as the escape sequence to enter into SCM mode. The default escape sequence is **<Esc><Esc>scm**. Note: the escape key on a PC keyboard is equivalent to <Ctrl>] (control key and right bracket).

6.19.6 Set <environment_variable> Command

The SCM has three environment variables.

Syntax

set {**hp_count** *n*, **hp_qbb_mask*n*** *mask*, **scm_csb_master_eligible** {*n, list*}}

Arguments

hp_count *n*	Defines the number of hard partitions that the system is to be divided into by the decimal number *n*, which may range from 0 – 8.
hp_qbb_mask*n mask*	Defines the QBBs to be included in hard partition *n* (0 – 7) by the hexadecimal value *mask*. A bit set in the *mask* indicates that the QBB is to be included in partition *n*.
scm_csb_master_eligible *n*	Set from within a hard partition to define whether the SCM for that partition is eligible (*n* = 1) or ineligible (*n* = 0 or not set) to become a master SCM.

Example

```
SCM_E0> set hp_count 3  ❶
SCM_E0> set hp_qbb_mask0 1
SCM_E0> set hp_qbb_mask1 2
SCM_E0> set hp_qbb_mask2 c
SCM_E0> set scm_csb_master_eligible 1    ❷
SCM_E0> power on -par 0
```

❶ Defines three hard partitions for the system. Hard partition 0 consists of QBB0. Hard partition 1 consists of QBB1. Hard partition 2 consists of QQB2 and QBB3. (See Chapter 4 for a detailed description of setting bit masks for the **hp_qbb_mask** environment variable.)

❷ The SCM node E0 is declared eligible to be selected as the master CSB node for the system.

6.19.7 Set Flow Command

The set flow command allows you to change the protocol used to handle flow control on one of the local, remote, or system COM1 ports.

Syntax

set [**local, modem, com1**] *protocol_type*

Options

local	Defines the protocol for the local port
remote	Defines the protocol for the remote port
com1	Defines the protocol for the COM1 port

Arguments

protocol_type	**hard**	Flow control protocol is handled by special cables that implement flow control signals. Typically used when a user wishes to transfer data that may contain software flow control characters that are not to be interpreted as such.
	soft	Flow control protocol implemented by software conventions defining certain characters that regulate flow, such as characters that define the start and end of a particular transmission. Binary transfers are not possible since data may inadvertently contain one of the special characters.
	both	Both hard and soft flow control may be used.
	none	No flow control is used.

6.19.8 Set Init Command

The set init command assigns the modem initialization string appropriate for the dial-out modem at the system site. The format of an initialization string is unique to each modem.

Syntax

set init *string* — Defines the parameter to be set as the initialization string for the local modem. See the user guide for your modem to determine how you want to set parameters controlling the modem's operation, and construct the *string* value accordingly. Table 6–5 lists some typical modem initialization strings.

Table 6–5 Modem Initialization Strings

Modem	Initialization String
Hayes Accura 336/56k	ate0v0&c1&k3s0=2
Compaq Microcom 510	ate0v0&c1s0=2
3Com USR Courier V.Everything	ate0v0&c1s0=2
3Com USR 56K* (5676) (Japanese)	ate0v0&c1s0=2
AT&T Dataport 14.4	ate0v0&c1s0=2
MultiTech MT5600ZDS	ate0v0&c1s0=2
Aiwa PV-BW5610 (Japanese)	ate0v0&c1s0=2
Omrom ME5614D (Japanese)	ate0v0&c1s0=2

Example

```
SCM_E0> set init ate0v0&c1s0=2
```

See Section 3.4 for an overview of how this command is used in conjunction with others to prepare for remote access.

6.19.9 Set Ocp_text Command

The set ocp_text command allows you to define a line of text to be displayed on the operator control panel after power-up. Typically used to identify a particular node in multi-system configurations.

Syntax

set ocp_text "*string*"

Argument

string Characters to be displayed on the OCP after power-up.

Example

```
SCM_E0> set ocp_text "Node A1"
```

6.19.10 Set Password Command

The set password command defines a password to be used by the remote user when dialing in to the system. The password must be set before the remote enable command is issued.

Syntax

set password *password*	The *password* argument consists of up to 14 characters. No escape characters can be used.

Example

```
SCM_E0> set password flamenco
```

See Section 3.4 for an overview of how this command is used in conjunction with others to prepare for remote access.

6.20 Show Commands

The show commands display the status of the requested components of the system.

6.20.1 Show CSB Command

The show csb command displays a list of all known nodes on the console serial bus. The CSB address is used to identify a particular node when using other SCM commands.

Syntax

show [**csb**]

Argument

csb Displays information about the nodes on the console serial bus.

Options

None

Example

```
SCM_E0> show csb ❶
❷    ❸          ❹                          ❺          ❻
CSB Type         Firmware Revision      FSL Revision  Power State

10  PBM          X05.7   (06.12/02:16)  X5.7 (06.09)  ON
11  PBM          X05.7   (06.12/02:16)  X5.7 (06.09)  ON
12  PBM          X05.7   (06.12/02:16)  X5.7 (06.09)  ON
13  PBM          X05.7   (06.12/02:16)  X5.7 (06.09)  ON
15  PBM          X05.7   (06.12/02:16)  X5.7 (06.09)  ON
16  PBM          X05.7   (06.12/02:16)  X5.7 (06.09)  ON
30  PSM          X05.7   (06.12/02:16)  X5.7 (06.09)  ON          SrvSw: NORMAL
30    XSROM      X05.7   (06.12/02:16)
c0    CPU0/SROM  V6.0-7                               ON
c1    CPU1/SROM  V6.0-7                               ON
c2    CPU2/SROM  V6.0-7                               ON
c3    CPU3/SROM  V6.0-7                               ON
c0    IOR0                                            ON
c1    IOR1                                            ON
c2    IOR2                                            ON
c3    IOR3                                            ON
31  PSM          X05.7   (06.12/02:16)  X5.7 (06.09)  ON          SrvSw: NORMAL
31    XSROM      X05.7   (06.12/02:16)
```

```
c4    CPU0/SROM  V6.0-7                                  ON
c5    CPU1/SROM  V6.0-7                                  ON
c6    CPU2/SROM  V6.0-7                                  ON
c7    CPU3/SROM  V6.0-7                                  ON
c4    IOR0                                               ON
c5    IOR1                                               ON
c6    IOR2                                               ON
c7    IOR3                                               ON
32  PSM          X05.7   (06.12/02:16)  X5.7 (06.09)   ON          SrvSw: NORMAL
32    XSROM      X05.7   (06.12/02:16)
c8    CPU0/SROM  V6.0-7                                  ON
c9    CPU1/SROM  V6.0-7                                  ON
ca    CPU2/SROM  V6.0-7                                  ON
cb    CPU3/SROM  V6.0-7                                  ON
c8    IOR0                                               ON
c9    IOR1                                               ON
33  PSM          X05.7   (06.12/02:16)  X5.7 (06.09)   ON          SrvSw: NORMAL
33    XSROM      X05.7   (06.12/02:16)
cc    CPU0/SROM  V6.0-7                                  ON
cd    CPU1/SROM  V6.0-7                                  ON
ce    CPU2/SROM  V6.0-7                                  ON
cf    CPU3/SROM  V6.0-7                                  ON
cc    IOR0                                               ON
cd    IOR1                                               ON
40  HPM          X05.7   (11.03/01:11)  T4.2 (09.08)   ON
e0  SCM MASTER   X05.7   (11.03/01:13)  T4.2 (09.08)   ON
e1  SCM SLAVE    X05.7   (11.03/01:13)  T4.2 (09.08)   ON          Ineligible
e2  SCM SLAVE    X05.7   (11.03/01:13)  T4.2 (09.08)   ON          Ineligible
e6  SCM SLAVE    X05.7   (11.03/01:13)  T4.2 (09.08)   ON          Ineligible
```

❶ The user enters the **show csb** command.

❷ This column identifies the address (hexadecimal) of each node currently on the console serial bus (CSB). You can use this address to identify nodes for SCM **examine**, **deposit**, **update**, and **test** commands.

❸ This column identifies the type of node on the console serial bus being described. This example shows seven PCI backplane managers (PBMs, four power system managers (PSMs), indicating four QBBs are present. Under each PSM are listed its XSROM and any CPUs (with attendant SROM and resident firmware). The HPM is the hierarchical switch power manager, and the SCM MASTER is the controlling firmware resident on the standard I/O module selected as the master.

❹ This column lists the firmware revision number and date the firmware was compiled.

❺ The FSL revision states the revision level of the fail-safe loader.

❻ This column lists the power state of each node. In this case, each node is receiving power. The SrvSw indicator for each PSM indicates either NORMAL (power on) or SERVICE (power off; the component can be serviced).

6.20.2 Show FRU Command

The show fru command displays information on field-replaceable units (FRUs) in the system.

Syntax

show fru

Example

```
SCM_E2> show fru
❶               ❷       ❸              ❹             ❺
FRUname         E  Part#               Serial#    Model/Other Alias/Misc
PBP0            02 54-25027-01.E01     NI92660355
PBP0.SIO        00 B4190-BA.A01        NI91260219
PBP0.RIO0       00    B4171-AA.E01     NI93671623 ..........  ..............
PBP0.RIO1       00 -B4171-AA.D02       NI93871224 RRISER      WF124A
PBP1            02 54-25027-01.E02     NI93772160 PCI_BPLANE   WF124
PBP1.RIO0       00 -B4171-AA.E01       NI94270223 RRISER      WF124A
PBP1.RIO1       00    -B4171-AA.E01    NI94270218 ..........  ..............
   .
   .
   .
QBB0            00 54-25043-02.D02     NI93870848 ..........  ..............
QBB0.PSM        00 54-25074-01.E03     NI92160238
QBB0.PWR        00 54-25017-01.E02     NI92660457 ..........  ..............
QBB0.AUX        00 54-25123-01.D01     NI90260771
QBB0.CPU0       00 7X-B4125-AA.B02     NI93260349 ..........  ..............
QBB0.CPU1       00 7X-B4125-AA.B02     NI93260309 ..........  ..............
QBB0.CPU2       00 7X-B4125-AA.B02     NI93260307 ..........  ..............
QBB0.CPU3       00 73-B4125-AA.B02     NI93470069 ..........  ..............
QBB0.MEM0       00    -B4150-AA.D02    NI93471470 ..........  ..............
QBB0.MEM0.DIM0  00 54-24941-FA.C03     AY92903143 ..........  ..............
QBB0.MEM0.DIM1  00 54-24941-FA.C03     AY92903017 ..........  ..............
QBB0.MEM0.DIM2  00 54-24941-FA.C03     AY92902981 ..........  ..............
QBB0.MEM0.DIM3  00 ................... ??000      ..........  ..............
QBB0.MEM0.DIM4  00 54-24941-FA.C03     AY92626235 ..........  ..............
QBB0.MEM0.DIM5  00 54-24941-FA.C03     AY92626256 ..........  ..............
QBB0.MEM0.DIM6  00 54-24941-FA.C03     AY92626234 ..........  ..............
QBB0.MEM0.DIM7  00 54-24941-FA.C03     AY92902979 ..........  ..............

QBB7.DIR        00    -B4140-AA.D01    NI93971621 ..........  ..............
QBB7.DIR0.DIM0  00 54-25023-BA.A01     AY92604007 ..........  ..............
QBB7.DIR0.DIM1  00 54-25023-BA.A01     AY92604006 ..........  ..............
QBB7.DIR0.DIM2  00 54-25023-BA.A01     AY92604484 ..........  ..............
QBB7.DIR0.DIM3  00 54-25023-BA.A01     AY92617927 ..........  ..............
QBB7.DIR0.DIM4  00 54-25023-BA.A01     AY92604007 ..........  ..............
QBB7.DIR0.DIM5  00 54-25023-BA.A01     AY92604006 ..........  ..............
QBB7.DIR0.DIM6  00 54-25023-BA.A01     AY92604484 ..........  ..............
QBB7.DIR0.DIM7  00 54-25023-BA.A01     AY92617927 ..........  ..............
QBB7.IO01       00    -B4170-AA.E01    NI94372518 ..........  ..............
QBB7.IO23       00    -B4170-AA.E01    NI94372517 ..........  ..............
QBB7.GP         00    -B4181-BB.A01    NI94870364 ..........  ..............
HSW8            00 B4188-AB.A01        SW00400007
```

```
HSW8.HPM0       00 54-25115-01.D01    NI93870661
HSW8.PWR1       00 54-30194-01.C03    NI94271569 ..........  ................
HSW8.PWR2       00 54-30194-01.C03    NI94271380 ..........  ................
```

❶ The name of the FRU; acronyms include:

PBP	PCI backplane
SIO	Standard I/O module
RIO	Remote I/O riser (located in PCI box)
QBB	Quad building block
PSM	Power system manager module
PWR	Main power module
AUX	Auxiliary power module
CPU	CPU module
MEM	Memory module
DIM	DIMM memory array
IO	Local I/O riser (located in QBB)
GP	Global port module

❷ Error field. FRUs without errors show 00 (hex). Other values are:

01	Hardware failure
02	TDD error has been logged
04	SDD error has been logged
08	Reserved
10	Checksum failure on bytes 0-62
20	Checksum failure on bytes 64-126
40	Checksum failure on bytes 128-524
80	FRU's system serial number does not match system's

❸ The part number by which the FRU can be ordered.

❹ The serial number of the FRU.

❺ Optional manufacturing specifications; usually blank.

6.20.3 Show nvram Command

The show nvram command displays the current settings of environment variables and parameters set in the NVRAM for the SCM.

Syntax

show nvr[**am**]

Example

```
SCM_E0> show nvram    ❶
COM1_PRINT_EN                   1
HP_COUNT                        4    ❷
HP_QBB_MASK0                    11
HP_QBB_MASK1                    22
HP_QBB_MASK2                    44
HP_QBB_MASK3                    88
HP_QBB_MASK4                    0
HP_QBB_MASK5                    0
HP_QBB_MASK6                    0
HP_QBB_MASK7                    0
SROM_MASK                       ff f
XSROM_MASK                      ff ff ff ff ff ff ff ff ff 1 0 0
PRIMARY_CPU                     ff
PRIMARY_QBB0                    ff
AUTO_QUIT_SCM                   1
FAULT_TO_SYS                    0
DIMM_READ_DIS                   0
SCM_CSB_MASTER_ELIGIBLE         1    ❸
PERF_MON                        20
SCM_FORCE_FSL                   0
OCP_TEXT                        Uninitialized
AUTO_FAULT_RESTART              1
SCM_E0>
```

❶ The user types the **show nvram** command.

❷ The system has four hard partitions.

❸ This SCM is eligible to become the SCM master.

6.20.4 Show Status Command

The show status command displays SCM status and information.

Syntax

show status

Example

```
SCM_E0> show status

  System Management Settings

SCM escape sequence           :    [ESC][ESC]scm    ❶

Local Baud/flow control       :    57600 / soft     ❷
COM1 Baud/flow control        :    57600 / hard
Modem Baud/flow control       :    57600 / hard
COM1 mode                     :    Pass-through
OCP power switch              :    Off     ❸
OCP halt                      :    Deasserted
OCP secure                    :    Non-Secure
Remote access                 :    Disabled     ❹
Remote user                   :    Not Connected
Alerts                        :    Disabled     ❺
Modem password                :    wffirmware     ❻
Modem init string             :    ate0v0&c1s0=2     ❼
Modem dial string             :    atdt915085551212
Modem alert string            :    ,,,,,,,5085551212#;
Alert pending                 :    No     ❽
Most recent alert             :    ❾
SCM_E0>
```

❶ The escape sequence that returns control to the SCM; in this case, the default.

❷ Settings of SRM environment variables controlling flow control, baud rate, and mode used for transmissions between the system and the local and remote console device.

❸ The current settings of the operator control panel (OCP) power, halt, and secure options.

❹ The state of remote access to the system; in this case, remote access is disabled, and no remote connection exists.

❺ Alerts have been disabled since no remote users currently allowed.

❻ The remote password is the default (wffirmware).

❼ Settings defining modem initialization, dial, and alert strings.

❽ No alerts are pending.

❾ There have been no alerts since the system was powered on.

6.20.5 Show System Command

The show system command displays an overall system summary, including the most recent system self-test results and the current temperature readings.

Syntax

show system

Example

```
SCM_E0> show system
System Primary QBB0 : 1                    ❶
System Primary CPU  : 1 on QBB1
❷    ❸      ❹    ❺         ❻                  ❼ ❽   ❾   ❿   ①

Par hrd/csb CPU  Mem   IOR3 IOR2 IOR1 IOR0  GP  QBB  Dir PS  Temp
     QBB#   3210 3210     (pci_box.rio)     Mod BP   Mod 321 (ºC)

(3) 0/30   --PP ---P  Px.x P2.0 Pf.1 Pf.0   P   P    P  PPP  28.0
(0) 1/31   --PP --PP  P4.0 Px.x Px.x Px.x   P   P    P  PPP  30.0
(2) 2/32   --PP ---P  Px.x P0.0 Px.x P3.0   P   P    P  PP-  26.5
(1) 3/33   --PP ---P  --.- --.- Px.x P5.0   P   P    P  PP-  28.1

HSwitch  Type    Cables 7 6 5 4 3 2 1 0    Temp(ºC)
                                ②          ③
HPM40    4-port         - - - - P P P P     26.0
④          ⑤                                ⑥  ⑦   ⑧
PCI Rise1-1  Rise1-0  Rise0-1  Rise0-0   RIO  PS  Temp
Cab 7 6 5 4   3 2 1   7 6 5 4   3 2 1    1 0  21  (ºC)

10  - - - -   - - -   - L - L   - - S    - P  -P  32.5
12  - - - -   - - -   - - - -   - L S    - P  P-  32.5
13  - - - -   - - -   - - L -   - - S    - P  P-  32.5
14  - - - -   - - -   - - L -   L - S    - P  PP  32.5
15  - - - -   - - -   L - - -   - - S    - P  PP  32.5
1f  - - - -   - - -   - - - H   L - S    P P  PP  32.5
```

❶ The display shows the system primary QBB and CPU.
❷ The hard partition number (if no hard partitions, a dash is displayed).
❸ The hard QBB number (defined by the location of the QBB; see Section 2.4) and the CSB address for the power system manager (PSM) for the QBB.
❹ CPU self-test results, identified by the slot number in the QBB. Note that each QBB has two CPUs and one or two memory modules.

❺ Memory self-test results, identified by the slot number in the QBB. Note that not all QBBs are fully populated with four memory modules.

❻ Status of I/O risers in the QBB.

P passed and powered on
p passed and powered off
m.n m = PCI box ID (0-f); n = remote I/O riser (0, 1)
x.x nothing is connected

❼ Global port self-test results for the QBB.

❽ QBB backplane self-test results.

❾ Directory module self-test results for the QBB.

❿ Status of the power supplies to the QBB. Note that QBBs 0 and 1 have redundant power supplies, while QBBs 2 and 3 do not.

① Temperature of the QBB backplane.

② This section of the report gives the status of the hierarchical switch. Possible values are

P Passed
V Valid; present, but may not be turned on
* present
- not present

③ Temperature surrounding the H-switch.

④ The CSB address of the PBM for each PCI box. This is equal to the PCI box ID (0 – f) with 10 (hex) added.

⑤ The status of the slots in the PCI boxes connected to the I/O risers.

S Standard I/O
* present
- not present

L Low power dissipation
M Medium power dissipation
H High power dissipation

⑥ Status of remote I/O riser

P = present
- = not present.

⑦ Status of power supplies to the PCI cabinets.

P passed and powered on
p passed and powered off
F failed and powered on
f failed and powered off
- not present
* present but no status yet

⑧ Temperature of the PCI box.

6.21 Test Alert Command

The test alert command tests the modem setup and connection to the remote pager or computer. The modem must be initialized, alerts must be enabled, and the OCP keyswitch must not be in the Secure position before this command can be issued.

Syntax

test alert

Example

```
SCM_E0> test alert
```

6.22 Test &pc*n* Command

The test &pc*n* command allows you to run tests on CPUs.

Syntax

test -&pc*n test_number*

Argument

-&pc*n* Defines the processor to be tested, where *n* is the soft CPU number of the processor as shown in a **show csb** command. The *test_number* parameter gives the number of the test to be run. See the *AlphaServer GS80/160/320 Service Manual* for a discussion of the tests that can be run for processors.

Example

```
SCM_E0> test -&pc8 1
```

Chapter 7
SRM Command Reference

This chapter describes the SRM console program's command language.

Sections in this chapter include:

- SRM Command Overview
- SRM Command Description Conventions
- SRM Command Language Environment Variables
- SRM Console Shell Syntax
- SRM Console Commands (in alphabetical order)

boot	**deposit**	**grep**	**lpinit**	**nettest**	**scm**
cat	**edit**	**halt**	**ls**	**nvram**	**set**
clear	**examine**	**help**	**man**	**power**	**show**
continue	**exer**	**info**	**memexer**	**ps**	**show_status**
crash	**galaxy**	**init**	**memexer_mp**	**reset**	**sys_exer**
csr	**gct**	**kill**	**migrate**	**rm**	**test**
		kill_ diags	**more**	**run bios**	**wwidmgr**

When the system or (partition) is in SRM console mode, the operating system is halted and the SRM console firmware is executing. The operator communicates with the firmware through the console terminal, which displays the following prompt:

P*nn*>>>

where *nn* may range from 00 to 1F (for 32-processor systems). The value depends on which processor module is the primary processor, as determined during power-up.

7.1 SRM Command Overview

The SRM console firmware supports capabilities for the operating systems offered with AlphaServer GS80/160/320 series systems.

Table 7–1 Summary of SRM Console Commands

Command	Function
boot	Boots the operating systems.
cat	Displays the named file.
clear	Clears the SRM **password** or an environment variable.
continue	Resumes processing after a Ctrl/P is issued (*OpenVMS* systems).
crash	Forces a crash dump of the operating system.
csr	Displays contents of control and status registers.
deposit	Writes data to the specified address.
edit	Invokes the console line editor, which can be used to edit a RAM file or the user power-up script, "nvram," which is always invoked during the power-up sequence.
examine	Displays the contents of a memory location or device register.
exer	Exercises one or more devices by performing specified read, write, and compare operations.
galaxy	Initializes soft partitions (same as **lpinit**).
gct	Rebuilds the Galaxy configuration tree in SRM main memory.
grep	Globally searches for regular expressions and prints matches.
halt	Halts the specified processor or device.
help (or **man**)	Displays information about all or a specific SRM command.

Table 7–1 Summary of SRM Console Commands (Continued)

Command	Description
info	Displays registers and data structures.
init	Stores any changes made to environment variables and reinitializes the hardware.
kill	Stops a process that is running on the system.
kill_diags	Stops all console-based diagnostic processes running on the system.
lpinit	Initializes soft partitions (same as **galaxy**).
ls	Displays names of files on the system.
memexer	Runs a requested number of memory tests in the background.
memexer_mp	Exercises ability of CPUs to share data and remain coherent by running memory tests on all CPUs.
migrate	Moves one or all CPUs to a given soft partition.
more	Displays a file one screen at a time.
nettest	Runs loopback tests for PCI-based Ethernet ports. Also used to test a port on a "live" network.
nvram	Runs the **nvram** script.
power	Turns power on or removes power from the specified CPU, I/O riser, or PCI box.
ps	Displays process status and statistics.
rm	Removes files from the file system.
scm	Executes an SCM command from the SRM console prompt.
set ***envar***	Sets the value of an environment variable.
show ***envar***	Displays the state of all or a specified environment variable.
show config	Displays the configuration at the last system initialization.
show device	Displays the controllers and bootable devices in the system.
show fru	Displays the configuration of field-replaceable units (FRUs).

Table 7–1 Summary of SRM Console Commands (Continued)

Command	Description
show memory	Displays memory module information.
show pal	Displays version of *Tru64 UNIX* and *OpenVMS* PALcode.
show_status	Displays the progress of diagnostic tests. Reports one line of information for each executing diagnostic.
show version	Displays the version of the SRM console program.
sys_exer	Exercises the entire system.
test	Tests the entire system.
wwidmgr	Manages the WWID registration (Fibre Channel).

Table 7–2 Notation Formats for SRM Console Commands

Command Parameter	Attribute or Action
Length	Up to 255 characters, not including the terminating carriage return or any characters deleted as the command is entered. To enter a command longer than 80 characters, use the backslash character for line continuation (see Table 7–3).
Case	Upper- or lowercase characters can be used for input. Characters are displayed in the case in which they are entered.
Abbreviation	Only by dropping characters from the end of words. You must enter the minimum number of characters to identify the keyword unambiguously. Abbreviation of environment variables is allowed with the **show** command.
Options	You can use command options, to modify the environment, after the command keyword or after any symbol or number in the command. See individual command descriptions for examples.
Numbers	Most numbers are decimal. Addresses and numbers used with the **deposit** command are hexadecimal.
No characters	A command line with no characters is a null command. The console program takes no action and does not issue an error message; it returns the console prompt. The SRM console supports command-line recall (using the up and down arrow keys) and editing, including use of the right and left arrow keys.
Spaces or tabs	Multiple adjacent spaces and tabs are compressed and treated as a single space. Leading and trailing spaces are ignored.

Table 7–3 Special Characters for the SRM Console

Character	Function
Return or Enter	Terminates a command line. No action is taken on a command until it is terminated.
Backslash (\)	Continues a command on the next line. Must be the last character on the line to be continued.
Delete	Deletes the previous character.
Ctrl/A	Toggles between insertion/overstrike mode. The default is overstrike.
Ctrl/B or up-arrow	Recalls previous command(s). The last 16 commands are stored in the recall buffer.
Ctrl/C	Terminates the running process. Clears Ctrl/S; resumes output suspended by Ctrl/O. When entered as part of a command line, deletes the current line. Ctrl/C has no effect as part of a binary data stream.
Ctrl/D or left-arrow	Moves the cursor left one position.
Ctrl/E	Moves the cursor to end of line.
Ctrl/F or right-arrow	Moves the cursor right one position.
Ctrl/H	Moves the cursor to beginning of the line.
Ctrl/J	Deletes the previous word.
Backspace	Deletes one character.
Ctrl/O	Stops output to console terminal for current command. Toggles between enable/disable. The output can be reenabled by other means as well: when the console prompts for a command, issues an error message, or enters program mode, or when Ctrl/P is entered.
Ctrl/P	Ignored in SRM mode. In program mode, on the *OpenVMS* operating system, causes the boot processor to halt and begin running the SRM console program.

Table 7–3 Special Characters for the SRM Console (Continued)

Character	Function
Ctrl/Q	Resumes output to the console terminal that was suspended by Ctrl/S.
Ctrl/R	Redisplays the current line. Deleted characters are omitted. This command is useful for hardcopy terminals.
Ctrl/S	Suspends output to the console terminal until Ctrl/Q is entered. Cleared by Ctrl/C.
Ctrl/U	Deletes the current line.
*	Wildcarding for certain commands such as **show**.
" "	Double quotes let you denote a string for assignment as an environment variable name.
#	Specifies that all text between it and the end of the line is a comment. Control characters are not considered part of a comment.

7.2 SRM Command Description Conventions

The following conventions are used in the descriptions of the SRM console command language.

Table 7–4 SRM Command Description Conventions

Convention	Meaning
`fixed-font`	SCM command examples are shown in a small fixed-width font.
bold	Command and option keywords are presented in bold type.
item	Italics indicate a placeholder for an item that the user supplies.
[*item*]	Square brackets are used to enclose optional parameters, qualifiers, and values. For example, **`help`** `[topic]`.
{*a*, *b*, *c*}	Braces containing items separated by commas imply mutually exclusive values. For example {*a*, *b*, *c*} indicates that you can choose one of *a*, *b*, or *c*.
{*a* | *b* | *c*}	Braces containing items separated by the vertical bar indicate that you can choose any combination of *a*, *b*, and *c*.

7.3 SRM Command Language Environment Variables

Environment variables allow the user to modify the way the console commands operate.

An environment variable is a name and value association maintained by the console program. The value associated with an environment variable is an ASCII string (up to 127 characters in length) or an integer. Some environment variables can be set to tailor the recovery behavior of the system on power-up and after system failures.

Volatile environment variables are initialized to their default by a system reset. Nonvolatile environment variables stay set across system failures.

Environment variables can be created, modified, displayed, and deleted using the SRM commands **create**, **set**, **show**, and **clear**. A default value is associated with any variable that is stored in the EEPROM area.

Table 7–5 (in the description of the **set** command, Section 7.5.33) lists the console environment variables, their attributes, and their functions.

7.4 SRM Console Shell Syntax

The SRM console command language is modeled after the Bourne shell which runs under the UNIX operating system. Some useful features of this implementation are described here.

7.4.1 I/O Pipes

A pipeline is a sequence of one or more commands separated by the pipe operator "|". The output of each command with the exception of the last command is used as input to the next command. For example, to locate SCSI devices in a system, pipe the output of the **show device** command into the **grep** command:

```
P00>>> show device | grep dk
dka0.0.0.1.0               DKA0                        RZ1DF-BF  1614
dkb0.0.0.7.1               DKB0              COMPAQ BB00911CA0  3B05
dkb100.1.0.7.1             DKB100            COMPAQ BB00911CA0  3B05
dkb200.2.0.7.1             DKB200            COMPAQ BB00911CA0  3B05
dkb300.3.0.7.1             DKB300            COMPAQ BB00911CA0  3B05
```

7.4.2 Redirecting Output

With the lengthy output provided by some of the commands, it may be useful to direct output to a file that can be examined with the **cat** or **more** command. You can direct the output of a command into a file using the output operator ">". For example:

```
P00>>> show config > cfgtemp
P00>>> more cfgtemp

    [first screen of show config output]

P00>>>
```

7.4.3 Background Operator

The background operator "&" is used at the end of the command line to execute command sequences in the background as a separate process. This is especially useful when starting concurrent tests or exercisers on the system. For example:

```
P00>>> memtest -sa 2000000 -ea 3000000 -p 0 &
P00>>> show_status
  ID        Program    Device   Pass   Hard/Soft Bytes Written   Bytes Read
--------    --------   ------   ------ --------- -------------   ----------
00000001       idle    system        0   0   0               0            0
00000184    memtest    memory   832438   0   0     6819323904    6819323904
```

7.5 SRM Console Commands

SRM commands allow you to examine and modify system state. Additionally, you can test specific components of the system.

7.5.1 Boot Command

The boot command boots the *Tru64 UNIX* and *OpenVMS* operating systems and the Loadable Firmware Update (LFU) utility.

Syntax

b[**oot**] [**-file** *filename*] [**-flags** [*value*]] [**-halt**] [**-protocols** *enet_protocol*] [*boot_dev*]

Options

-file *filename* Specifies the name of the file to load into the system. Use the **set boot_file** command to set a default bootfile. See Section 7.5.33.

NOTE: *For booting from Ethernet, the filename is limited by the MOP V3 load protocol to 15 characters. The MOP protocol is used with OpenVMS systems.*

-flags [*value*] Provides additional operating system-specific boot information. In *Tru64 UNIX*, specifies boot flags. In *OpenVMS*, specifies the system root number and boot flags. Preset default boot flag values are 0,0. Use the **set boot_osflags** command to change the default boot flag values. See Section 7.5.33.

-halt Forces the bootstrap operation to halt and invoke the SRM console program. The console is invoked after the bootstrap image is loaded and page tables and other data structures are set up. Console device drivers are not shut down. Transfer control to the bootstrap image by entering the **continue** command.

protocols
enet_protocol

Specifies the Ethernet protocol to be used for the network boot. Either **mop** (for *OpenVMS*) or **bootp** (for *Tru64 UNIX*) may be specified. Use the **set_ew*0_protocols** command to set a default network boot protocol. See Section 7.5.33.

Arguments

boot_dev

A device path or list of devices from which the SRM console program attempts to boot. Use the **set bootdef_dev** command to set a default boot device. See Section 7.5.33.

Entering values for boot flags, the boot device name, or Ethernet protocol with the **boot** command overrides the current default value for the current boot request, but does not change the corresponding environment variable. For example, if you have defined a value for **boot_osflags** and you specify the **–flags** option on the **boot** command line, the **–flags** argument takes precedence for that boot session.

Example (Boot Command)

```
P00>>> b dkb4 -fl A -file vmunix     ❶
(boot dkb400.4.0.1.2 -file vmunix -flags A)
block 0 of dkb400.4.0.1.2 is a valid boot block
reading 13 blocks from dkb400.4.0.1.2
bootstrap code read in
base = 344000, image_start = 0, image_bytes = 1a00
initializing HWRPB at 2000    ❷
initializing page table at 3fffb2000
initializing machine state
setting affinity to the primary CPU
jumping to bootstrap code    ❸
    .
    .
    .
Digital UNIX Version V4.0 (wfsys6.eng.pko.dec.com) console
login: root    ❹
Password:
Last login: Wed Sep  8 23:49:02 on console
   .
   .
   .
sys6>    ❺
```

The example shows an operator booting the *Tru64 UNIX* operating system, using the SRM console command **boot** (abbreviated as **b** in the example). The SRM console operates in a processor on the system called the *primary processor,* selected during power-up, as described in Chapter 2.

❶ The operator issues the **boot** command. The boot disk is named directly as **dkb4**, the **–fl** option is used to request boot flag **A**, which boots the system disk to multiuser mode for *Tru64 UNIX*. The **–file** option specifies the name of the file to boot. (See Appendix A for a list of all boot options.)

❷ The HWRPB (hardware restart parameter block) is an area in memory where parameters are passed between the operating system and the SRM console. (The HWRPB is discussed in detail in the *Alpha Architecture Manual*.)

❸ The SRM jumps to bootstrap code that has been loaded into memory from the boot device to boot the operating system from the disk named in the **boot** command.

❹ The *Tru64 UNIX* operating system is given control and displays the prompt for login name and password.

❺ The operator is at the operating system prompt and can initiate operations to be performed by the operating system.

7.5.2 Cat Command

This command concatenates files that you specify to the standard output. If you do not specify files on the command line, cat copies standard input to standard output.

Syntax

cat [**-length** *n*] [**-block** *n*] [**-start** *offset*] [**-quiet**] *file…*

Options

-length *n*	Specifies the number of bytes in hex of each input file to copy.
-block *n*	Size of the internal buffer **cat** uses to copy files, in hex. By default, this is DEF_ALLOC (2048) bytes. For performance in copying disks, a number such as 10000 can be used.
-start *n*	Specifies the offset to seek to in hex. If the file(s) are not seekable, then this qualifier has no effect.
-quiet	Uses silent mode on fopens.

Argument

file…	The name of the input file or files to be copied.

Example

```
P00>>> cat el    ❶

starting console on CPU 0    ❷
initialized idle PCB
initializing semaphores
initializing heap
initial heap 2c0c0
memory low limit = 1f6000
heap = 2c0c0, 1ffc0
initializing driver structures
initializing idle process PID
initializing file system
initializing timer data structures
lowering IPL
CPU 0 speed is 731 MHz
create dead_eater
```

```
create poll
create timer
create powerup
access NVRAM
QBB 0 memory, 8 GB
QBB 1 memory, 8 GB
total memory, 16 GB
probe I/O subsystem
probing hose 0, PCI
probing PCI-to-ISA bridge, bus 1
    .
    .
    .
Change to Internal loopback.
Change to Normal Operating Mode.
Change to Internal loopback.
Change to Normal Operating Mode.
fwb0.0.0.3.8 StateExpt = 4 StateRcv = 5
fwb0.0.0.3.8 StateExpt = 4 StateRcv = 5
fwb0.0.0.3.8 StateExpt = 4 StateRcv = 5
fwb0.0.0.3.8 StateExpt = 4 StateRcv = 5
P00>>>
```

❶ The user types the command **cat el** to display the event log on the console device.

❷ The event log for the current period of execution is displayed.

7.5.3 Clear Command

This command clears the SRM password or an environment variable.

Syntax

clear {**password,** *environment_variable*}

Arguments

password	The **clear password** command is used in conjunction with the **set secure**, **set password**, and login commands. The **clear password** command clears the password; there must be a valid password and the console must be logged in for the command to function.
environment_ variable	Clears the named environment variable, if it is volatile (including environment variables created by the user with the **set** command). Will not clear nonvolatile environment variables.

Example

```
P00>>> show hp_count    ❶

hp_count                2

P00>>> clear hp_count
P00>>> show hp_count

hp_count                0

P00>>> clear password    ❷
Please enter the password:
Password successfully cleared
P00>>>
```

❶ The **show** command shows the value of the **hp_count** environment variable to be 2. The **clear hp_count** command clears it to 0.

❷ The **clear password** command prompts for the current password. When it is entered, the password is cleared, and a message to that effect is displayed.

7.5.4 Continue Command

For OpenVMS systems, the continue command resumes processing at the point where it was interrupted by a Ctrl/P at the console terminal, by the Halt button on the operator control panel, or by an SCM halt in command.

Syntax

c[ontinue]

Example

```
P00>>> continue  ❶

continuing CPU   ❷

<Enter>  ❸

$    ❹
```

❶ The user types the **continue** command.

❷ The SRM console code prints the "continuing CPU" message.

❸ Control returns to the operating system. No displays are forthcoming, so the user presses the Enter (or Return) key to see what happens.

❹ In this case, the *OpenVMS* system was waiting for command line input. When the user enters Return, it redisplays the $ prompt.

7.5.5 Crash Command

The crash command causes the operating system to be restarted and generates a memory dump.

Syntax

cra[sh]

Example

```
P00>>> crash    ❶

CPU 0 restarting

DUMP: 1983738 blocks available for dumping
DUMP: 118178 wanted for a partial compressed dump.
DUMP: Allowing 2060017 of the 2064113 available on 0x800001
device string for dump = SCSI 1 1 0 0 0 0 0.
DUMP.prom: dev SCSI 1 1 0 0 0 0 0, block 2178787
DUMP: Header to 0x800001 at 2064113 (0x1f7ef1)
device string for dump = SCSI 1 1 0 0 0 0 0.
DUMP.prom: dev SCSI 1 1 0 0 0 0 0, block 2178787
DUMP: Dump to 0x800001: .......: End 0x800001
device string for dump = SCSI 1 1 0 0 0 0 0.
DUMP.prom: dev SCSI 1 1 0 0 0 0 0, block 2178787
DUMP: Header to 0x800001 at 2064113 (0x1f7ef1)
succeeded
halted CPU 0
halt code = 5
HALT instruction executed
PC = fffffc0000568704
P00>>>
```

❶ The **crash** command restarts the operating system and generates a memory dump.

This command is useful when the operating system is hung. The user can return to the SRM console, and then issue the **crash** command to restart the operating system and generate a memory dump.

7.5.6 CSR Command

The csr command displays the contents of the system's control and status registers (CSRs). If a hex data value is specified, the command deposits to the specified register or registers before displaying.

Syntax

csr [*name* [*data*]]

Arguments

name Name of the CSR register to be displayed, and if data is supplied, deposited to and then displayed. Wildcarding is permissible (see example). If no name is specified, all registers are displayed or deposited to.

data A hexadecimal value to be deposited in the named register or registers.

Example

```
P00>>> csr *scratch ❶
       ❷                                    ❸          ❹
CSR Name                                  CSR Address   CSR Data
----------------------------------------  -----------   ----------------
QBB0.QSD.CPU0_SCRATCH                     fffff940800   0000000000000000
QBB0.QSD.CPU1_SCRATCH                     fffff940840   0000000000000000
QBB0.QSD.CPU2_SCRATCH                     fffff940880   0000000000000000
   .
   .
   .
QBB0.IOP.IOA_SCRATCH                      fffffa01400   0000000a0000000a
QBB0.IOP.IOD_SCRATCH                      fffffa01440   aaaaaaaaaaaaaaaa
QBB0.IOPort0.PCA_SCRATCH                  fffffb00140   aaaaaaaaaaaaaaaa
QBB0.IOPort3.PCA_SCRATCH                  fffffbc0140   aaaaaaaaaaaaaaaa
QBB0.IOPort0.NE_SCRATCH                   fffffb10240   aaaaaaaaaaaaaaaa
QBB0.IOPort3.NE_SCRATCH                   fffffbd0240   aaaaaaaaaaaaaaaa
QBB0.IOPort0.FE_SCRATCH                   fffffb18240   aaaaaaaaaaaaaaaa
QBB0.IOPort3.FE_SCRATCH                   fffffbd8240   aaaaaaaaaaaaaaaa
QBB0.GP.GPA_SCRATCH                       fffffc02800   0000000000000000
QBB0.GP.GPD_SCRATCH                       fffffc21000   0000000000000000
P00>>> csr iop*a*scratch cedebeadfeedface ❺
CSR Name                                  CSR Address   CSR Data
----------------------------------------  -----------   ----------------
QBB0.IOP.IOA_SCRATCH                      fffffa01400   cedebeadfeedface
QBB0.IOPort0.PCA_SCRATCH                  fffffb00140   cedebeadfeedface
QBB0.IOPort3.PCA_SCRATCH                  fffffbc0140   cedebeadfeedface
P00>>>
```

❶ The user issues a **csr** command to display information about the control and status registers. In this case, all registers ending with the characters “scratch” are requested.

❷ The names of the control and status registers.

❸ The addresses of the registers.

❹ The contents of the registers.

❺ The user issues a **csr** command. In this case, all registers beginning with the letters “iop”, containing an “a”, and terminating with “scratch” are to be displayed. In addition, the hexadecimal value “cedebeadfeedface” is to be deposited in each register before it is displayed.

7.5.7 Deposit and Examine Commands

The deposit command stores data in an address that you specify: a memory location, a register, a device, or a file. The examine command displays from the same locations.

Syntax

d[**eposit**] [- {**b**, **w**, **l**, **q**, **o, h**}], [-{**physical**, **virtual**, **gpr**, **fpr**, **ipr**}]
[**-n** *count*] [**-s** *step*] [*device***:**] *address data*

e[**xamine**] [**-**{**b, w, l, q, o, h**}] [-{**physical**, **virtual**, **gpr**, **fpr**, **ipr**}]
[**-n** *count*] [**-s** *step*] [*device*:] *address*

Options

-b The *data* deposited is a byte (8 bits).

-w The *data* deposited is a word (16 bits).

-l The *data* deposited is a longword (32 bits).

-q The *data* deposited is a quadword (64 bits). This is the default.

-o The *data* deposited is an octaword (128 bits).

-h The *data* deposited is a hexword (256 bits).

-gpr The address space is general-purpose registers.

-ipr The address space is internal processor registers.

-fpr The address space is floating-point registers.

-physical The address space is physical memory.

-virtual The address space is virtual memory.

-n *count* The address will be incremented *count* (hex) times.

-s *step* The increment size (hex). Normally this defaults to the data size, but is overriden by the presence of this qualifier. This option must be specified each time; it does not apply to following **deposit** or **examine** commands.

Arguments

device: The optional device name (or address space) selects the device to access. Possible values are:

pmem: Physical memory
vmem: Virtual memory. All access and protection checking occur. If the access would not be allowed to a program with the current PS, the SRM console issues an error message. If memory mapping is not enabled, virtual adressses are equal to physical addresses.

gpr: General purpose register set R0 – R31
Data size default = q
fpr: Floating-point register set, F0-F31
Data size default = q
pt: PAL temporary register set PT0-PT31
Data size default = q
eerom: 8 KB NVRAM
flash: 2 MB flash EEPROM
ipr: Internal processor register
pcicfg: PCI configuration space
pciio: PCI I/O space
pcimem: PCI memory space
psr: Processor status register
toy: Time of year clock

address An address that specifies the offset within a device into which data is deposited. The address may be any valid hexadecimal offset in the device's address space.

data The data (hex) to be written to the specified address or register.

Symbolic forms can be used for the address. They are:

pc The program counter. The address space is set to GPR.

+ The location immediately following the last location referenced in a **deposit** or **examine** command. For physical and virtual memory, the referenced location is the last location plus the size of the reference (1 for byte, 2 for word, etc.) For other address spaces, the address is the last referenced address plus 1.

- The location immediately preceding the last location referenced in a

deposit or **examine** command. Memory and other address spaces are handled as above.

* The last location referenced in a **deposit** or **examine** command.

@ The location address by the last location referenced in a **deposit** or **examine** command.

Example

```
P00>>> deposit -q -p -n 3 0 a5a5a5a5   ❶
P00>>> examine -q -p -n 10 0   ❷
pmem:                0 00000000A5A5A5A5   ❸
pmem:                8 00000000A5A5A5A5
pmem:               10 00000000A5A5A5A5
pmem:               18 00000000A5A5A5A5
pmem:               20 0000000000000000
pmem:               28 0000000000000000
pmem:               30 0000000000000000
pmem:               38 0000000000000000
pmem:               40 0000000000000000
pmem:               48 0000000000000000
pmem:               50 0000000000000000
pmem:               58 0000000000000000
pmem:               60 0000000000000000
pmem:               68 0000000000000000
pmem:               70 0000000000000000
pmem:               78 0000000000000000
pmem:               80 0000000000000000
P00>>>
```

❶ The **deposit** command deposits four quadwords (the original deposit plus three increments) with the value a5a5a5a5 in physical memory beginning at location 0.

❷ The **examine** command requests the display of 11 (hexadecimal) quadwords of physical memory beginning at location 0 and incrementing this address 10 (hexadecimal) times.

❸ The value a5a5a5a5 has been stored in the first four memory locations, as the display shows.

7.5.8 Edit Command

The system comes with a nonvolatile file named "nvram" that is stored in EEROM on the standard I/O module. The nvram file is a user-created power-up script (set of commands) that is always invoked during the power-up sequence. Use the edit command to create or alter the nvram script.

Syntax

edit *file*

Argument

file The name of the file to be edited. Most commonly used to create and edit the file named **nvram.**

Description

You can create an nvram script to include any commands you want the system to execute at power-up. You create and edit the nvram script using the SRM **edit** command. With **edit**, lines may be added, overwritten, or deleted. To clear the script, enter the existing line numbers without any text. This deletes the lines.

Once you issue the **edit** command, the editor displays informative messages (see example) and displays an asterisk prompt (*). You can then use the following commands:

help	Displays the brief help file.
list	Displays the current file prefixed with line numbers.
renumber	Renumbers the lines of the file in increments of 10.
exit	Leaves the editor and closes the file, saving all changes.
quit	Leaves the editor and closes the file without saving changes.
nn	Deletes line number *nn*.
nn text	Adds or overwrites line number *nn* with the specified text.

Example

```
P00>>> edit nvram  ❶
editing 'nvram'        ❷
0 bytes read in
*10 set mopv3_boot 1  ❸
*exit    ❹
17 bytes written out to nvram
P00>>> edit nvram  ❺
editing 'nvram'
20 bytes read in
*list   ❻
10 set mopv3_boot 1
*10   ❼
*quit    ❽
17 bytes written out to nvram
P00>>>
```

❶ The user issues the command to edit the nvram file.

❷ The editor displays the name of the file being edited and the number of bytes currently in the file.

❸ The user creates a line numbered "10" with a command that sets the **mopv3_boot** environment variable to 1. (By default, MOP boots send four MOPV4 requests before defaulting to MOPV3. This environment variable setting forces the SRM console to bypass MOP V4 requests. This speeds up MOP booting on networks with MOP V3 software.)

❹ The **exit** command writes the file named nvram to EEROM on the standard I/O module, including the new line, and returns control to the SRM console code.

❺ The user again issues a command to edit the file named nvram.

❻ The **list** command displays the contents of the file.

❼ The user enters the line number 10 without text, deleting the line.

❽ The **quit** command writes the file to the standard I/O module without the changes made during the current editing session. The file is unchanged.

CAUTION: *An inappropriate command in the nvram script can disable the system. For example, the **init** command will cause the system to go into an endless loop.*

To correct this error, press the Halt button during power-up. When the Pnn>>> prompt is displayed, edit the script to remove the improper command.

7.5.9 Exer Command

The exer command exercises one or more devices by performing specified read, write, and compare operations. Advanced users may want to use the specific options described here.

CAUTION: *Running **exe**r on disks can destroy data on the disks.*

Syntax

exer [**-sb** *start_block*] [**-eb** *end_block*] [**-p** *pass_count*]
[**-l** *blocks*] [**-bs** *block_size*] [**-bc** *blocks_per_io*]
[**-d1** *buf1_string*] [**-d2** *buf2_string*]
[**-a** *action_string*] [**-sec** *seconds*] [**-m**] [**-v**] [**-delay** *millisecs*] *device_name*

Options

-sb *start_block*	Specifies the starting block number (hex) within the filestream. The default is 0.
-eb *end_block*	Specifies the ending block number (hex) within the filestream. The default is 0.
-p *pass_count*	Specifies the number of passes to run the exerciser. If 0, then run forever or until Ctrl/C. The default is 1.
-l *blocks*	Specifies the number of blocks (hex) to exercise. The option **l** has precedence over **eb**. If only reading, then not using either **–l** nor **–eb** defaults to read until end-of-file. If writing, and neither **–l** or **–eb** are specified, then **exer** will write for the size of device. The default for *blocks* is 1.
-bs *block_size*	Specifies the block size (hex) in bytes. The default is 200 (hex).
-bc *blocks_per_io*	Specifies the number of blocks (hex) for each I/O operation. On devices without length (tape), use the specified pack size or default to 2048. The maximum block size allowed with variable-length block reads is 2048 bytes. Default = 1.
-d1 *buf1_string*	String argument for eval to generate buffer 1 data pattern from. Buffer 1 is initialized only once before any I/O occurs.

Default = all bytes set to hex 5As.

-d2 *buf2_string* String argument for eval to generate buffer 2 data pattern from. Buffer 2 is initialized only once before any I/O occurs. Default = all bytes set to hex 5As.

-a *action_string* Specifies an exerciser action string that determines the sequence of reads, writes, and compares to various buffers. The default action string is ?r. The action string characters are:

- r Read into buffer 1
- w Write from buffer 1
- R Read into buffer 2
- W Write from buffer 2
- n Write without lock from buffer 1
- N Write without lock from buffer 2
- c Compare buffer1 with buffer 2
- \- Seek to file offset prior to last read or write
- ? Seek to a random block offset within the specified range of blocks. **exer** calls the program, random, to "deal" each one of a set of numbers once. **exer** chooses a set that is a power of two and is greater than or equal to the block range. Each call to random results in a number that is then mapped to the set of numbers that are in the block range and **exer** seeks to that location in the filestream. Since **exer** starts with the same random number seed, the set of random numbers generated will always be over the same set of block range numbers.
- s Sleep for a number of milliseconds specified by the **delay** qualifier. If no **delay** qualifier is present, sleep for 1 millisecond. Note: Times as reported in verbose mode will not necessarily be accurate when this action character is used.
- z Zero buffer 1
- Z Zero buffer 2
- b Add constant to buffer 1
- B Add constant to buffer 2

-sec *seconds*	Specifies termination of the exercise after the number of seconds have elapsed. By default, the exerciser continues until the specified number of blocks of passes are processed.
-m	Specifies metric mode. At the end of the exercise, a total throughput line is displayed.
-v	Specifies verbose mode. Data read is also written to the standard output. This is not applicable on writes or compares. The default is verbose mode off.
-delay *millisecs*	Specifies the number of milliseconds to delay when "s" appears as a character in the action string.

Description

The **exer** command reports performance statistics:

- A read operation reads from a specified device into a buffer.
- A write operation writes from a buffer to a specified device.
- A compare operation compares the contents of the two buffers.
- The **exer** command uses two buffers, buffer 1 and buffer 2, to carry out the operations. A read or write operation can be performed using either buffer. A compare operation uses both buffers.

Examples

```
P00>>> exer dk*.* -p 0 -secs 36000    ❶
P00>>> exer -l 2 dkb0    ❷
P00>>> exer -sb 1 -eb 3 -bc 4 -a 'w' -d1 '0x5a' dka100    ❸
P00>>> exer -eb 64 -bc 4 -a '?w-Rc' dkb0    ❹
P00>>> exer -a '?r-w-Rc' dka400    ❺
```

❶ Reads all SCSI type disks for the entire length of each disk. Repeat this until 36000 seconds (10 hours) have elapsed. All disks will be read concurrently. Each block read will occur at a random block number on each disk.

❷ Read block number 0 and 1 from device dkb0.

❸ Write hex 5As to every byte of blocks 1, 2, and 3 of dka100. The packet size is bc times bs, or 4 times 512, or 2048 for all writes.

❹ A destructive write test over block numbers 0 through 100 on disk dkb0. The packet size is 2048 bytes. The action string specifies the following sequence of operations:

1. Set the current block address to a random block number on the disk between 0 and 97. A four-block packet, starting at block numbers 98, 99, or 100 would access blocks beyond the end of the length to be processed, so 97 is the largest possible starting block address of a packet.
2. Write a packet of hex 5As from buffer 1 to the current block address.
3. Set the current block address to what it was just prior to the previous write operation.
4. From the current block address, read a packet into buffer 2.
5. Compare buffer 1 with buffer 2 and report any discrepancies.
6. Repeats steps 1 through 5 until enough packets have been written to satisfy the length requirement of 101 blocks.

❺ A nondestructive write test with packet size of 512 bytes. The action string specifies the following sequence of operations:

1. Set the current block address to a random block number on the disk.
2. From the current block address on the disk, read a packet into buffer 1.
3. Set the current block address to the device address where it was just before the previous read operation occurred.
4. Write a packet of hex 5As from buffer 1 to the current block address.
5. Set the current block address to what it was just prior to the previous write operation.
6. From the current block address on the disk, read a packet into buffer 2.
7. Compare buffer 1 with buffer 2 and report any discrepancies.
8. Repeat the above steps until each block on the disk has been written once and read twice.

7.5.10 Galaxy Command

The galaxy command initializes soft partitions (same as lpinit).

Syntax

galaxy

Example

```
P00>>> galaxy
lp_count = 3
lp_mem_size0 = 2000 (8 GB)
CPU 0 chosen as primary CPU for partition 0
lp_mem_size1 = 4000 (16 GB)
CPU 4 chosen as primary CPU for partition 1
lp_mem_size2 = 2000 (8 GB)
CPU 12 chosen as primary CPU for partition 2
lp_shared_mem_size = 2000 (8 GB)
initializing shared memory
partitioning system

OpenVMS PALcode V1.8-1, Tru64 UNIX PALcode V1.75-1

system = QBB 0 1 2 3          + HS
 QBB 0 = CPU 0 1 2 3 + Mem 0 1 2 3 + Dir + IOP + PCA 0     3 + GP  (Hard QBB 0)
 QBB 1 = CPU 0 1 2 3 + Mem 0 1 2 3 + Dir + IOP + PCA       3 + GP  (Hard QBB 1)
 QBB 2 = CPU 0 1 2 3 + Mem 0 1 2 3 + Dir + IOP + PCA         + GP  (Hard QBB 2)
 QBB 3 = CPU 0   2 3 + Mem 0     3 + Dir + IOP + PCA   1 2 3 + GP  (Hard QBB 3)
partition 0
 CPU 0 1 2 3
 IOP 0
 private memory size is 8 GB
 shared memory size is 8 GB
micro firmware version is X5.7
shared RAM version is 1.4
hose 0 has a standard I/O module
   .
   .
   .
initializing GCT/FRU at 1f6000
initializing pke pka pkb pkd ewa dra pkc dqa eia eib eic eid eie eif eig
 eih eii pkf pkg
AlphaServer Console V5.8-1, built on May 26, 2000 at 12:15:01
P00>>>
```

7.5.11 Gct Command

The Galaxy configuration tree is built automatically when hard partitions are powered up. The gct command rebuilds the configuration tree.

Syntax

gct

Example

```
P00>>> gct
P00>>>
```

7.5.12 Grep Command

The grep command is very similar to the UNIX grep command. It searches the named files for the expression and prints any lines that match. Grep works only on ASCII files.

Syntax

grep [- {**c** | **i** | **n** | **v** |}], [-**f** *file*] [*expression*] [*file*..]

Options

-c Prints only the number of lines matched.

-i Ignores case in the search. By default, **grep** is case sensitive.

-n Prints the line numbers of the matching lines.

-v Prints all the lines that do not contain the expression.

-f *file* Takes the regular expressions from the named file, instead of the command.

Arguments

expression Specifies the target regular expression. If any metacharacters (see following) are present, the expression should be enclosed with quotes so the metacharacters will not be confused with characters to be searched for. The metacharacters are:

^ Matches the beginning of line

$ Matches the end of line

. Matches any single character

[ ] Set of characters; [ABC] matches either 'A' or 'B' or 'C'. A dash (other than first or last of the set) denotes a range of characters. For example [A-Z] matches any uppercase letter. If the first character of the set is '^', then the sense of the match is reversed. For example, [^0-9] matches any non-digit. Several characters need to be quoted with backslash (\) if they occur in a set: '\', ']' '-', and '^'.

* Repeated matching. When placed after a pattern, indicates that the pattern should match any number of times. For example, 'a[a-z][0-9]*' matches a lowercase letter followed by zero or more digits.

\+ Repeated matching. When placed after a pattern, indicates that the pattern should mach one or more times. For example, '[0-9]+' matches any non-empty sequence of digits.

? Optional matching. Indicates that the patern can match zero or one times. For example, '[a-z][0-9]?' matches lowercase letter alone or followed by a single digit.

\ Quote character. Prevents the character that follows from having special meaning.

file… Specifies the file(s) to be searched. If none are present, then the standard input is searched.

Example

```
P00>>> scm show csb | grep CPU

c0    CPU0/SROM  V6.0-7                                    ON
c1    CPU1/SROM  V6.0-7                                    ON
c2    CPU2/SROM  V6.0-7                                    ON
c3    CPU3/SROM  V6.0-7                                    ON
c4    CPU0/SROM  V6.0-7                                    ON
c5    CPU1/SROM  V6.0-7                                    ON
c6    CPU2/SROM  V6.0-7                                    ON
c7    CPU3/SROM  V6.0-7                                    ON
c8    CPU0/SROM  V6.0-7                                    ON
c9    CPU1/SROM  V6.0-7                                    ON
ca    CPU2/SROM  V6.0-7                                    ON
cb    CPU3/SROM  V6.0-7                                    ON
cc    CPU0/SROM  V6.0-7                                    ON
cd    CPU1/SROM  V6.0-7                                    ON
ce    CPU2/SROM  V6.0-7                                    ON
cf    CPU3/SROM  V6.0-7                                    ON
P00>>>
```

❶ The user types a command to pipe the SCM command **show csb** into a **grep** to search for the characters CPU.

❷ The **grep** command displays the lines of the **show csb** command that contain the letters CPU.

7.5.13 Halt

The halt command halts the specified processor or device. Equivalent to the stop command.

Syntax

halt [-**drivers** [*device_prefix*]] [*processor-number*]

Options

-drivers [*device_prefix*]	Specifies the name of the device or device class to stop. If no device prefix is specified, then all drivers are stopped.

Argument

processor-number	The soft processor number (from **show config** or the SCM's **show csb**) of the processor to stop.

Example

```
P00>>> halt
P00>>>
```

7.5.14 Help or Man Command

The help (or man) command provides basic information on the console commands.

Syntax

help [*command*]

Argument

command The command for which information is to be displayed. If omitted, help for all commands available is displayed.

Example

```
P00>>> help edit
NAME
        edit
FUNCTION
        Invoke the console BASIC-like line editor on a file.
SYNOPSIS
        edit <file>
                    [Subcommands: HELP, LIST, RENUMBER, EXIT or CTL/Z, QUIT]
                    [<nn> : Delete line number <nn>.]
                    [<nn> <text> : Add or overwrite line <nn> with <text>.]
```

7.5.15 Info Command

The info command displays registers and data structures. You can enter the command by itself or followed by a number (0-6). If you do not specify a number, a list of selections is displayed and you are prompted to enter a selection.

Syntax

info [*n*]

Argument

n A number from 0 – 6 selecting the information to be displayed:

0 Displays the SRM memory description as described in the *Alpha System Reference Manual.*

1 Reserved.

2 Dumps the Galaxy Configuration Tree (GCT) FRU table. Galaxy is a software architecture that allows multiple instances of *OpenVMS* to execute cooperatively on a single computer.

3 Reserved.

4 Displays the per CPU impure area in abbreviated form. The console uses this scratch area to save processor context.

5 Displays the per CPU impure area in full form. The console uses this scratch area to save processor context.

6 Reserved.

Example – Info 0

```
P00>>> info 0


HWRPB: 2000    MEMDSC:7340   Cluster count: 5

Cluster: 0, Usage: Console
START_PFN: 00000000  PFN_COUNT: 0000026b PFN_TESTED: 00000000
     619 pages from 0000000000000000 to 00000000004d5fff

Cluster: 1, Usage: System
START_PFN: 0000026b  PFN_COUNT: 000ffd7c PFN_TESTED: 000ffd7c
BITMAP_VA: 0000000000000000 BITMAP_PA: 00000001fffe0000
 1047932 good pages from 00000000004d6000 to 00000001fffcdfff

Cluster: 2, Usage: Console
START_PFN: 000fffe7  PFN_COUNT: 00000019 PFN_TESTED: 00000000
      25 pages from 00000001fffce000 to 00000001ffffffff

Cluster: 3, Usage: System
START_PFN: 00800000  PFN_COUNT: 000fffe4 PFN_TESTED: 000fffe4
BITMAP_VA: 0000000000000000 BITMAP_PA: 00000011fffc8000
 1048548 good pages from 0000001000000000 to 00000011fffc7fff

Cluster: 4, Usage: Console
START_PFN: 008fffe4  PFN_COUNT: 0000001c PFN_TESTED: 00000000
      28 pages from 00000011fffc8000 to 00000011ffffffff
P00>>>
```

Example – Info 2

```
P00>>> info 2
GCT_ROOT_NODE

GCT_NODE:                1fc000
type                       1
subtype                    0
hd_extension               0
size                       30000
rev_major                  6
rev_minor                  0
id                         0000000000000000
node_flags                 0
saved_owner                0
affinity                   0
parent                     0
child                      2c0
fw_usage                   0
Root->lock                 ffffffff
Root->transient_level      2b
Root->current_level        2b
Root->console_req          200000
Root->min_alloc            800000
Root->min_align            800000
Root->base_alloc           4000000
Root->base_align           4000000
Root->max_phys_addr        7fffffffff
Root->mem_size             c0000000
Root->platform_type        50000500000023
Root->platform_name        0000000000000280
Root->primary_instance     0
Root->first_free           b700
Root->high_limit           2fcc0
Root->lookaside            0
Root->available            21780
Root->max_partition        8
Root->partitions           0000000000000180
Root->communities          00000000000001c0
Root->bindings             0000000000000200
Root->max_plat_partition   8
Root->max_desc             8
Root->galaxy_id            1fc128
Root->root_flags           3

dump depth view ? (Y/<N>) n
dump each node ? (Y/<N>) n
dump binary ? (Y/<N>) n
show flags? ( Y/<N>) n
```

Example – Info 4

```
P00>>> info 4
      cpu00   cpu03
per_cpu impure area  00008000 00008c00
cns$flag             00000001 00000001 : 0000
cns$flag+4           00000000 00000000 : 0004
cns$hlt              00000000 00000000 : 0008
cns$hlt+4            00000000 00000000 : 000c
cns$mchkflag         00000158 00000158 : 0210
   .
   .
   .
cns$dc_ctl           000000c3 000000c3 : 0330
cns$dc_ctl+4         00000000 00000000 : 0334
cns$dc_stat          00000000 00000000 : 0338
cns$dc_stat+4        00000000 00000000 : 033c
cns$write_many       73fff8c1 73fff8c1 : 0340
cns$write_many+4     00000003 00000003 : 0344
```

Example – Info 5

```
P00>>> info 5

      cpu00   cpu03
per_cpu impure area  00008000 00008c00
cns$flag             00000001 00000001 : 0000
cns$flag+4           00000000 00000000 : 0004
cns$hlt              00000000 00000000 : 0008
cns$hlt+4            00000000 00000000 : 000c
cns$gpr[0]           00020000 00020000 : 0010
cns$gpr[0]+4         00000000 00000000 : 0014

   .
   .
   .
cns$write_many       73fff8c1 73fff8c1 : 0340
cns$write_many+4     00000003 00000003 : 0344
P00>>>
```

7.5.16 Init Command

The init command resets the SRM console firmware, incorporating any changes made to environment variables during the foregoing console session, and reinitializes the hardware.

Syntax

init

Example

```
P00>>> init

OpenVMS PALcode V1.8-1, Tru64 UNIX PALcode V1.75-1

system = QBB 0 1             + HS                              (Hard Partition 0)
 QBB 0 = CPU 0 1 2 3 + Mem 0   2   + Dir + IOP + PCA 0 1     + GP  (Hard QBB 0)
 QBB 1 = CPU 0 1 2 3 + Mem 0 1     + Dir + IOP + PCA 0 1     + GP  (Hard QBB 1)
micro firmware version is X5.7
shared RAM version is 1.4
hose 0 has a standard I/O module
starting console on CPU 0
initialized idle PCB
initializing semaphores
initializing heap
initial heap 2c0c0
memory low limit = 1f6000
heap = 2c0c0, 1ffc0
initializing driver structures
initializing idle process PID
initializing file system
initializing timer data structures
lowering IPL
CPU 0 speed is 731 MHz
create dead_eater
   .
   .
   .
initializing GCT/FRU at 1f6000
initializing pka pkb pkc pkd pke pkf pkg pkh pki pkj pkk fwa fwb dqa dqb
 eia eib pga pga0.0.0.1.10 - Nvram read failed.

Testing the System
Testing the Disks (read only)
Reservation conflict for dkc0.0.0.5.1
   .
   .
   .
Testing the Network
AlphaServer Console V5.801, built on May 26, 2000 at 12:15:01
P00>>>
```

7.5.17 Kill Command

The kill command allows you to kill a process that is running on the system. This is useful for stopping exercisers that may be running. First, use the show_status or ps command to get the process ID. Then use the kill command specifying that process ID.

Syntax

kill *process_id*

Example

```
P00>>> show_status      ❶
 ID       Program      Device       Pass  Hard/Soft Bytes Written  Bytes Read
-------- ------------ ------------ ------ --------- ------------- -------------
00000001         idle system            0    0    0             0             0
00000123      memtest memory            2    0    0     520093696     520093696
P00>>> kill 123    ❷
P00>>> show_status      ❸
 ID       Program      Device       Pass  Hard/Soft Bytes Written  Bytes Read
-------- ------------ ------------ ------ --------- ------------- -------------
00000001         idle system            0    0    0             0             0
P00>>>
```

❶ The user types the **show_status** command to show the status of any background processes. Process 123 is shown as a memory exerciser.

❷ The user issues the **kill 123** command to terminate the execution of the memory exerciser.

❸ This **show_status** command confirms that the memory exerciser is no longer running.

NOTE: *Issue an **init** command after running tests and/or exercisers before booting to ensure that registers are properly initialized.*

7.5.18 Kill_diags Command

The kill_diags command stops all console-based diagnostic processes running on the system.

Syntax

kill_diags

Example

```
P00>>> show_status    ❶
 ID       Program      Device       Pass  Hard/Soft Bytes Written  Bytes Read
-------- ------------ ------------ ------ --------- ------------- -------------
00000001         idle system            0    0    0             0             0
0000011d      memtest memory            2    0    0     520093696     520093696
00000123      memtest memory            2    0    0     520093696     520093696
00000162      memtest memory            2    0    0     520093696     520093696
P00>>> kill_diags     ❷
P00>>> show_status    ❸
 ID       Program      Device       Pass  Hard/Soft Bytes Written  Bytes Read
-------- ------------ ------------ ------ --------- ------------- -------------
00000001         idle system            0    0    0             0             0
P00>>>
```

❶ The **show_status** command shows three memory exerciser processes running on the system.

❷ The **kill_diags** command terminates all diagnostic processes.

❸ This **show_status** command shows that the memory exercisers are no longer running.

NOTE: *Issue an **init** command after running tests and/or exercisers before booting to ensure that registers are properly initialized.*

7.5.19 Lpinit Command

The lpinit command initializes soft partitions (same as galaxy).

Syntax

lpinit

Example

```
P00>>> sh lp*  ❶
lp_count                3
lp_cpu_mask0            000f
lp_cpu_mask1            0ff0
lp_cpu_mask2            f000
lp_cpu_mask3            0
lp_cpu_mask4            0
lp_cpu_mask5            0
lp_cpu_mask6            0
lp_cpu_mask7            0
lp_error_target         0
lp_io_mask0             1
lp_io_mask1             6
lp_io_mask2             8
lp_io_mask3             0
lp_io_mask4             0
lp_io_mask5             0
lp_io_mask6             0
lp_io_mask7             0
lp_mem_size0            0 = 8 GB
lp_mem_size1            1 = 8 GB, 2 = 8 GB
lp_mem_size2            3 = 8 GB
lp_mem_size3            0
lp_mem_size4            0
lp_mem_size5            0
lp_mem_size6            0
lp_mem_size7            0
lp_shared_mem_size      8 GB
P00>>> lpinit  ❷
lp_count = 3
lp_mem_size0 = 2000 (8 GB)
CPU 0 chosen as primary CPU for partition 0
lp_mem_size1 = 4000 (16 GB)
CPU 4 chosen as primary CPU for partition 1
lp_mem_size2 = 2000 (8 GB)
CPU 12 chosen as primary CPU for partition 2
lp_shared_mem_size = 2000 (8 GB)
initializing shared memory
partitioning system  ❸


OpenVMS PALcode V1.8-1, Tru64 UNIX PALcode V1.75-1

system = QBB 0 1 2 3          + HS
 QBB 0 = CPU 0 1 2 3 + Mem 0 1 2 3 + Dir + IOP + PCA 0     3 + GP  (Hard QBB 0)
```

```
 QBB 1 = CPU 0 1 2 3 + Mem 0 1 2 3 + Dir + IOP + PCA        3 + GP  (Hard QBB 1)
 QBB 2 = CPU 0 1 2 3 + Mem 0 1 2 3 + Dir + IOP + PCA          + GP  (Hard QBB 2)
 QBB 3 = CPU 0   2 3 + Mem 0     3 + Dir + IOP + PCA   1 2 3 + GP  (Hard QBB 3)
partition 0
 CPU 0 1 2 3
 IOP 0
 private memory size is 8 GB
 shared memory size is 8 GB
micro firmware version is X5.7
shared RAM version is 1.4
hose 0 has a standard I/O module
   .
   .
   .
initializing GCT/FRU at 1f6000
initializing pke pka pkb pkd ewa dra pkc dqa eia eib eic eid eie eif eig
 eih eii pkf pkg
AlphaServer Console V5.8-1, built on May 26, 2000 at 12:15:01
P00>>>
```

❶ The **show lp*** command shows the soft partition definition for the system. At this point, the system has no soft partitions.

❷ The **lpinit** command initializes the soft partitions for the system.

❸ At this point, the display is only for partition 0. The other partitions are initializing in their own console windows provided through the SMC terminal emulator.

7.5.20 Ls Command

The ls command lists files in the system. Files include script files, diagnostics, and executable shell commands.

Syntax

ls [**-l**] [*filename…*]

Option

-l Specifies that the list is to be in long format; listing other information besides the file name.

Argument

filename… Specifies the file(s) to be listed.

Example

```
P00>>> ls                                                    ❶
alloc           b               boot            bootp           bpt
break           buf             buildfru        c               call
    .
    .
    .
srflashload     start           stop            sym             sys_exer
test            tftp            toy             true            tt
tta0            tta1            uptime          vmem            vmem_render
wc              wf              wwidmgr
P00>>> ls *pr                                                ❷
fpr             gpr             ipr
P00>>> ls -l sys_exer                                        ❸
rwx-  rd                3705/3705           1c0cd4      0    sys_exer
P00>>>
```

❶ The **ls** command lists the names of all files in the system.

❷ Wildcarding is allowed with the **ls** command. Here, all files ending in the characters "pr" are listed.

❸ This example shows the long form of the listing. The information is in six fields: (1) access: r=read, w=write, x=execute, b=binary, (2) driver name, (3) actual size/allocated size in bytes, (4) location in memory (if an rd file), (5) reference count (how many times this file is currently open, and (6) file name.

7.5.21 Memexer Command

The memexer command allows you to test the memory on your system, using a Gray code memory exerciser. The program randomly allocates and tests blocks of memory two times the size of the B-cache using all available memory. The memexer command automatically does testing in background mode without using the &.

Syntax

memexer [*n*]

Argument

n Specifies the number of memory test processes to start. The default is l.

Example

```
P00>>> memexer 3    ❶
memtest -bs 1000000 -rb -p 0 &
memtest -bs 1000000 -rb -p 0 &
memtest -bs 1000000 -rb -p 0 &
P00>>> show_status    ❷
 ID       Program      Device       Pass  Hard/Soft Bytes Written  Bytes Read
-------- ------------ ------------ ------ --------- ------------- -------------
00000001         idle system            0    0    0             0             0
0000011d      memtest memory            2    0    0     520093696     520093696
00000123      memtest memory            2    0    0     520093696     520093696
00000162      memtest memory            2    0    0     520093696     520093696
P00>>> kill_diags    ❸
P00>>> show_status    ❹
 ID       Program      Device       Pass  Hard/Soft Bytes Written  Bytes Read
-------- ------------ ------------ ------ --------- ------------- -------------
00000001         idle system            0    0    0             0             0
P00>>>
```

❶ Starts three memory test processes, run in the background.

❷ The **show_status** command shows the status of the executing exercisers.

❸ The **kill_diags** command terminates the processes.

❹ The **show_status** command shows that the processes are no longer running.

7.5.22 Memexer_mp

The memexer_mp command invokes pairs of Gray code memory exercisers on a multiprocessor system. The exercisers are run in the background. This command exercises the ability of CPUs to share data and remain coherent.

Syntax

memexer_mp

Description

The **memexer_mp** command starts a copy of **memexer** on each CPU, testing a different longword in a cache block. Since there are 16 longwords in a cache block, at most 16 **memexer**s are started. The first **memexer** runs on CPUs 0 and 16 (if they exist), the second, on CPUs 1 and 17 (if they exist), the third, on CPUs 2 and 18 (if they exist), and so on.

NOTE*: Do not call memexer_mp multiple times, as you will get a stream of data compare errors. Two copies of each exerciser will be touching the same areas in memory, but they are not synchronized.*

Example

```
P00>>> memexer_mp
memtest -sa 400000 -t 1 -l 40000 -i 10 -p 0 -z &
memtest -sa 400004 -t 1 -l 40000 -i 10 -p 0 -z &
memtest -sa 400008 -t 1 -l 40000 -i 10 -p 0 -z &
memtest -sa 40000C -t 1 -l 40000 -i 10 -p 0 -z &
P00>>> show_status
 ID       Program      Device       Pass  Hard/Soft Bytes Written  Bytes Read
-------- ------------ ------------ ------ --------- ------------- -------------
00000001         idle system            0    0    0             0             0
00000984      memtest memory          901    0    0     235929600     235929600
00000998      memtest memory          895    0    0     234356736     234356736
000009ac      memtest memory          886    0    0     231997440     231997440
000009f9      memtest memory          869    0    0     227540992     227540992
P00>>> ps | grep memtest
000009f9 002531a0 2          0 00080008 3      memtest waiting on tqe 682b0
000009ac 00286d80 2          2 00040004 2      memtest waiting on tqe 682b0
00000998 0027f7c0 2          3 00020002 1      memtest ready
00000984 00278200 2          3 00010001 0      memtest waiting on tqe 682b0
```

7.5.23 Migrate Command

The migrate command switches one or all CPUs from one soft partition to another.

Syntax

migrate [**-cpu** *cpu_id*, **-all**] -**partition** *partition_number*

Options

-cpu *cpu_id*	Specifies that one CPU identified by the soft CPU number *cpu_id* (from the **show config** command) is to be transferred to the specified soft partition.
-all	Specifies that all CPUs in this hard partition are to be transferred to the specified soft partition.
-partition *partition_number*	Specifies the soft partition to which the CPU(s) are to be transferred.

Example

```
P00>>> migrate -cpu 0 -partition 2    ❶
```

❶ Transfer CPU 0 to soft partition 2.

7.5.24 More Command

The more command displays output one screen at a time.

Syntax

more [–*n*] [*file…*]

Option

-*n* The number of lines to be displayed before waiting for a prompt. The default is 23. At the prompt, you can type a space for the next series of lines, press Enter to display the next line, or Q to quit the **more** command.

Argument

file… Specifies the file(s) to be displayed.

Example

```
P00>>> help  | more -10      ❶
NAME
        help
FUNCTION
        Display information about console commands.
SYNOPSIS
        help [<command>...]
                  Command synopsis conventions:
                  <item> Implies a placeholder for user specified item.
                  <item>... Implies an item or list of items.
                  [] Implies optional keyword or item.

--More--  [SPACE –next page, ENTER – next line, Q – quit)   ❷
MORE> q   ❸
P00>>>    ❹
```

❶ The user pipes the **help** command into the **more** command requesting a screen size of 10 lines. Ten lines of the **help** output are displayed by **more.**

❷ The **more** command displays a line describing what commands are available at the prompt.

❸ The prompt is displayed and the user types **q** to quit.

❹ Control returns to the SRM prompt. (This also occurs at the end of file.)

7.5.25 Nettest Command

The nettest command tests the network ports by running maintenance operations protocol (MOP) loopback tests. Many environment variables can be set to customize nettest. These may be set from the SRM console before nettest is started.

Syntax

nettest [**-f** *file*] [**-mode** *port_mode*] [**-p** *pass_count*] [**-sv** *mop_version*] [**-to** *loop_time*] [**-w** *wait_time*] [*port_name*]

Options

-f *file*
Specifies the file containing the list of network station addresses to loop messages to. The default file name is **lp_nodes_ew***__n__ for Tulip ports. The default file name for Intel Ethernet controller drivers is **lp_nodes_ei***__n__. In both cases, * is a letter of the alphabet and ***n*** is the controller number.

-mode *port_mode*
Specifies the mode to set the port adapter (TGEG). The default is **ex** (external loopback), the most likely to be useful in general network testing.

df	Default, use environment variable values
ex	External loopback
in	Internal loopback
nm	Normal mode
nf	Normal filter
pr	Promiscuous
mc	Multicast
ip	Internal loopback and promiscuous
fc	Force collisions
nofc	Do not force collisions
nc	Do not change mode

-p *pass_count*
Specifies the number of passes for the diagnostic. If 0, then run forever. The default is 1. Each pass will send the number of loop messages as set by the environment variable **ewa*_loop_count** (Tulip driver) or **ela*_loop_count** (Intel Ethernet controller driver). Note

that this is the number of passes for the diagnostic. Each pass will send the number of loop messages as set by the environment variable ***ew*n_loop_count*** or ***ei*n_loop_count.***

-sv *mop_version* Specifies the MOP (maintenance operations protocol) version to use. If 3, then MOP V3 (DECnet Phase IV) packet format is used. If 4, then MOP V4 (DECnet Phase V IEEE 802.3) format is used.

-to *loop_time* Specifies the time, in seconds, allowed for the loop messages to be returned. The default is 2 seconds.

-w *wait_time* Specifies the time, in seconds, to wait between passes of the test. The default is 0 (no delay). The network device can be very CPU intensive. This option will allow other processes to run.

Related Environment Variables

ew*n_loop_count or **ei*n_loop_count** Specifies the number, in hex, of loop requests to send. The default is 0x3E8 (l000 decimal) loop packets.

ew*n_loop_inc or **ei*n_loop_inc** Specifies the number of bytes (in hex) to increase the message size by in successive messages. The default is 0xA (10 decimal) bytes.

ew*n_loop_patt or **ei*n_loop_patt** Specifies the loop messages. The following are legitimate values:

0	All zeros
1	All ones
2	All fives
3	All 0xAs
4	Incrementing data
ffffffff	All patterns

loop_size Specifies the size (in hex) of the loop message, in bytes. The default packet size is 0x2E.

Argument

port_name The Ethernet port on which to run the test.

7.5.26 Nvram Script

The system comes with a script (set of commands) named "nvram" that is stored in EEROM. Nvram is a power-up script that is always invoked during the power-up sequence. Use the SRM edit command to create or alter the nvram script.

Examples

```
P00>>> edit nvram
editing 'nvram'
0 bytes read in
*10   set mopv3_boot 1
*exit
17 bytes written out to nvram
P00>>> edit nvram
editing 'nvram'
20 bytes read in
*10
*exit
0 bytes written out to nvram
P00>>>
```

Description

You can create an nvram script with any commands you want the system to execute at power-up. You create and edit the nvram script using the SRM **edit** command. (See Section 7.5.8 for a description of the **edit** command.)

In the examples above, an environment variable called **mopv3_boot** is created and set to 1 on each power-up. By default, MOP boots send four MOP V4 requests before defaulting to MOP V3. This user-created environment variable forces the SRM console to bypass MOP V4 requests. This speeds up MOP booting on networks with MOP V3 software.

CAUTION: *An inappropriate command can disable the system. For example, the **init** command will cause the system to go into an endless loop. To correct this error, press the Halt button or issue the SCM **halt in** command, then power up or reset the system. When the P00>>> prompt is displayed, edit the nvram script to remove the illegal command.*

7.5.27 Power Command

The power command supplies or removes power from a CPU, I/O riser port, or PCI box. To prevent catastrophic errors, shut down the operating system before using these commands.

Syntax

power {**on | off** } {*none* |**-cpu** *n* | **-ior** *n* | **-pci** *id* }

Options

none **For hard partitions:** On **power off**, the command stops all CPUs (except the primary) in the hard partition from executing instructions, stops and disables all interrupts; and disables all DMA. Power is then removed from all QBBs in that hard partition. PCI boxes are not powered off. (Note that powering the partition on again is done with the SCM **power on** command.)

-cpu *n* Supplies or removes power from the specified CPU, where *n* is the soft CPU number (see **show config**) for the system or hard partition).

-ior *n* Supplies or removes power from the specified I/O riser port, where *n* is the IOR number (see **show config)** for the system or hard partition.

WARNING: There are two IOR ports on a single IOR module. You must power off both IOR ports on the module before the IOR module is safe for removal.

-pci *id* Supplies power to or removes power from the specified PCI box, where *id* is the setting for that PCI box's node ID switch.

Example

```
P00>>> power on -cpu 3
Powering on CPU3 (CPU3 in QBB 0)
System Event QBB summary: 01
System Event reported by QBB 0
QBB 0 System Event Summary: 0000000020000000
Restarting console on CPU3
P00>>>
System Event QBB Summary: 01
System Event reported by QB 0
QBB 0 System Event Summary: 0000000008000000
P00>>>
```

7.5.28 Ps Command

The ps command allows you to display information about process status and statistics. This information is useful when you are running diagnostic processes. The most useful fields are process ID, CPU number, program name, and process state.

Syntax

ps

Example

```
P00>>> memexer_mp
memtest -sa 400000 -t 1 -l 40000 -i 10 -p 0 -z &
memtest -sa 400004 -t 1 -l 40000 -i 10 -p 0 -z &
memtest -sa 400008 -t 1 -l 40000 -i 10 -p 0 -z &
memtest -sa 40000C -t 1 -l 40000 -i 10 -p 0 -z &
P00>>> ps
```

```
  ❶                                      ❷              ❸
ID          PCB     Pri CPU Time Affinity CPU   Program  State
--------- --------- --- -------- -------- --- ---------- --------------------
000009f9 002531a0 2           0 00080008 3      memtest waiting on tqe 682b0
000009ac 00286d80 2           2 00040004 2      memtest waiting on tqe 682b0
00000998 0027f7c0 2           3 00020002 1      memtest ready
00000984 00278200 2           3 00010001 0      memtest waiting on tqe 682b0
P00>>>
```

❶ The process ID is needed for **kill** commands to terminate a specific process.

❷ Indicates the number of the CPU being tested.

❸ State of the executing test.

7.5.29 Reset command

The reset command is used to reset a hard partition.

Syntax

reset

Description

On the hard partition in which the command was entered, the **reset** command stops all CPUs (except the primary) from executing instructions, stops and disables all interrupts, and disables all DMA.

Example

```
P00>>> reset
```

7.5.30 Rm Command

The rm command removes the named file(s) from the file system.

Syntax

rm *file…*

Argument

file.. The name of the file(s) to be removed.

Example

```
P00>>> ls
sample
datafile
P00>>> rm sample
P00>>> ls
datafile
P00>>>
```

7.5.31 Run BIOS

Certain PCI options contain a BIOS extension ROM which provides extended services for the option. The run bios command executes code on the BIOS extension ROM that invokes the utility that provides the service.

Syntax

run bios *device-name*

Arguments

device-name PCI device designator. (See the **show bios** command.)

Description

The **run bios** command invokes code on the BIOS extension ROM on a PCI option.

A BIOS extension ROM resides on a PCI option and provides one or more extended services for that option, for example, a RAID configuration utility or a firmware update utility. Once invoked, the BIOS ROM provides a graphical menu-driven interface to select the service.

After you issue the **run bios** command, you type a control sequence specific to the PCI option to bring up the menu interface. See the documentation supplied with the option for information on running the utility.

At present, a VGA monitor connected to the system is required for running the utility. Or, you can use a PC. A future release of the SRM console will support running the utility on the serial line.

- If you use a VGA monitor, set the **console** environment variable to **graphics** and issue the **init** command before issuing **run bios.**
- If you use a PC, install the option card into a PCI slot on the PC and then type the control sequence.

Example

```
P00>>> set console graphics  ❶
P00>>> init
P00>>> show bios    ❷
Resetting all I/O buses
```

```
pga0.0.0.5.25 – LINK_DOWN timeout
pza0.4.0.3.0 – I2O
vga0.0.0.2.0 – ELSA Gloria Synergy
eia0.0.0.2.2 – DE600 –AA
pkb0.7.0.5.1 –Qlogic ISP10x0
pkd0.7.0.3.28 – QLOGIC ISP10x0
P00>>> run bios pza0    ❸
    .
    .
    .
Hit (CTRL + D) for DPT Setup, Waiting for devices
Resetting all I/O buses
```

❶ The **console** environment variable is set to **graphics** and the system is initialized to put the graphics setting into effect.

❷ The **show bios** command displays the names of all devices on the system (or in the hard partition) that have BIOS extension ROMs.

❸ The **run bios** command executes code on the BIOS extension ROM on device pza0. The operator is prompted for a control sequence that allows entry into the utility.

7.5.32 SCM Command

The scm command allows the user to execute an SCM command from the SRM console prompt.

Syntax

scm *scm_command*

Argument

scm_command Any of the SCM commands described in Chapter 6.

Example

```
P00>>> scm show status    ❶

System Management Settings    ❷

SCM escape sequence           :    [ESC][ESC]scm

Local Baud/flow control       :    57600 / soft
COM1 Baud/flow control        :    57600 / hard
Modem Baud/flow control       :    57600 / hard
COM1 mode                     :    Pass-through
OCP power switch              :    Off
OCP halt                      :    Deasserted
OCP secure                    :    Non-Secure
Remote access                 :    Disabled
Remote user                   :    Not Connected
Alerts                        :    Disabled
Modem password                :    wffirmware
Modem init string             :    ate0v0&c1s0=2
Modem dial string             :    atdt915085551212
Modem alert string            :    ,,,,,,,5085551212#;
Alert pending                 :    No
Most recent alert             :
P00>>>
```

❶ The user issues the **scm** command to display a **show status** from the SRM command prompt.

❷ The **show status** command displays. Control returns to the SRM prompt.

7.5.33 Set envar Command

The set envar command allows you to set or modify the value of an environment variable.

Syntax

se[t] *envar* [*value*]

Argument

envar [*value*] Environment variables and their values are given in Table 7–5.

Table 7–5 Environment Variables

Variable	Attribute	Function
auto_action	Nonvolatile	Specifies the action the console will take following an error halt or power-up. Values are: **restart** - Automatically restart the system. If restart fails, boot the operating system. **boot** - Automatically boot the operating system. Systems will use as the default device that defined by manufacturing (for factory-installed software), or a default boot device selected by setting the **bootdef_dev** environment variable. **halt** (default) - Enter SRM console mode.
bootdef_dev	Nonvolatile	Defines the default device or device list from which booting is attempted when no device name is specified by the **boot** command.
boot_file	Nonvolatile	Defines the default file name used for the primary bootstrap when no file name is specified by the **boot** command, if appropriate.
boot_osflags	Nonvolatile	Defines additional parameters to be passed to the system software during booting if none are specified by the **boot** command with the **-flags** specifier.

Table 7-5 Environment Variables (Continued)

Variable	Attribute	Function
com1_baud	Nonvolatile	Sets the default baud rate of the COM1 serial interface. The default is 9600. Supported values: 1800, 2000, 2400, 3600, 4800, 7200, 9600, 19200, 38400, 57600.
com2_baud	Nonvolatile	Sets the default baud rate of the COM2 port. Select a baud rate to match that of the device connected to the COM2 port.
com1_flow **com2_flow**	Nonvolatile	The **com1_flow** and **com2_flow** environment variables indicate the flow control on the serial ports. Defined values are: **none** No data flows in or out of the serial ports. Use this setting for devices that do not recognize XON/XOFF or that would be confused by these signals. **software** Use XON/XOFF (default). This is the setting for a standard serial terminal. **hardware** Use modem signals CTS/RTS. Use this setting if you are connecting a modem to a serial port.
com1_mode	Nonvolatile	Specifies the COM1 data flow paths so that data either flows through the SCM or bypasses it. See Section 3.6.
com1_modem **com2_modem**	Nonvolatile	Used to tell the operating system whether a modem is present on the COM1 or COM2 ports, respectively. Values are: **on** Modem is present. **off** Modem is not present (default).
console	Nonvolatile	Defines the type of console device. **serial** A serial console terminal **graphics** A graphics console device.

Continued on next page

Table 7-5 Environment Variables (Continued)

Variable	Attribute	Function
cpu	Nonvolatile	Selects the current boot processor.
cpu_enabled	Nonvolatile	A bitmask indicating which processors are enabled to run (leave console mode). Default is **0xffffffff**.
d_harderr	Volatile	Determines action taken following a hard error. Values are **halt** (default) and **continue.** Applies only when using **test**.
d_report	Volatile	Determines level of information provided by the diagnostic reports. Values are **summary** and **full** (default). Applies only when using **test**.
d_softerr	Volatile	Determines action taken following a soft error. Values are **continue** (default) and **halt**. Applies only when using **test**.
d_trace	Nonvolatile	Specifies whether or not to display **test** trace messages. **off** (default) - Disables trace messages **on** - Enables trace messages
dump_dev	Nonvolatile	Device to which dump file is written if the system crashes, if supported by the operating system.
enable_audit	Nonvolatile	If set to **on** (default), enables the generation of audit trail messages. If set to **off**, audit trail messages are suppressed. Console initialization sets this to **on**.
ew*0_loop_count	Nonvolatile	Specifies number of times message is looped for a **test** command exercising a PCI network adapter.
ew*0_loop_inc	Nonvolatile	Specifies the amount the message size is increased from message to message.

Table 7-5 Environment Variables (Continued)

Variable	Attribute	Function
ew*0_loop_patt	Nonvolatile	Specifies data pattern used for loopback. 0xffffffff All the patterns 1 All zeros 2 All ones 3 All As 4 Incrementing 5 Decrementing
ew*0_loop_size	Nonvolatile	Size of loop data used.
ew*0_lp_msg_node	Nonvolatile	Number of messages originally sent to each node.
ew*0__mode	Nonvolatile	Value for the Ethernet port node when it is started. Allowed values are: Auto-sensing BNC AUI FastFD (full duplex) Twisted-pair Auto-negotiate Full duplex, twisted pair
lp_count	Nonvolatile	Number of soft partition (0 – 8)
lp_cpu_mask*n*	Nonvolatile	Bit mask identifying CPUs in soft partition *n*.
lp_error_target	Nonvolatile	Number identifying soft partition to which errors are to be logged
lp_io_mask*n*	Nonvolatile	Bit mask identifying soft QBB numbers defining I/O for soft partition *n*.
lp_mem_size*n*	Nonvolatile	Memory size of soft partition *n*.
lp_shared_mem_ size	Nonvolatile	Size of memory shared between all soft partitions.
os_type	Nonvolatile	Used to store operating system type. Values are **vms**, **openvms**, **osf**, and **unix**.

7.5.34 Show BIOS

Use the show bios command to display the devices on the system that have BIOS extension ROMs.

Syntax

show bios

Description

The **show bios** command displays the names of all devices on the system (or in the hard partition) that have BIOS extension ROMs. It is used in conjunction with the **run bios** command.

A BIOS extension ROM resides on a PCI option and provides one or more extended services for that option. The service depends on the code on the extension ROM — for example, a RAID configuration utility or a firmware update utility. Once invoked, the BIOS ROM provides a graphical menu-driven interface from which to select the service.

Example

```
P00>>> show bios    ❷
Resetting all I/O buses
pga0.0.0.5.25 - LINK_DOWN timeout
pza0.4.0.3.0 - I2O
vga0.0.0.2.0 - ELSA Gloria Synergy
eia0.0.0.2.2 - DE600 -AA
pkb0.7.0.5.1 -Qlogic ISP10x0
pkd0.7.0.3.28 - QLOGIC ISP10x0
```

7.5.35 Show Configuration

The show configuration command displays the configuration seen at the last system initialization.

Syntax

sh[ow] c[onfiguration]

Example

```
P00>>> show config
                        Compaq Computer Corporation
                        Compaq AlphaServer GS80 6/627

SRM Console     V5.8-1, built on May 26, 2000 at 12:15:01                    ❶
PALcode         OpenVMS PALcode V1.81-1, Tru64 UNIX PALcode V1.75-1

QBB 0           Hard QBB 0                                                   ❷
 Quad Switch                                QSA rev 4, QSD revs 0/0/0/0
 Duplicate Tag              Up To 4 MB Caches   DTag revs 1/1/1/1
 Processor 0    CPU 0       4 MB Cache          EV67 pass 2.4, 731 MHz
 Processor 1    CPU 1       4 MB Cache          EV67 pass 2.4, 731 MHz
 Processor 2    CPU 2       4 MB Cache          EV67 pass 2.4, 731 MHz
 Memory 0                                   MPA rev 1, MPD revs 1/1
 Memory 1                                   MPA rev 1, MPD revs 1/1
 Memory 2                                   MPA rev 1, MPD revs 1/1
 Memory 3                                   MPA rev 1, MPD revs 1/1
 Directory                                  DMA rev 1, DMD rev 0
 IOP                                        IOA rev 2, IOD revs 0/0
  Local Link 0    IOR 0                     NE ML rev 2
  Remote Link 0                             FE ML rev 2
  I/O Port 0                                PCA rev 3                       ❸
   PCI Box 0     Riser 0     Right Side          Backplane rev 3
    PCI Bus 0    Hose 0      64 Bit, 33 MHz      PCI rev 2.1 compliant
    PCI Bus 1    Hose 1      64 Bit, 33 MHz      PCI rev 2.1 compliant
  Local Link 1    IOR1                      NE ML rev 2
  Remote Link 1                             FE ML rev 2
  I/O Port 1                                PCA rev 3                       ❸
   PCI Box 0     Riser 1     Left Side           Backplane rev 3
    PCI Bus 0    Hose 2      64 Bit, 33 MHz      PCI rev 2.1 compliant
    PCI Bus 1    Hose 3      64 Bit, 33 MHz      PCI rev 2.1 compliant
  Local Link 2     IOR2                      NE ML rev 2
  Remote Link 2                             FE ML rev 2
  I/O Port 2                                PCA rev 3                       ❹
   PCI Box 3     Riser 0     Right Side          Backplane rev 3
    PCI Bus 0    Hose 4      64 Bit, 33 MHz      PCI rev 2.1 compliant
    PCI Bus 1    Hose 5      64 Bit, 33 MHz      PCI rev 2.1 compliant
  Local Link 3      IOR3                      NE ML rev 2
  Remote Link 3                             FE ML rev 2
  I/O Port 3                                PCA rev 3                       ❹
   PCI Box 3     Riser 1     Left Side           Backplane rev 3
    PCI Bus 0    Hose 6      64 Bit, 33 MHz      PCI rev 2.1 compliant
    PCI Bus 1    Hose 7      64 Bit, 33 MHz      PCI rev 2.1 compliant
 Global Port                                GPA rev 0, GPD revs 0/0
```

❶ **Firmware**. Version numbers of the SRM console, *OpenVMS* PALcode, and *Tru64 UNIX* PALcode.

❷ **QBB 0**. Components listed include the quad switch and the following modules: CPUs, memory modules, directory module, IOP module, and global port. Chip revision numbers are also listed. Component information for each QBB in the system is displayed.

❸ **PCI I/O information, PCI Box 0.** In this example, QBB0 is connected to PCI Box 0 and PCI box 3 (see ❹).

QBB0 I/O port 0 is linked to remote I/O riser 0 located on the right side of PCI box 0. Logical hose numbers are 0 and 1.

QBB0 I/O port 1 is linked to remote I/O riser 1 located on the left side of PCI box 0. Logical hose numbers are 2 and 3.

❹ **PCI I/O information, PCI Box 3.** QBB0 is also connected to PCI box 3.

QBB0 I/O port 0 is linked to remote I/O riser 0 located on the right side of PCI box 3. Logical hose numbers are 4 and 5.

QBB0 I/O port 1 is linked to remote I/O riser 1 located on the left side of PCI box 3. Logical hose numbers are 6 and 7.

Continued on next page

Show Config Example (Continued)

```
QBB 1           Hard QBB 1                                              ❺
 Quad Switch                                 QSA rev 4, QSD revs 0/0/0/0
 Duplicate Tag              Up To 4 MB Caches   DTag revs 1/1/1/1
 Processor 1    CPU 5       4 MB Cache          EV67 pass 2.4, 731 MHz
 Processor 2    CPU 6       4 MB Cache          EV67 pass 2.4, 731 MHz
 Memory 0                                    MPA rev 1, MPD revs 1/1
 Memory 3                                    MPA rev 1, MPD revs 1/1
 Directory                                   DMA rev 1, DMD rev 1
 IOP                                         IOA rev 2, IOD revs 0/0
  Local Link 0     IOR 4                     NE ML rev 2
  Remote Link 0                              FE ML rev 2
  I/O Port 0                                 PCA rev 3
   PCI Box 1    Riser 0     Right Side          Backplane rev 3         ❻
    PCI Bus 0   Hose 8      64 Bit, 33 MHz      PCI rev 2.1 compliant
    PCI Bus 1   Hose 9      64 Bit, 33 MHz      PCI rev 2.1 compliant
  Local Link 1     IOR5                      NE ML rev 2
  Remote Link 1                              FE ML rev 2
  I/O Port 1                                 PCA rev 3
   PCI Box 1    Riser 1     Left Side           Backplane rev 3
    PCI Bus 0   Hose 10     64 Bit, 33 MHz      PCI rev 2.1 compliant
    PCI Bus 1   Hose 11     64 Bit, 33 MHz      PCI rev 2.1 compliant
  Local Link 2     IOR6                      NE ML rev 2
  Local Link 3     IOR7                      NE ML rev 2
 Global Port                                 GPA rev 0, GPD revs 0/0

 QBB     Size    Interleave                  System Memory 12 GB         ❼
  0       8 GB     32-Way
  1       4 GB     16-Way

 Hose   IOP   PCI Box                        Standard I/O Modules        ❽
   0     0       0          Primary          Acer Labs M1543C rev A1-E
   4     0       3                           Acer Labs M1543C rev A1-E
   8     1       1                           Acer Labs M1543C rev A1-E

System Memory 12 GB

 Board   Set   Array    Size      Address       QBB 0 Memory              ❾
   0      0      0       1 GB   00000000000
   1      0      1       1 GB   00000000000
   2      0      2       1 GB   00000000000
   3      0      3       1 GB   00000000000
   0      1      4       1 GB   00000000000
   1      1      5       1 GB   00000000000
   2      1      6       1 GB   00000000000
   3      1      7       1 GB   00000000000
 Total Available         8 GB   32-Way Interleave

 Board   Set   Array    Size      Address       QBB 1 Memory
   0      0      0       1 GB   01000000000
   3      0      3       1 GB   01000000000
   0      1      4       1 GB   01000000000
   3      1      7       1 GB   01000000000
 Total Available         4 GB   16-Way Interleave
```

❺ **QBB1**. QBB1 components are listed.

❻ **PCI I/O information, PCI box 1**. QBB1 is connected to only one PCI box, PCI box 1.

QBB1 I/O port 0 is linked to remote I/O riser 0 located on the right side of PCI box 1. Logical hose numbers are 8 and 9.

QBB1 I/O port 1 is linked to remote I/O riser 1 located on the left side of PCI box 1. Logical hose numbers are 10 and 11.

❼ The total system memory size is reported.

QBB0 has 8 GB in a 32-way interleave; QBB1 has 4 GB in a 16-way interleave.

❽ Standard I/O modules in the system. In this example, three PCI boxes (0, 1, and 3) each have a standard I/O module. The primary standard I/O module is in PCI box 0. Hose and IOP connections are also listed.

❾ System memory shown by board, set, array, and size.

Continued on next page

Show Config Example (Continued)

```
⑩                                ①                          ②    ③    ④       ⑤        ⑥
PCI Box  Riser  Slot  Option                    Hose  Bus  Slot  Function  Name
    0       0      1   Standard I/O Module        0
                       + Acer Labs M1543C         0    0     7
                       + Acer Labs M1543C IDE     0    0    15              dqa
                       + Acer Labs M1543C IDE     0    0    15              dqb
                       + Acer Labs M1543C USB     0    0    19
                       + QLogic ISP10x0           0    0     1              pka
    0       0      2   DE500-BA Network Con       0    0     2              ewa
    0       0      3   ELSA GLoria Synergy        0    0     3              vga
    0       0      7   DEGPA-SA                   1    0     7
    0       1      1   QLogic ISP10x0             2    0     1              pkb
    0       1      2   NCR 53C896                 2    0     2        0     pkc
                       + NCR 53C896               2    0     2        1     pkd
    0       1      3   09608086                   2    0     3        0
                       + 19608086/0415129A        2    0     3        1
    0       1      6   DEC KZPSA                  3    0     6              pke
    0       1      7   DEC PCI FDDI               3    0     7              fwa
    1       0      1   Standard I/O Module        8
                       + Acer Labs M1543C         8    0     7
                       + Acer Labs M1543C IDE     8    0    15              dqe
                       + Acer Labs M1543C IDE     8    0    15              dqf
                       + Acer Labs M1543C USB     8    0    19
                       + QLogic ISP10x0           8    0     1              pkg
    1       0      2   DEC KZPSA                  8    0     2              pkh
    1       0      3   PowerStorm 350             8    0     3              vgb
    1       0      5   DECchip 21154-AA           9    0     5
                       + DE602-AA                 9    2     4              eia
                       + DE602-AA                 9    2     5              eib
    1       0      7   DEC PCI MC                 9    0     7              mca
    1       1      1   QLogic ISP10x0            10    0     1              pki
    1       1      2   DEC KZPSA                 10    0     2              pkj
    1       1      3   DEC KZPSA                 10    0     3              pkk
    1       1      7   DEGPA-SA                  11    0     7
    3       0      1   Standard I/O Module        4
                       + Acer Labs M1543C         4    0     7
                       + Acer Labs M1543C IDE     4    0    15              dqc
                       + Acer Labs M1543C IDE     4    0    15              dqd
                       + Acer Labs M1543C USB     4    0    19
                       + QLogic ISP10x0           4    0     1              pkf
    3       1      1   DE500-BA Network Con       6    0     1              ewb
    3       1      2   PBXNP-AA Token Ring        6    0     2              dwa
    3       1      7   001B1011/001B1011          7    0     7              dqa

⑦      ⑧                        ⑨
Slot   Option                   Hose 0, Bus 0, PCI
   1    QLogic ISP10x0           pka0.7.0.1.0            SCSI Bus ID 7  ⑩
                                 dka0.0.0.1.0            COMPAQ BB00921B91
                                 dka400.4.0.1.0          RRD46
   2    DE500-BA Network Con     ewa0.0.0.2.0            08-00-2B-C3-C3-B9
   3    ELSA GLoria Synergy
```

Show Config Example (Continued)

```
  7    Acer Labs M1543C                                  Bridge to Bus 1, ISA
 15    Acer Labs M1543C IDE    dqa.0.0.15.0
                               dqb.0.1.15.0
                               dqa0.0.0.15.0             COMPAQ CDR-8435
 19    Acer Labs M1543C USB
```

⑩ **PCI Box, Riser, Slot**. Each PCI box in the system is identified by a number (0 to F hexadecimal). A system can have a maximum of 16 PCI boxes. The physical location of options in the PCI box is identified by the remote I/O riser (0 or 1) and slot number in the PCI box.

① **Option**. Options are listed. Note that the standard I/O module occupies slot 1, on hose 0, riser 0, PCI box 0.

② **Hose**. PCI devices connect to logical hoses in the PCI box. A **logical** hose number identifies the logical hose for each device. The firmware assigns the logical hose number (0 to 63), as described in Section 2.7. Each PCI box has a maximum of four logical hose numbers.

③ **Bus**. The remote I/O riser PCI bus will always be numbered 0. Buses behind device bridges are numbered 1, 2, 3, etc.

④ **Slot**. This is the logical slot ID number of each device or controller.

⑤ **Function**. On multifunction devices, the function ID is displayed.

⑥ **Name**. Device names are listed.

⑦ This next section of the **show config** output lists I/O devices in order by logical hose number (see ⑨). The logical **Slot** number for each option appears in the first column.

⑧ The option name appears next.

⑨ **Hose 0, Bus 0, PCI**. In this example, the standard I/O module is logical slot 1 on PCI Bus 0, Hose 0. See ① for a different presentation of this information.

⑩ Because the standard I/O module is multifunctional, the next devices shown are the controllers and devices connected to the module. COMPAQ BB00921B91 and RRD46 are devices on **SCSI bus ID 7.**

Continued on next page

Show Config Example (Continued)

```
Slot   Option                  Hose 1, Bus 0, PCI
   7    DEGPA-SA

Slot   Option                  Hose 2, Bus 0, PCI
   1    QLogic ISP10x0          pkb0.7.0.1.2            SCSI Bus ID 7
                                dkb0.0.0.1.2            RZ1CB-CA
                                dkb100.1.0.1.2          RZ1CB-CS
                                dkb200.2.0.1.2          RZ1CB-CS
                                dkb400.4.0.1.2          COMPAQ BB00911CA0
                                dkb500.5.0.1.2          RZ1DF-CB
                                dkb600.6.0.1.2          RZ1DF-CF
   2/0  NCR 53C896              pkc0.7.0.2.2            SCSI Bus ID 7
   2/1  NCR 53C896              pkd0.7.0.102.2          SCSI Bus ID 7
   3/0  09608086                                        Bridge to Bus 2, PCI
   3/1  19608086/0415129A
 Slot   Option                  Hose 3, Bus 0, PCI                      ❶
   6    DEC KZPSA               pke0.7.0.6.3            SCSI Bus ID 7
                                dke100.1.0.6.3          RZ1CF-CF
                                dke200.2.0.6.3          RZ1CF-CF
                                dke300.3.0.6.3          COMPAQ BB00911CA0
                                dke400.4.0.6.3          RZ1CF-CF
   7    DEC PCI FDDI            fwa0.0.0.7.3            00-00-F8-48-56-7A

 Slot   Option                  Hose 4, Bus 0, PCI
   1    QLogic ISP10x0          pkf0.7.0.1.4            SCSI Bus ID 7
                                dkf0.0.0.1.4            RZ1DF-BF
   7    Acer Labs M1543C                                Bridge to Bus 1, ISA
  15    Acer Labs M1543C IDE    dqc.0.0.15.4
                                dqd.0.1.15.4
                                dqc0.0.0.15.4           TOSHIBA CD-ROM XM-6302B
  19    Acer Labs M1543C USB

 Slot   Option                  Hose 5, Bus 0, PCI
   4    QSW ELan3

 Slot   Option                  Hose 6, Bus 0, PCI
   1    DE500-BA Network Con    ewb0.0.0.1.6            08-00-2B-C3-C0-38
   2    PBXNP-AA Token Ring

 Slot   Option                  Hose 7, Bus 0, PCI
   7    001B1011/001B1011
 Slot   Option                  Hose 8, Bus 0, PCI
   1    QLogic ISP10x0          pkg0.7.0.1.8            SCSI Bus ID 7
                                dkg0.0.0.1.8            RZ1DF-BF
   2    DEC KZPSA               pkh0.7.0.2.8            SCSI Bus ID 7
                                dkh100.1.0.2.8          RZ1CF-CF
                                dkh200.2.0.2.8          RZ1CB-CS
                                dkh300.3.0.2.8          COMPAQ BB00911CA0
                                dkh400.4.0.2.8          COMPAQ BB00911CA0
```

Continued on next page

Show Config Example (Continued)

```
   3    PowerStorm 350
   7    Acer Labs M1543C                                  Bridge to Bus 1, ISA
  15    Acer Labs M1543C IDE    dqe.0.0.15.8
                                dqf.0.1.15.8
                                dqe0.0.0.15.8             COMPAQ CDR-8435
  19    Acer Labs M1543C USB


 Slot   Option                  Hose 9, Bus 0, PCI
   5    DECchip 21154-AA                                  Bridge to Bus 2, PCI
   7    DEC PCI MC                                        Rev: 22, mca0

 Slot   Option                  Hose 9, Bus 2, PCI
   4    DE602-AA                eia0.0.0.2004.9           00-50-8B-6F-2F-92
   5    DE602-AA                eib0.0.0.2005.9           00-50-8B-6F-2F-93

 Slot   Option                  Hose 10, Bus 0, PCI
   1    QLogic ISP10x0          pki0.7.0.1.10             SCSI Bus ID 7
   2    DEC KZPSA               pkj0.7.0.2.10             SCSI Bus ID 7
                                dkj100.1.0.2.10           RZ1CF-CF
                                dkj200.2.0.2.10           RZ1CB-CS
                                dkj300.3.0.2.10           RZ1CF-CF
                                dkj400.4.0.2.10           RZ1CB-CS
   3    DEC KZPSA               pkk0.7.0.3.10             SCSI Bus ID 7

 Slot   Option                  Hose 11, Bus 0, PCI
   7    DEGPA-SA

P00>>>                                                                  ❷
```

❶ Looking back at the information presented under ⑩ helps to locate the physical PCI slot of each device listed in the logical PCI device output. For example, to find where the DEC KZPSA, Hose 3, Bus 0, PCI device is located, see 3 in the Hose column ②, identify the option by name, and see that it is in PCI box 0, riser 1, slot 6.

❷ The console prompt is displayed.

7.5.36 Show Device

The show device command displays device information for devices on the system.

Syntax

sh[ow] dev[ice] [*dev_name*]

Argument

dev_name Any adapter name (wildcarding is allowed). For example, **show device dk*** will display information on all SCSI devices on the system. If *dev_name* is omitted, the display shows all devices in the system.

Example

```
P00>>> sho dev
dka0.0.0.1.0              DKA0                      RZ1DF-BF  1614
dkb0.0.0.7.1              DKB0              COMPAQ BB00911CA0  3B05
dkb100.1.0.7.1            DKB100            COMPAQ BB00911CA0  3B05
dkb200.2.0.7.1            DKB200            COMPAQ BB00911CA0  3B05
dkb300.3.0.7.1            DKB300            COMPAQ BB00911CA0  3B05
dqa0.0.0.15.0             DQA0  ❶     TOSHIBA CD-ROM XM-6302B  1017
dqc0.0.0.15.6             DQC0        TOSHIBA CD-ROM XM-6302B  1017
dqe0.0.0.15.8             DQE0        TOSHIBA CD-ROM XM-6302B  1017
eia0.0.0.3.8              EIA0              08-00-2B-C3-BD-00
fwa0.0.0.4.1              FWA0              00-00-F8-CD-1F-39
fwb0.0.0.5.7              FWB0              00-00-F8-CD-22-E0
fwc0.0.0.1.10             FWC0              00-00-F8-CD-22-FB
pga0.0.0.7.7              PGA0        WWN 1000-0000-c920-da1e
pka0.7.0.1.0              PKA0                   SCSI Bus ID 7  5.57
pkb0.7.0.7.1              PKB0                   SCSI Bus ID 7  5.57
pkc0.7.0.1.2              PKC0                   SCSI Bus ID 7  5.57
```

Table 7–6 Device Naming Conventions

❶	Category	Description
dq	Driver ID	Two-letter designator of port or class driver dk SCSI drive or CD ew Ethernet port dq IDE CD-ROM fw FDDI device dr RAID set device mk SCSI tape ei Ethernet port pk SCSI port
a	Storage adapter ID	One-letter designator of storage adapter (a, b, c…).
0	Device unit number	Unique number (MSCP unit number). SCSI unit numbers are forced to 100 X node ID.
0	Bus node number	Bus node ID.
0	Channel number	Used for multichannel devices.
15	Logical slot number	The slot number assigned by the firmware.
0	Hose number	The hose number assigned by the firmware.

7.5.37 Show *envar* Command

Show envar displays the current state of the specified environment variable.

Syntax

sh[ow] *envar*

or

sh[ow] *

envar An environment variable name (see Table 7–5). Wildcarding can be used. Unambiguous abbreviations can be used for an environment variable name when using this command. See the **set <envar>** command for related information.

* Show all environment variables and their current values.

Example

```
P00>>> show boot*  ❶
boot_dev                dka0.0.0.1.1
boot_file
boot_osflags            a
boot_reset              OFF
bootdef_dev             dka0.0.0.1.1
booted_dev
booted_file
booted_osflags
```

❶ The user issues a command to list all environment variables beginning with the characters “boot” and followed by any characters. The command displays the boot environment variables and their current values.

7.5.38 Show FRU Command

The show fru command displays the physical configuration of field replaceable units (FRUs).

Syntax

sh[ow] fru

Example

❶ ❷ ❸ ❹ ❺

```
P00>>> show fru
FRUname           E  Part#              Serial#    Model/Other Alias/Misc
CAB0.PWR          00 -                  -          -           -
QB01.RAK1         00 -                  -          -           -
QB01.PS1          00 -                  -          -           -
QB01.PS3          00 -                  -          -           -
QB23.RAK2         00 -                  -          -           -
QB23.PS1          00 -                  -          -           -
QB23.PS2          00 -                  -          -           -
CAB1.SYS          00 -                  -          -           -
CAB1.SYSFAN       00 -                  -          -           -
QBB0.             00 54-25045-01.A01    NI93900000 WF08LTA111
QBB0.PSM          00 B12345-AA.H02      NI12312312
QBB0.PWR          00 54-25017-01.D01    NI90665954 WF08LTA111
QBB0.AUX          00 54-25123-01.D03    NI93060135 WF08LTA111
QBB0.CLKSPLT      00 -                  -          -           -
    .
    .
    .
PBP0.PS2          00 -                  -          -           -
PBP0.PS1          00 -                  -          -           -
PBP1.PCI0         00 -                  -          -           -
QBB1.IOR01        00 -B4170-AA.E01      NI94173767 WF08LTA111
PBP1.SYSFAN2      00 -                  -          -           -
PBP1.SYSFAN1      00 -                  -          -           -
PBP1.PS2          00 -                  -          -           -
PBP1.PS1          00 -                  -          -           -
```

❶ The name of the FRU; acronyms are:

CAB	Power cabinet (0), System cabinet 1 (1), System cabinet 2 (2)
RAK	DC power subrack
PS	Individual DC power supplies
PBP	PCI backplane
SIO	Standard I/O module
RIO	Remote I/O riser (located in PCI box)
QBB	Quad building block
PSM	Power system manager module
PWR	Main power module
AUX	Auxiliary power module
CPU	CPU module
MEM	Memory module
DIM	DIMM memory array
IO	Local I/O riser (located in QBB)
GP	Global Port module
CLKSPLT	Clock splitter module
SYSFAN	System or PCI box blower
PCI	Individual PCI modules

❷ Error field. FRUs without errors show 00 (hex). Other values are:

01 Hardware failure
02 TDD error has been logged
04 SDD error has been logged
08 Reserved
10 Checksum failure on bytes 0-62
20 Checksum failure on bytes 64-126
40 Checksum failure on bytes 128-524
80 FRU's system serial number does not match system's

❸ The part number by which the FRU can be ordered.

❹ The serial number of the FRU.

❺ Optional manufacturing specifications; usually blank.

7.5.39 Show Memory Command

The show memory command shows the configuration of main memory on the system.

Syntax

sh[**ow**] **mem**[**ory**] [**-br**[**ief**], **-fu**[**ll**]]

Options

-brief A summary display of memory is given.

-full Detail on specific DIMMs is given in addition to the board information.

Example

```
P00>>> sh mem
System Memory 8 GB    ❶
   ❷      ❸      ❹         ❺         ❻              ❼
 Board    Set    Array     Size      Address     QBB 0 Memory
   0        0        0       1 GB    00000000000
   1        0        1       1 GB    00000000000
   2        0        2       1 GB    00000000000
   3        0        3       1 GB    00000000000
   0        1        4       1 GB    00000000000
   1        1        5       1 GB    00000000000
   2        1        6       1 GB    00000000000
   3        1        7       1 GB    00000000000
Total Available          8 GB   32-Way Interleave  ❽

P00>>> sh mem -br    ❾
System Memory 8 GB

 QBB    Boards    Arrays     Size     Interleave
  0        4         8        8 GB      32-Way
```

```
P00>>> sh mem -fu    ❿
System Memory 8 GB

 Board   Set   Array    Size      Address    QBB 0 Memory
   0      0      0       1 GB   00000000000
   1      0      1       1 GB   00000000000
   2      0      2       1 GB   00000000000
   3      0      3       1 GB   00000000000
   0      1      4       1 GB   00000000000
   1      1      5       1 GB   00000000000
   2      1      6       1 GB   00000000000
   3      1      7       1 GB   00000000000
 Total Available          8 GB   32-Way Interleave

 DIMM    Size    Memory Array               QBB 0 Directory
   0      2 GB          0
   1      2 GB          1
   2      2 GB          2
   3      2 GB          3
   4      2 GB          4
   5      2 GB          5
   6      2 GB          6
   7      2 GB          7
P00>>>
```

❶ The total system memory size is reported, in all three types of displays.

❷ Each memory module (board) in the QBB is listed (0 to 3). Each QBB can have up to four memory modules.

❸ Each memory module has two sets (arrays) of DIMMs installed. A set is numbered 0 or 1. Each set consists of four DIMMs.

❹ In this example, all DIMMs have the same density (4 GB) and are in a 32-way interleave. The first array on board 0 is **board 0, set 0,** and is referred to as **array 0**; the second array on board 0 is **board 0, set 1,** and is referred to as **array 4**, and so forth.

❺ The size, or density of the array.

❻ The starting address of the array.

❼ Identifies each QBB in the system.

❽ The total available memory in the QBB.

❾ In the **-brief** display, memory data for each QBB is summarized.

❿ In the **–full** display, additional information on each DIMM in each array is given.

7.5.40 Show PAL and Show Version Commands

The show pal command displays the versions of Tru64 UNIX and OpenVMS PALcode. The show version command shows the version of the SRM console firmware code.

Syntax

sh[**ow**] {**pal**, **version**}

Example

```
P00>>> sho pal
pal       OpenVMS PALcode V1.81-1, Tru64 UNIX PALcode V1.75-1
P00>>>
P00>>>
P00>>>
P00>>> sho version
version                 V5.8-1 May 26 2000 12:15:01
P00>>>
```

7.5.41 Show_status Command

The show_status command is used to display information on system exercisers and diagnostic firmware running in the background.

Syntax

show_status

Example

```
P00>>> 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
000002b9      memtest memory            2     0     0    2667577344    2667577344
000002bf      memtest memory            2     0     0    2634022912    2634022912
000002c5      memtest memory            3     0     0    2667577344    2667577344
00000314      memtest memory            3     0     0    2684354560    2684354560
0000032d      memtest memory            2     0     0    2147344384    2147344384
00000346      memtest memory            4     0     0    3220881408    3220881408
0000035f      memtest memory            2     0     0    2147352576    2147352576
00000368      memtest memory            4     0     0    3220881408    3220881408
000003bd     exer_kid dka0.0.0.1.0      0     0     0             0       4128256
000003be     exer_kid dkb0.0.0.5.1      0     0     0             0       1092096
000003bf     exer_kid dkb100.1.0.5      0     0     0             0       1091584
000003c0     exer_kid dkb400.4.0.5      0     0     0             0       1091584
000003c1     exer_kid dkb500.5.0.5      0     0     0             0       1091584
000003cb     exer_kid dqa0.0.0.15.      0     0     0             0       1475584
000003de     exer_kid dva0.0.0.100      0     0     0             0        241664
0000040d      nettest ewa0.0.0.2.2    112     0     0        157536        157696
```

❶ The user enters the **show_status** command.

❷ Process ID. Can be used in a **kill** command to stop a specific test.

❸ The type of exerciser or test being run.

❹ The device being tested.

❺ The number of complete passes the exerciser has made thus far.

❻ The number of hard/soft errors that have been detected.

❼ The number of bytes read/written by the exerciser thus far.

7.5.42 Sys_exer Command

The sys_exer command tests the entire system, including memory, disks, tapes, serial ports, parallel port, network, and VGA.

Syntax

sys_exer [**-t** *runtime*]

-t *n* — Specifies the time, in seconds, that the exerciser is to run. A prompt will not be displayed until the time has expired and the **kill_diags** script has completed. The default is 0, run forever.

Description

All tests run concurrently for the run time specified (default is forever). The **sys_exer** command can be run as either a background or foreground process.

Use the **set** command to establish parameters, such as whether to halt, loop, or continue on error, as described in the *AlphaServer GS80/160/320 Service Manual*. The passcount environment variable, **d_passes**, is ignored by **sys_exer**.

Example

```
P00>>> sys_exer &     ❶
Default zone extended at the expense of memzone.
Use INIT before booting
Exercising the Memory
memtest -bs 1000000 -rb -p 0 &
memtest -bs 1000000 -rb -p 0 &
memtest -bs 1000000 -rb -p 0 &
memtest -bs 1000000 -rb -p 0 &
memtest -sa 80000000 -ea FFFDE000 -z -p 0 &
memtest -sa 1000000000 -ea 103FFE4000 -z -p 0 &
memtest -sa 2000000000 -ea 207FFE0000 -z -p 0 &
memtest -sa 3000000000 -ea 303FFE4000 -z -p 0 &
Exercising the DK* Disks (read only)
Exercising the DQ* Disks (read only)
Exercising the EW* Network
Type "show_status" to display testing progress
Type "cat el" to redisplay recent errors
Type "init" in order to boot the operating system
0000040d      nettest ewa0.0.0.2.2     19    0    0         26592         26752
```

Continued on the next page

```
P00>>> show_status ❷
 ID       Program      Device       Pass  Hard/Soft Bytes Written  Bytes Read
-------- ----------- ------------ ------ --------- ------------- -------------
00000001         idle system           0    0    0             0             0
000002b9      memtest memory           2    0    0    2667577344    2667577344
000002bf      memtest memory           2    0    0    2634022912    2634022912
000002c5      memtest memory           3    0    0    2667577344    2667577344
00000314      memtest memory           3    0    0    2684354560    2684354560
0000032d      memtest memory           2    0    0    2147344384    2147344384
00000346      memtest memory           4    0    0    3220881408    3220881408
0000035f      memtest memory           2    0    0    2147352576    2147352576
00000368      memtest memory           4    0    0    3220881408    3220881408
000003bd     exer_kid dka0.0.0.1.0     0    0    0             0       4128256
000003be     exer_kid dkb0.0.0.5.1     0    0    0             0       1092096
000003bf     exer_kid dkb100.1.0.5     0    0    0             0       1091584
000003c0     exer_kid dkb400.4.0.5     0    0    0             0       1091584
000003c1     exer_kid dkb500.5.0.5     0    0    0             0       1091584
000003cb     exer_kid dqa0.0.0.15.     0    0    0             0       1475584
0000040d      nettest ewa0.0.0.2.2   112    0    0        157536        157696
P00>>> kill_diags ❸
Loop Reply from: 08-00-2b-c4-7f-ee
Loop Reply from: 08-00-2b-c4-7f-ee
Loop Reply from: 08-00-2b-c4-7f-ee
Loop Reply from: 08-00-2b-c4-7f-ee
P00>>> show_status ❹
 ID       Program      Device       Pass  Hard/Soft Bytes Written  Bytes Read
-------- ----------- ------------ ------ --------- ------------- -------------
00000001         idle system           0    0    0             0             0
P00>>>
```

❶ The user types the **sys_exer** command to run in the background, as specified by the ampersand (**&**).. The exerciser displays a message as it starts each exerciser and provides other information on useful related commands.

❷ After some time, the user executes a **show_status** command to look at the progress of the exercisers. As shown, each exerciser has completed a number of passes on each test.

❸ The user executes a **kill_diags** command to stop the exercisers.

❹ This **show_status** shows that all diagnostic processes have stopped.

7.5.43 Test Command

The test command allows you to test the entire system.

Syntax

test

Description

The test command tests the entire system, including memory, disks, tapes, serial ports, parallel port, network, and VGA.

All tests run serially for a minimum of 10 seconds per test. The run time of a test is proportional to the amount of memory to be tested and the number of disk drives to be tested.

Only one instance of **test** can be run at a time; **test** can be run as either a background or foreground process.

Use the **set** command to establish parameters, such as whether to halt, loop, or continue on error, as described in the *AlphaServer GS80/160/320 Service Manual*. The passcount environment variable, **d_passes**, is ignored by **test**.

Example

```
P00>>> test
Testing the DK* Disks (read only)
No DU* Disks available for testing
No DR* Disks available for testing
Testing the DQ* Disks (read only)
No DF* Disks available for testing
No MK* Tapes available for testing
No MU* Tapes available for testing
No VGA available for testing
Testing the EW* Network
P00>>>
```

7.5.44 Wwidmgr Command

Wwidmgr manages wwid device registration on the Fibre Channel loop or fabric.

Syntax

wwidmgr [**-quickset** { **-item** *n*, **-udid** *n*}]
[**-set** { **wwid** | **port** } **–item** *n* [**-unit** *n*] [**-col** *n*] [**-filter** *string*]
[**-show** { **wwid** | **port** } [**-full**] [**-filter** *string*]
[**-show** { **ev** | **reachability** }
[**-clear** { **all** | **wwid** *n* | **N***n* }]

Options

-quickset	Sets up a small integer alias for a WWID in the environment variables.
	-item *n* Specifies a WWID or PORT menu item
	-udid *n* Specifies a UDID
-set {**wwid | port** }	Sets up a small integer alias for a WWID in the environment variables.
	-item *n* Specifies a WWID or PORT menu item.
	-unit *n* Specifies unit number associated with WWID.
	-col *n* Specifies a collision value. The default is 1.
	-filter *string* Specifies a string used to narrow the displays of **–set.**
-show {**wwid** | **port** }	Displays information about the WWID or N_ports.
	-full Provides more detailed information.
	-filter *string* Specifies a string used to narrow the displays of **–show**.
-show { **ev** | **reachability** }	Displays information on FC environment variables, or the reachability of devices.
-clear	Clears the FC related environment variables, either one at a time or all at once.

NOTE: *Documents describing wwidmgr are available under the names wwidmgr.pdf or wwidmgr.ps at ftp://ftp.digital.com/pub/Digital/Alpha/firmware/readmes/v5.7/doc/*

7.5.45 Comment (#)

A comment can be introduced using the # symbol. The entire text following the # and before Return is ignored.

Example

```
1. P00>>> # This example illustrates the comment command.
   P00>>>

2. P00>>> exam pmem:0400EC  # Examine physical memory.
          pmem: 000400EC D0FFFFFD
   P00>>>
```

Appendix A
Boot Options

Table A-1 lists the *Tru64 UNIX* options used with the **boot** command. Table A-2 lists the options used with the **boot** command for the *OpenVMS* operating system. These options allow you to control various phases of booting.

Table A–1 Tru64 UNIX Boot Options

Option	Function
a	Boots the system disk to multiuser mode.
d	Do full dumps.
i	Boot to interactive mode plus options (prompt for system image to boot and boot options.
s	Boot to single-user mode.

Table A–2 OpenVMS Boot Options

Hexadecimal Value	Function
1	Allows a conversational boot.
2	Maps XDELTA to a running system.
4	Stops the boot procedure at the initial system breakpoint.
8	Performs a diagnostic bootstrap.
10	Stops the boot procedure at the bootstrap breakpoints.
20	Omits the header from the secondary bootstrap image.
40	Inhibits memory testing.
80	Prompts for the name of the secondary bootstrap file.
100	Halts the system before the secondary bootstrap.
2000	Marks corrected read data error pages as bad.
10000	Enables debug messages in the APB.EXE, SYSBOOT.EXE, and EXEC_INIT.EXE files.
8200	Enables user messages in the APB.EXE, BOOT.EXE, and EXEC_INIT.EXE files.

Appendix B
Loadable Firmware Update Utility

This appendix describes the Loadable Firmware Update utility (LFU). Sections include:

- Overview
- Booting LFU
- Display Command
- Exit Command
- List Command
- Update Command
- Verify Command

B.1 Overview

LFU allows you to update and verify firmware on *AlphaServer* GS80/160/320 systems.

Example B–1 LFU Menu Screen

```
***** Loadable Firmware Update Utility *****
--------------------------------------------------------------------
 Function    Description
--------------------------------------------------------------------

Display    Displays the system's configuration table.
Exit       Done exit LFU (reset).
List       Lists the device, revision, firmware name, and update
           revision.
Update     Replaces current firmware with loadable data image.
Verify     Compares loadable and hardware images.
? or Help  Scrolls this function table.
--------------------------------------------------------------
```

The Loadable Firmware Update utility (LFU) is a software program that allows you to update and verify the firmware on a variety of microprocessors and adapters on nonpartitioned *AlphaServer* GS80/160/320 systems. If your system is partitioned, partitions must be removed before LFU can be run, as described in the *AlphaServer GS80/160/320 Service Manual.*

LFU can update images for the CPU flash ROM on the standard I/O module, CSB micro flash ROM (SCM, PSM, PBM, HPM), and PCI adapter EEPROMs (PCI/SCSI, PCI/NI, PCI/FDDI).

Example B–1 shows the banner displayed when LFU is booted. It summarizes the commands available.

B.2 Booting LFU

LFU can be booted from DVD/CD-ROM or from over the network.

Example B–2 Booting LFU from the Network

```
P00>>> boot -file  T410_WFUPDATE  ewa0
❶
(boot ewa0.0.0.1.2 -file  T410_WFUPDATE   -flags a)

Trying MOP boot.
.................................
Network load complete.
Host name: DUFFY
Host address: aa-00-04-00-dc-74
.
.
.
Option firmware files were not found on CD.
If you want to load the options firmware,
please enter the device on which the files are located(ewa0),
or just hit <return> to proceed with a standard console update:


                ***** Loadable Firmware Update Utility *****
❷
-------------------------------------------------------------------
 Function    Description
-------------------------------------------------------------------

Display    Displays the system's configuration table.
Exit       Done exit LFU (reset).
List       Lists the device, revision, firmware name, and update
           revision.
Update     Replaces current firmware with loadable data image.
Verify     Compares loadable and hardware images.
? or Help  Scrolls this function table.
-------------------------------------------------------------------
UPD>    ❸
```

LFU can be booted over the network, from the latest firmware CD-ROM (distributed quarterly), or from diskettes created for updating. Example B–2 shows an example boot.

❶ In this case, LFU is booted over the network. See the discussion of the **boot** command in Chapter 7 for further information on booting.

❷ LFU boots and its banner displays.

❸ The LFU command prompt displays. LFU is ready to accept commands.

B.3 Display Command

The display command presents the I/O section of the SRM show config command.

Example B–3 Display Command

```
UPD> display

                        Compaq Computer Corporation
                       Compaq AlphaServer GS80 6/731

 PCI Box  Riser  Slot  Option                  Hose  Bus  Slot  Function  Name
    0       0      1   Standard I/O Module       6
                       + Acer Labs M1543C        6    0     7
                       + Acer Labs M1543C IDE    6    0    15             dqa
                       + Acer Labs M1543C USB    6    0    19
                       + QLogic ISP10x0          6    0     1             pka
    0       0      2   ELSA GLoria Synergy       6    0     2             vga
    0       0      4   QLogic ISP10x0            7    0     4             pkb
    0       0      6   DECchip 21154-AA          7    0     6
                       + DE602-AA                7    2     4             eia
                       + DE602-AA                7    2     5             eib
    1       0      1   Standard I/O Module      12
                       + Acer Labs M1543C       12    0     7
                       + Acer Labs M1543C IDE   12    0    15             dqb
                       + Acer Labs M1543C USB   12    0    19
                       + QLogic ISP10x0         12    0     1             pkc
    1       0      2   DECchip 21152-AA         12    0     2
                       + Permedia - P2V Graph   12    2     4             vgb
                       + Intel 8255x Ethernet   12    2     5             eic

 Hose  QBB  PCA  PCI Bus  Slots
   6    0    3      0      1-3
   7    0    3      1      4-7
  12    1    2      0      1-3
  13    1    2      1      4-7
  14    1    3      0      1-3
  15    1    3      1      4-7

 Slot   Option                 Hose 6, Bus 0, PCI
   1    QLogic ISP10x0         pka0.7.0.1.6          SCSI Bus ID 7
   2    ELSA GLoria Synergy
   7    Acer Labs M1543C                             Bridge to Bus 1, ISA
  15    Acer Labs M1543C IDE   dqa.0.0.15.6
  19    Acer Labs M1543C USB


 Slot   Option                 Hose 7, Bus 0, PCI
   4    QLogic ISP10x0         pkb0.7.0.4.7          SCSI Bus ID 7
                               dkb100.1.0.4.7        RZ2ED-LS
                               dkb200.2.0.4.7        RZ2ED-LS
```

```
                                dkb300.3.0.4.7          RZ2ED-LS
   6    DECchip 21154-AA                                Bridge to Bus 2, PCI

 Slot   Option                  Hose 7, Bus 2, PCI
   4    DE602-AA                eia0.0.0.2004.7         00-50-8B-CF-1A-AA
   5    DE602-AA                eib0.0.0.2005.7         00-50-8B-CF-1A-AB

 Slot   Option                  Hose 12, Bus 0, PCI
   1    QLogic ISP10x0          pkc0.7.0.1.12           SCSI Bus ID 7
                                dkc0.0.0.1.12           COMPAQ BB00921B91
   2    DECchip 21152-AA                                Bridge to Bus 2, PCI
   7    Acer Labs M1543C                                Bridge to Bus 1, ISA
  15    Acer Labs M1543C IDE    dqb.0.0.15.12
                                dqb0.0.0.15.12          COMPAQ CDR-8435
  19    Acer Labs M1543C USB



 Slot   Option                  Hose 12, Bus 2, PCI
   4    Permedia - P2V Graph
   5    Intel 8255x Ethernet    eic0.0.0.2005.12        00-06-2B-00-6E-9B
UPD>
```

B.4 Exit Command

The exit command terminates execution of LFU and the SRM console prompt returns.

Example B–4 Exit Command

```
UPD> exit    ❶
Initializing…

   [power-up map displays]

P00>>>
```

❶ The exit command initializes the system and terminates execution of LFU, returning control to the SRM console.

NOTE: *LFU terminates automatically when the* ***update micro*** *command is completed, and returns control to the SRM console.*

B.5 List Command

The list command displays the files on the system that can be updated, giving the device, revision, firmware name, and update revision.

Example B–5 List Command

```
UPD> list ❶
   ❷                ❸                ❹                ❺
Device          Current Revision       Filename          Update Revision

SRM               V5.8-1               srm_fw             V5.8-1
micro             T5.4(03.23/16:11)    micro_fw           x5.7 (05.26/12:15)
```

❶ The user issues the **list** command.

❷ The device name field is what you type in an **update** command to update a particular piece of firmware.

❸ This column lists the current revision level of the firmware on the system.

❹ The name of the file to be updated. Note that LFU does not list each firmware file associated with each micro. Instead it lumps them into one file, micro_fw.

❺ The version of the firmware on the update media.

B.6 Update Command

The update command copies the new firmware from the update media to the appropriate device.

Example B–6 Update Command for a PCI device

```
UPD> u fwa0      ❶
Confirm update on:
Fwa0
[Y/(N)] y
WARNING: updates may take several minutes to complete for each device.

                          DO NOT ABORT!

Fwa0             Updating to 3.10...     Verifying 3.10...  PASSED.

UPD>
```

Example B–7 Update Command for the SRM console

```
UPD> u srm*      ❷
Confirm update on:
SRM46
SRM
[Y/(N)] y
WARNING: updates may take several minutes to complete for each device.

                          DO NOT ABORT!
  ❸
SRM46            Updating to V5.8-1...  Verifying x5.8-1...  PASSED.

SRM              Updating to V5.8-1...  Verifying x5.8-1...  PASSED.
```

Example B–8 Update Command for Micros

```
UPD> update micro                                                    ❹
Confirm update on:
micro
[Y/(N)] y                                                            ❺
WARNING: updates may take several minutes to complete for each device.

                          DO NOT ABORT!

micro            Updating to V5.1(02.23/01:07)...                    ❻
```

```
Updating SCM nodes E0,E1                                          ❼
Update Cmd processed
Transferring hex file...........Flash ON........Flash ON........Flash
ON........
Flash ON....$<XFE>

Initializing shared ram                                           |
Modified SR init-SCM previously ON                                |
Shared RAM Initialized                                            |
Master SCM                                                        |
Testing SCM EEPROM - Passed                                       |
Initializing EVs                                                  |     ❽
SCM Selftest Passed                                               |
Polling CSB............................                           |
OCP will be inactive for first 12 seconds after micro reset       |
```

❶ An **update** command is issued for a PCI device. Confirmation is required and the update proceeds.

❷ An **update** command is issued for the SRM console firmware. A * wildcard is used to incorporate all SRM update files.

❸ Two files are updated: SRM46 and SRM.

❹ The LFU **update micro** command is issued.

❺ Confirmation of the update is required.

❻ SCM update and micro update begin.

❼ SCMs are found at nodes E0 and E1. Updating the master and slave SCM commences in parallel and completes.

❽ Once the new SCM code is written into flash, the master SCM polls and maps the entire CSB prior to LFU continuing and updating the other micros in the system.

Example B–8 Update Command for Micros (Continued)

```
Updating XSROM node 30,31,32,33                                    ❾
Update Cmd processed
Transferring hex file..................
~I~ Flashing node 30 (please wait)
~I~ Flashing node 31 (please wait)
~I~ Flashing node 32 (please wait)
~I~ Flashing node 33 (please wait)
CSB download of .Hex file complete

Updating HPM node 40                                               ❿
Update Cmd procesed
Transferring hex file..................
CSB download of .Hex file complete

Updating PSM node 330,31,32,33                                     ⓫
Update Cmd processed
Transferring hex file.......................
CSB download of .Hex file complete


Updating PBM nodes 10,11                                           ⓬
Update Cmd processed
Transferring hex file..................
CSB download of .Hex file complete

The Master SCM will now finalize the micro flash updates to each
CSB Node.
        After the update is complete, you will be returned to the
SCM CLI
last flash packet to 30                    |
last flash packet to 31                    |
last flash packet to 32                    |                       ⓭
last flash packet to 33                    |
last flash packet to 10                    |
last flash packet to 11                    |

     Please wait.  System will complete power off in roughly 10
seconds

Local escape sequence verified
SCM_E0>
```

B.7 Verify Command

The verify command compares the loaded image with the image on the update media and reports any differences.

Example B–9 Verify Command

```
UPD> v fwa0      ❶

WARNING: verifies may take several minutes to complete for each device.

                          DO NOT ABORT!

Fwa0              Verifying 3.10...  PASSED.   ❷

UPD>
```

❶ The user issues the verify command for **fwa0**.

❷ LFU compares the two images and issues a message saying the verification has passed.

Index

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

W