hp AlphaServer **ES47/ES80/GS1280**
Server Management

SRM Console Reference

Version 1.0

This document provides a reference for the SRM console commands.

December 2002

Table Of Contents

SRM Command Description Conventions

Convention	Meaning
`fixed-font`	ASM 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.*

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 and 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.
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.

SRM Command Notation Formats

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.
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.

Device Naming Conventions

The table below explains the meaning of the device name **dqa0.0.0.15.0**.

	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	Bus/slot number	The logical PCI slot number.			
0	Hose number	The hose number assigned by the firmware.			

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>>>
```

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
```

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
 --------    --------   ------   ------ --------- -------------   ----------
```

This operator also is used to run a command on another CPU. The syntax is **&P*n***, where *n* is the ID of the target CPU.

Comment (#)

A comment can be introduced using the # symbol. The entire text following the # and before Return is ignored.

Example

```
P00>>>#this is a
comment
P00>>>
```

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.

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.

In the examples, 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, issue the server management CLI **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.

Example

```
P00>>>edit nvram
editing ‘nvram’
0 bytes read in
*10 set mopv3_boot 1
*^Z
17 bytes written out to
nvram
P00>>>cat nvram
set mopv3_boot 1
P00>>>
```

SRM Command Language Environment Variables

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 power cycles.

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.

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.
boot_reset	Nonvolatile	A boot reset will occur before booting.
console	Nonvolatile	Defines the type of console device. **serial** A serial console terminal **graphics** A graphics console device.
d_complete	Volatile	Specifies whether or not to display test completion messages. **off** (default) Disables completion messages **on** Enables completion messages
d_eop	Volatile	Specifies whether or not to display test end of pass messages. **off** (default) Disables end of pass messages **on** Enables end of pass messages
d_harderr	Volatile	Determines action taken following a hard error. Values are **halt** (default) and **continue**. Applies only when using **test**.

Continued on next page.

Continued from previous page.

Variable	Attribute	Function
d_passes	Volatile	Specifies the default number of passes for a test to execute. Can be overridden on the test command line. Default value is 1.
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_startup	Volatile	Specifies whether or not to display test startup messages. **off** (default) Disables startup messages **on** Enables startup messages
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**.
e*0_loop_count	Nonvolatile	Specifies number of times message is looped for a **test** command exercising a PCI network adapter.
e*0_loop_inc	Nonvolatile	Specifies the amount the message size is increased from message to message.
e*0_loop_patt	Nonvolatile	Specifies data pattern used for loopback. 0xfffff All the patterns fff 1 All 0's 2 All 1's 3 All A's 4 Incrementing 5 Decrementing
e*0_loop_size	Nonvolatile	Size of loop data used.
e*0_lp_msg_node	Nonvolatile	Number of messages originally sent to each node.
e*0__mode	Nonvolatile	Value for the Ethernet port node when it is started. Allowed values are:
os_type		Auto-sensing BNC AUI FastFD (full duplex) Twisted-pair Auto-negotiate Full duplex, twisted pair
os_type	Nonvolatile	Used to store operating system type. Values are **vms**, **openvms**, **osf**, and **unix**.

SRM Command Summary

Command	Function
bash	Exerciser for the CPU interprocessor ports.
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.
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.
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.
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.
set *envar*	Sets the value of an environment variable.

Continued on next page.

Continued from previous page.

Command	Function
show bios	Displays the devices in the system that have BIOS extension ROMs.
show config	Displays the configuration at the last system initialization.
show cpu	Displays processor information.
show device	Displays the controllers and bootable devices in the system.
show *envar*	Displays the state of all or a specified environment variable.
show fru	Displays the configuration of field-replaceable units (FRUs).
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).

bash

Exercisor for the CPU interprocessor ports. It is designed to saturate the N,S,E, and W port on each CPU in an AlphaServer ES47/ES80/GS1280 Platform by simultaneously moving data from one region of memory on a remote CPU to another. By default, the exerciser tests all ports of each CPU.

Syntax

bash [**-i** <iterations>] [**-s** <memory block size>] [**-n** <neighbor_cpu:master_cpu>]

Options

-i <iterations	Number of times to copy memory block across a link, in thousands. Defaults to 16 (16,000).
-s <memory_block_size>	Size of memory block to be copied across a link, in megabytes. Defaults to 1024 (1Gb).
-n <neighbor_cpu:master_cpu>	To exercise a specific link. <master_cpu> specifies the id of the CPU that copies the memory block targeting the CPU whose id is specified by <neighbor_cpu>.

Arguments

None

Example

```
P00>>>bash
 bash: Number of IP exerciser processes = 32
 P00>>>
```

boot

Boots the supported 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. **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.
-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.

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.

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

```
P00>>>b -fl a dka0
 (boot dka0.0.0.2002.0 -flags a)
 block 0 of dka0.0.0.2002.0 is a valid boot block
 reading 14 blocks from dka0.0.0.2002.0
 bootstrap code read in
 base = b6a000, image_start = 0, image_bytes = 1c00(7168)
 initializing HWRPB at 10000
 GCT base = 552000
 initializing page table at b58000
 initializing machine state
 setting affinity to the primary CPU
 jumping to bootstrap code
UNIX boot - Wednesday November 28, 2001
```

```
Loading vmunix ...
 Loading text at 0xffffffff00000000
 Loading data at 0xffffffff00800000
Sizes:
 text =  8344960
 data =  1937856
 bss  =  2323248
 Starting at 0xffffffff00012cd0
```

cat

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.
-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

Displaying the event log on the console device.

```
P00>>> cat el
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
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>>>
```

clear

Clears the SRM password or an environment variable.

Syntax

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

Options

None

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>>>set foo bar
 environment variable foo created
P00>>>show foo
 foo                    bar
P00>>>clear foo
P00>>>show foo
P00>>>
```

continue

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]

Options

None

Arguments

None

Example

```
$ show time
 16-AUG-2002 16:32:10
 $
 halted CPU 0
 CPU 1 is not halted
 CPU 2 is not halted
 CPU 3 is not halted
halt code = 1
 operator initiated halt
 PC = ffffffffa17d3c20
 P00>>>cont
continuing CPU 0
$ show time
 16-AUG-2002 16:32:17
 $
```

crash

Causes the operating system to be restarted and generates a memory dump.

Syntax

cra[sh]

Options

None

Arguments

None

Example

```
$
 halted CPU 0
 CPU 1 is not halted
 CPU 2 is not halted
 CPU 3 is not halted
halt code = 1
 operator initiated halt
 PC = ffffffffa17d3c20
 P00>>>crash
CPU 0 restarting
%BUGCHECK-I-DIAGNOSTICS, Bugcheck diagnostic messages enabled
 %BUGCHECK-I-DUMPSTYLE, SYSGEN parameter DUMPSTYLE is 00000409
 %BUGCHECK-I-BOOTED_DEV, booted device is "SCSI 0 2002 0 1 100 0 0"
 %BUGCHECK-I-EMPTYDUMPDEV, DUMP_DEV environment variable is empty
 %BUGCHECK-I-INTOPRIBUGCHK, into PrimaryBugCheck for error logs
 %BUGCHECK-I-XDELTA, checking XDELTA
 %BUGCHECK-I-REBOOT, checking for automatic reboot
 %BUGCHECK-I-SETHALT, setting halt request code
 %BUGCHECK-I-SAVESTATE, allowing crash CPU to save state
 %BUGCHECK-I-NOTRECURSIVE, not a recursive bugcheck
 %BUGCHECK-I-BUGCHECKCODE, code = 0000064C
**** OpenVMS (TM) Alpha Operating System V7.3     - BUGCHECK ****
%BUGCHECK-I-INTOINITBCB, into InitBootControlBlock
 %BUGCHECK-I-CONSOPENING, opening channel to device "SCSI 0 2002 0 1 100 0 0"
```

csr

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*]]

Options

None

Arguments

name Name of the CSR register to be displayed, and if data is supplied, deposited to and then displayed. Wildcarding is permissible. 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*
 CSR Name                              CSR Address   CSR Data
 ------------------------------------  -----------   ----------------
 PID0.EV7.RBOX_SCRATCH1                fffffc000b0   0000000000000000
 PID0.IO7.Port0.POx_SCRATCH            fffff800600   0000000000000000
 PID0.IO7.Port1.POx_SCRATCH            ffeff800600   0000000000000000
 PID0.IO7.Port2.POx_SCRATCH            ffdff800600   0000000000000000
 PID0.IO7.Port3.POx_SCRATCH            ffcff800600   0000000000000000
 PID0.IO7.Port7.PO7_SCRATCH            ff8ffb00600   0000000000000000
 PID1.EV7.RBOX_SCRATCH1                ff7ffc000b0   0000000000000000
 PID2.EV7.RBOX_SCRATCH1                fefffc000b0   0000000000000000
 PID3.EV7.RBOX_SCRATCH1                fe7ffc000b0   0000000000000000
P00>>>
```

deposit

Stores data in an address that you specify: a memory location, a register, a device, or a file.

Syntax

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

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

The examine command requests the display of 11 (hexadecimal) quadwords of physical memory beginning at location 0 and incrementing this address 10 (hexadecimal) times. The value a5a5a5a5 has been stored in the first four memory locations, as the display shows.

```
P00>>> deposit -q -p -n 3 0 a5a5a5a5
P00>>> examine -q -p -n 10 0
pmem:                0 00000000A5A5A5A5
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>>>
```

edit

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*

Options

None

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 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 foo
 editing ‘foo’
 16 bytes read in
 *10 echo hello world
 *^Z
 17 bytes written out to foo
 P00>>>cat foo
 echo hello world
P00>>>
```

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.

examine

Displays data in an address that you specify: a memory location, a register, a device, or a file.

Syntax

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

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

The examine command requests the display of 11 (hexadecimal) quadwords of physical memory beginning at location 0 and incrementing this address 10 (hexadecimal) times. The value a5a5a5a5 has been stored in the first four memory locations, as the display shows.

```
P00>>> deposit -q -p -n 3 0 a5a5a5a5
P00>>> examine -q -p -n 10 0
pmem:                0 00000000A5A5A5A5
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>>>
```

exer

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

Example descriptions (in sequence):

1. 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
2. Read block number 0 and 1 from device dkb0.
3. 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.
4. 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:
    - 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.
    - Write a packet of hex 5As from buffer1 to the current block address.

- Set the current block address to what it was just prior to the previous write operation.
- From the current block address, read a packet into buffer2.
- Compare buffer1 with buffer2 and report any discrepancies.
- Repeats steps 1 through 5 until enough packets have been written to satisfy the length requirement of 101 blocks.

6. A nondestructive write test with packet size of 512 bytes. The action string specifies the following sequence of operations:
    - Set the current block address to a random block number on the disk.
    - From the current block address on the disk, read a packet into buffer1.
    - Set the current block address to the device address where it was just before the previous read operation occurred.
    - Write a packet of hex 5As from buffer1 to the current block address.
    - Set the current block address to what it was just prior to the previous write operation.
    - From the current block address on the disk, read a packet into buffer2.
    - Compare buffer1 with buffer2 and report any discrepancies.
    - Repeat the above steps until each block on the disk has been written once and read twice.

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

grep

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 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>>>show config | grep EV7
 NS,EW (0,0)    Hard ID 0       1.50 MB Cache          EV7 rev 2.0, 800 MHz
 NS,EW (1,0)    Hard ID 1       1.50 MB Cache          EV7 rev 2.0, 800 MHz
 NS,EW (0,1)    Hard ID 2       1.50 MB Cache          EV7 rev 2.0, 800 MHz
 NS,EW (1,1)    Hard ID 3       1.50 MB Cache          EV7 rev 2.0, 800 MHz
```

halt

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

```
$ ^P
halted CPU 0
CPU 1 is not halted
CPU 2 is not halted
CPU 3 is not halted

halt code = 1
operator initiated
halt
PC = ffffffffdca86980
P00>>>halt 1
P00>>>halt 2
P00>>>halt 3
P00>>>
```

help (or man)

Provides basic information on the console commands.

Syntax

help [*command*]

Options

None

Argument

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

Example

```
P00>>>help
 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.
              {a,b,c} Implies any one of a, b, c.
              {a|b|c} Implies any combination of a, b, c.
The following help topics are available:
alloc           assign_hw       bash            boot            bpt
 brcm570_seeprom break           buildfru        call            cat
 check           chmod           chown           clear           clear_error
 cmp             continue        crash           csr             debug1
 deposit         dynamic         echo            edit            eval
 examine         exer            exit            fakedisk        false
 fill_in_ctb_ws_ find_field      fptest          free            gct
 gctverify       grep            halt            hd              help
 info            initialize      isp1020_edit    kill            kill_bash
 kill_diags      line            lpinit          ls              man
 mc_cable        mc_diag         memexer         memexer_mp      memtest
 migrate         more            net             nettest         php_button_test
 php_led_test    prcache         ps              rm              run
 sa              semaphore       set             set host        shell
 show            show bios       show cluster    show fru        show hwrpb
 show iobq       show map        show_status     sleep           sp
 start           stop            true            uptime          wc
 wwidmgr
```

info

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*]

Options

None

Argument

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

- 0 Displays the SRM Memory Descriptors as described in the *Alpha System Reference Manual.*
- 1 Reserved.
- 2 Dumps the FRU table.
- 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 Displays machine check logout frame data.

Example

```
P00>>>info
   0. HWRPB MEMDSC
   1. Console PTE
   2. GCT/FRU 5
   3. Dump System CSRs
   4. IMPURE area (abbreviated)
   5. IMPURE area (full)
   6. LOGOUT area
   7. Dump Error Log
   8. Clear Error Log
 Enter selection: 0
HWRPB: 10000    MEMDSC:1a340   Cluster count: 8
Cluster: 0, Usage: Console
 START_PFN: 00000000  PFN_COUNT: 000005b5 PFN_TESTED: 000005b5
 1461 pages from 0000000000000000 to 0000000000b69fff
Cluster: 1, Usage: System
 START_PFN: 000005b5  PFN_COUNT: 0001fa4b PFN_TESTED: 0001fa4b
 BITMAP_VA: 0000000000000000 BITMAP_PA: 0000000000b52000
 129611 good pages from 0000000000b6a000 to 000000003fffffff
Cluster: 2, Usage: Console
 START_PFN: 00200000  PFN_COUNT: 0000003a PFN_TESTED: 0000003a
 58 pages from 0000000400000000 to 0000000400073fff
Cluster: 3, Usage: System
 START_PFN: 0020003a  PFN_COUNT: 0001ffc6 PFN_TESTED: 0001ffc6
 BITMAP_VA: 0000000000000000 BITMAP_PA: 0000000400070000
 131014 good pages from 0000000400074000 to 000000043fffffff
Cluster: 4, Usage: Console
 START_PFN: 00400000  PFN_COUNT: 0000003a PFN_TESTED: 0000003a
 58 pages from 0000000800000000 to 0000000800073fff
```

```
Cluster: 5, Usage: System
 START_PFN: 0040003a  PFN_COUNT: 0001ffc6 PFN_TESTED: 0001ffc6
 BITMAP_VA: 0000000000000000 BITMAP_PA: 0000000800070000
 131014 good pages from 0000000800074000 to 000000083fffffff
Cluster: 6, Usage: Console
 START_PFN: 00600000  PFN_COUNT: 0000003a PFN_TESTED: 0000003a
 58 pages from 0000000c00000000 to 0000000c00073fff
Cluster: 7, Usage: System
 START_PFN: 0060003a  PFN_COUNT: 0001ffc6 PFN_TESTED: 0001ffc6
 BITMAP_VA: 0000000000000000 BITMAP_PA: 0000000c00070000
 131014 good pages from 0000000c00074000 to 0000000c3fffffff
P00>>>
```

init

Resets the SRM console firmware, incorporating any changes made to environment variables during the foregoing console session, and reinitializes the hardware.

Syntax

init

Options

None

Arguments

None

Example

```
P00>>>init
starting console on CPU 0
initialized idle PCB
initializing semaphores
initializing heap
initial heap 700c0
memory low limit = 54a000 heap = 700c0, 1fffc0
initializing driver structures
initializing idle process PID
initializing file system
initializing timer data structures
lowering IPL
CPU 0 speed is 533 MHz
create dead_eater
create poll
create timer
create powerup
entering idle loop
access NVRAM
Get Partition DB
hpcount = 1, spcount = 2, ev7_count = 8, io7_count = 1
hard_partition = 0
IO7-100 (Pass 2) at PID 0
IO7 North port speed is 133 MHz
Hose 0 - 33 MHz PCI
Hose 1 - 66 MHz PCI
Hose 2 - 33 MHz PCI
Hose 3 - 2X AGP
0 sub-partition 0:   start:00000000 00000000   size:00000000 40000000
PID 0 console memory base: 0, 1 GB
1 sub-partition 0:   start:00000004 00000000   size:00000000 40000000
PID 1 memory: 400000000, 1 GB
2 sub-partition 0:   start:00000008 00000000   size:00000000 40000000
PID 2 memory: 800000000, 1 GB
3 sub-partition 0:   start:0000000c 00000000   size:00000000 40000000
```

```
PID 3 memory: c00000000, 1 GB
4 sub-partition 0:   start:00000020 00000000   size:00000000 40000000
PID 4 memory: 2000000000, 1 GB
5 sub-partition 0:   start:00000024 00000000   size:00000000 40000000
PID 5 memory: 2400000000, 1 GB
6 sub-partition 0:   start:00000028 00000000   size:00000000 40000000
PID 6 memory: 2800000000, 1 GB
7 sub-partition 0:   start:0000002c 00000000   size:00000000 40000000
PID 7 memory: 2c00000000, 1 GB
total memory, 8 GB
probe I/O subsystem
probing hose 0, PCI
probing PCI-to-PCI bridge, hose 0 bus 2
do not use secondary IDE channel on CMD controller
probing PCI-to-PCI bridge, hose 0 bus 3
bus 2, slot 0, function 0 -- usba -- USB
bus 2, slot 0, function 1 -- usbb -- USB
bus 2, slot 0, function 2 -- usbc -- USB
bus 2, slot 0, function 3 -- usbd -- USB
bus 2, slot 1 -- dqa -- CMD 649 PCI-IDE
bus 2, slot 2 -- pka -- Adaptec AIC-7892
bus 3, slot 0 -- fwa -- DEC PCI FDDI
probing hose 1, PCI
probing hose 2, PCI
probing PCI-to-PCI bridge, hose 2 bus 2
bus 0, slot 1, function 0 -- pkb -- Adaptec AIC-7899
bus 0, slot 1, function 1 -- pkc -- Adaptec AIC-7899
bus 2, slot 4 -- eia -- DE602-AA
bus 2, slot 5 -- eib -- DE602-AA
bus 0, slot 3 -- pga -- KGPSA-C
probing hose 3, PCI
bus 0, slot 5 -- vga -- 3D Labs OXYGEN VX1 AGP
starting drivers
Starting secondary CPU 1 at address 400030000
Starting secondary CPU 2 at address 800030000
Starting secondary CPU 3 at address c00030000
Starting secondary CPU 4 at address 2000030000
Starting secondary CPU 5 at address 2400030000
Starting secondary CPU 6 at address 2800030000
Starting secondary CPU 7 at address 2c00030000
initializing GCT/FRUinitializing keyboard
..... at 54a000
Initializing fwa dqa eia eib
*** Error (eib0.0.0.2005.2), No link, Auto Negotiation did not
complete.
pka pkb pkc pga pga0.0.0.3.2 - Nvram read failed.

AlphaServer Console T6.4-3, built on Dec  5 2002 at 14:21:43
```

```
P00>>>

P00>>>show dev
dka0.0.0.2002.0            DKA0              COMPAQ BD0366349C  3B06
dka100.1.0.2002.0          DKA100            COMPAQ BD0366349C  3B06
eia0.0.0.2004.2            EIA0              00-02-A5-89-9D-36
eib0.0.0.2005.2            EIB0              00-02-A5-89-9D-37
fwa0.0.0.3000.0            FWA0              08-00-2B-B9-1B-7D
pga0.0.0.3.2               PGA0        WWN 1000-0000-c929-4dbc
pka0.7.0.2002.0            PKA0                 SCSI Bus ID 7
pkb0.7.0.1.2               PKB0                 SCSI Bus ID 7
pkc0.7.0.101.2             PKC0                 SCSI Bus ID 7
P00>>>set bootdef_dev dka02
device dka02 is invalid
P00>>>set bootdef_dev dka0.0.0.2002.0
P00>>>
```

kill

Kills 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*

Options

None

Arguments

None

Example

1. The user types the show_status command to show the status of any background processes. Process 123 is shown as a memory exerciser.
2. The user issues the kill 123 command to terminate the execution of the memory exerciser.
3. The show_status command confirms that the memory exerciser is no longer running.

```
P00>>> show_status
 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
P00>>> show_status
 ID       Program      Device       Pass  Hard/Soft Bytes Written  Bytes Read
-------- ------------ ------------ ------ --------- ------------- -------------
00000001         idle system            0    0    0             0             0
P00>>>
```

kill_diags

Stops all console-based diagnostic processes running on the system.

Syntax

kill_diags

Options

None

Arguments

None

Example

```
P00>>>show_status
 ID       Program      Device       Pass  Hard/Soft Bytes Written  Bytes Read
 -------- ------------ ------------ ------ --------- ------------- -------------
 00000001         idle system            0    0    0             0             0
 000001ed      memtest memory            9    0    0    8398831616    8398831616
 00000206      memtest memory           10    0    0    9659400192    9659400192
 0000021f      memtest memory           10    0    0    9659400192    9659400192
 00000228      memtest memory            9    0    0    8586133504    8586133504
P00>>>kill_diags
P00>>>show_status
 ID       Program      Device       Pass  Hard/Soft Bytes Written  Bytes Read
 -------- ------------ ------------ ------ --------- ------------- -------------
 00000001         idle system            0    0    0             0             0
P00>>>
```

ls

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

List all files that begin with "d" in long format.

```
P00>>>ls -l d*
 r-xb  rd                  0/0               3c9bb0     0    d
 r-xb  rd                  0/0               3c7380     0    debug1
 r---  decode              0/0                    0     0    decode
 r-xb  rd                  0/0               3c9bb0     0    deposit
 r---  dk                  0/0                    0     0
dka0.0.0.2002.0
 r---  dk                  0/0                    0     0
dka100.1.0.2002.0
 r-xb  rd                  0/0               3c3980     0    dynamic
```

memexer

Tests the memory on thesystem, 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*]

Options

None

Argument

n Specifies the number of memory test processes to start. The default is l.

Example

```
P00>>>memexer
 memtest -bs 1c0000 -rb -p 0 &
 memtest -sa 400074000 -ea 440000000 -z -p 0 &
 memtest -sa 800074000 -ea 840000000 -z -p 0 &
 memtest -sa C00074000 -ea C40000000 -z -p 0 &
P00>>>
```

memexer_mp

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

Options

None

Arguments

None

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 -t 1 -sa B64020 -i 8 -l 1C0000 -p 0 -z &p0 &
 memtest -t 1 -sa B64024 -i 8 -l 1C0000 -p 0 -z &p1 &
 memtest -t 1 -sa B64028 -i 8 -l 1C0000 -p 0 -z &p2 &
 memtest -t 1 -sa B6402C -i 8 -l 1C0000 -p 0 -z &p3 &
 memtest -t 1 -sa B64030 -i 8 -l 1C0000 -p 0 -z &p4 &
 memtest -t 1 -sa B64034 -i 8 -l 1C0000 -p 0 -z &p5 &
 memtest -t 1 -sa B64038 -i 8 -l 1C0000 -p 0 -z &p6 &
 memtest -t 1 -sa B6403C -i 8 -l 1C0000 -p 0 -z &p7 &
 P00>>>
```

migrate

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.

Arguments

None

Example

Migrate CPU 2 to partition 1.

```
P00>>>migrate -cpu 2 -partition 1
 migrating CPU 2 to partition 1
 P00>>>
```

more

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>>>show config | more
                          Compaq Computer Corporation
                       hp AlphaServer GS1280 7/800
SRM Console     X6.3-9195, built on Aug 16 2002 at 13:59:21
 PALcode         OpenVMS PALcode X2.11-0, Tru64 UNIX PALcode X2.08-0
PID 0           CPU 0           Cabinet 0  Drawer 0
 NS,EW (0,0)    Hard ID 0       1.50 MB Cache           EV7 rev 2.0, 800 MHz
 Memory 0                       1 GB
 IO7 0                          3.3V PCI-X I/O          IO7 pass 1
 I/O Drawer 1                  Cabinet 0  Riser 0      Backplane rev 0
 PCI Bus 0    Hose 0           64 Bit, 33 MHz          PCI 2.2 mode
 PCI Bus 1    Hose 1           64 Bit, 33 MHz          PCI 2.2 mode
 PCI Bus 2    Hose 2           64 Bit, 66 MHz          PCI 2.2 mode
 AGP Bus 3    Hose 3           AGP rev 2.0             AGP 2x mode
PID 1           CPU 1           Cabinet 0  Drawer 0
 NS,EW (1,0)    Hard ID 1       1.50 MB Cache           EV7 rev 2.0, 800 MHz
 Memory 1                       1 GB
 No Local I/O
PID 2           CPU 2           Cabinet 0  Drawer 0
 --More-- (SPACE - next page, ENTER - next line, Q - quit)
```

nettest

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.

Example

Nettest eia0, do not change the mode, use file lp_nodes_eia0, one pass

```
P00>>>nettest eia0 -mode nc -f lp_nodes_eia0 -p 1
 P00>>>
```

power

Removes power from all hard partitions. To prevent catastrophic errors, shut down the operating system before using this command.

Syntax

power off

Options

none

Arguments

none

Example

```
P00>>>power off
```

ps

Displays 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

Options

None

Arguments

None

Example

```
P00>>>ps
 ID       PCB      Pri CPU Time Affinity CPU  Program    State
 -------- -------- --- -------- -------- --- ---------- ------------------------
 0000011a 0024d2a0 3          0 80000000 0          ps running
 00000032 00249020 3       3013 80000000 0       shell ready
 0000002a 00246aa0 6          0 80000000 0   pgb0__poll waiting on kgppollwake
 00000029 00245640 6          0 80000000 0   pgb0_fcint waiting on kgpfcwake
 00000028 00242ec0 6          0 80000000 0   pga0__poll waiting on kgppollwake
 00000027 00241ca0 6          0 80000000 0   pga0_fcint waiting on kgpfcwake
 0000001d 00232160 5          0 80000000 0      rx_eib0 waiting on rx_isr_eib0
 0000001a 0022aa40 5          0 80000000 0      rx_eia0 waiting on rx_isr_eia0
 00000018 0021ac80 5          1 ffffffff 0      rx_fwa0 waiting on tqe 3403d0
 00000016 0021e340 3          1 80000003 3      shell_3 ready
 00000014 00435960 0     133934 80000003 3         idle running
 00000013 002171e0 3       2547 80000000 0      shell_0 waiting on rxq_ready
 00000012 00213240 3          2 80000002 2      shell_2 ready
 00000010 00434770 0     133954 80000002 2         idle running
 0000000f 0020cb60 3          1 80000001 1      shell_1 ready
 0000000d 00433580 0     133946 80000001 1         idle running
 0000000c 001e3ea0 5          0 80000000 0     dup_poll waiting on tqe 332154
 0000000b 001e2a20 5          0 80000000 0    mscp_poll waiting on tqe 330468
 0000000a 001da8c0 5          2 80000000 0      usbd_cb waiting on usb callback
 00000009 001d61e0 5          3 80000000 0      usbc_cb waiting on usb callback
 00000008 001d2020 5         22 80000000 0      usbb_cb waiting on usb callback
 00000007 001b6640 5          3 80000000 0      usba_cb waiting on usb callback
 00000006 001aa1a0 6          0 ffffffff 0   tt_control waiting on tt_control
 00000004 000786a0 7          0 ffffffff 0        timer waiting on timer
 00000003 00077240 2     100025 ffffffff 0         poll ready
 00000002 00076020 6          0 ffffffff 0   dead_eater waiting on dead_beef
 00000001 00432390 0      28230 80000000 0         idle ready
P00>>>
```

rm

Removes the named file(s) from the file system.

Syntax

rm *file…*

Options

None

Argument

file.. The name of the file(s) to be removed.

Example

```
P00>>>echo echo hello world >foo
P00>>>cat foo
 echo hello world
P00>>>ls foo
 foo
P00>>>foo
 hello world
P00>>>rm foo
P00>>>ls foo
 foo no such file
P00>>>
```

set *envar*

Sets or modifies the value of an environment variable.

Syntax

se[t] *envar* [*value*]

Options

None

Argument

envar [*value*] Environment variables and their values.

Example

```
P00>>>set bootdef_dev dka02
device dka02 is invalid
P00>>>set bootdef_dev dka0.0.0.2002.0
P00>>>
```

show bios

Displays 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.

Options

None

Arguments

None

Example

```
P00>>>show bios
resetting all I/O buses

pkb0.7.0.1.2 - Adaptec AIC-7899
pkc0.7.0.101.2 - Adaptec AIC-7899
pya0.0.0.2.2 - CPQ SmartArray 5300
vga0.0.0.5.3 - 3D Labs OXYGEN VX1 AGP
P00>>>
```

show configuration

Displays the configuration seen at the last system initialization.

Syntax

sh[ow] c[onfiguration]

Options

None

Arguments

None

Example

```
P00>>>show config
                        Compaq Computer Corporation
                     hp AlphaServer GS1280 7/800
SRM Console     X6.3-9195, built on Aug 16 2002 at 13:59:21
 PALcode         OpenVMS PALcode X2.11-0, Tru64 UNIX PALcode X2.08-0
PID 0           CPU 0           Cabinet 0  Drawer 0
 NS,EW (0,0)    Hard ID 0       1.50 MB Cache           EV7 rev 2.0, 800 MHz
 Memory 0                       1 GB
 IO7 0                          3.3V PCI-X I/O          IO7 pass 1
 I/O Drawer 1                  Cabinet 0  Riser 0      Backplane rev 0
 PCI Bus 0    Hose 0           64 Bit, 33 MHz          PCI 2.2 mode
 PCI Bus 1    Hose 1           64 Bit, 33 MHz          PCI 2.2 mode
 PCI Bus 2    Hose 2           64 Bit, 66 MHz          PCI 2.2 mode
 AGP Bus 3    Hose 3           AGP rev 2.0             AGP 2x mode
PID 1           CPU 1           Cabinet 0  Drawer 0
 NS,EW (1,0)    Hard ID 1       1.50 MB Cache           EV7 rev 2.0, 800 MHz
 Memory 1                       1 GB
 No Local I/O
PID 2           CPU 2           Cabinet 0  Drawer 0
 NS,EW (0,1)    Hard ID 2       1.50 MB Cache           EV7 rev 2.0, 800 MHz
 Memory 2                       1 GB
 No Local I/O
PID 3           CPU 3           Cabinet 0  Drawer 0
 NS,EW (1,1)    Hard ID 3       1.50 MB Cache           EV7 rev 2.0, 800 MHz
 Memory 3                       1 GB
 No Local I/O
System Memory 4 GB
                       RIMMs
 PID   Cab Drw CPU  0123456789     Size     Address
 0     0   0   0   PPPP.PPPP.     1 GB           0   Non-Striped
 1     0   0   1   PPPP.PPPP.     1 GB    400000000   Non-Striped
 2     0   0   2   PPPP.PPPP.     1 GB    800000000   Non-Striped
 3     0   0   3   PPPP.PPPP.     1 GB    c00000000   Non-Striped
 Slot   Option                  Hose 0, Bus 0, PCI
 1    DECchip 21154-AA                              Bridge to Bus 2, PCI
 2    DECchip 21154-AA                              Bridge to Bus 3, PCI
 Slot   Option                  Hose 0, Bus 2, PCI
 0/0  USB                     usba0.0.0.2000.0       hub
 0/1  USB                     usbb0.0.0.2100.0       hub/mouse
 0/2  USB                     usbc0.0.0.2200.0       hub
 0/3  USB                     usbd0.0.0.2300.0       hub
 1    CMD 649 PCI-IDE         dqa.0.0.2001.0
 2    Adaptec AIC-7892        pka0.7.0.2002.0         SCSI Bus ID 7
                            dka0.0.0.2002.0         COMPAQ BD0366349C
                            dka100.1.0.2002.0       COMPAQ BD036635C5
```

```
 Slot   Option                Hose 0, Bus 3, PCI
 1    DEC PCI FDDI           fwa0.0.0.3001.0         08-00-2B-B9-1B-7D
 Slot   Option                Hose 1, Bus 0, PCI
 1    KGPSA-C                pga0.0.0.1.1            WWN 1000-0000-c929-4dbc
 2    DECchip 21154-AA                               Bridge to Bus 2, PCI
 Slot   Option                Hose 1, Bus 2, PCI
 4    DE602-AA               eia0.0.0.2004.1         00-02-A5-89-9D-36
 5    DE602-AA               eib0.0.0.2005.1         00-02-A5-89-9D-37
 Slot   Option                Hose 2, Bus 0, PCI
 1/0  Adaptec AIC-7899       pkb0.7.0.1.2            SCSI Bus ID 7
 1/1  Adaptec AIC-7899       pkc0.7.0.101.2          SCSI Bus ID 7
 2    FCA-2354               pgb0.0.0.2.2            WWN 1000-0000-c927-2ebd
 Slot   Option                Hose 3, Bus 0, AGP
 5    3D Labs OXYGEN VX1 A   vga0.0.0.5.3
P00>>>
```

show cpu

Displays processor information.

Syntax

sh[ow] cpu [*dev_name*]

Options

None

Argument

None

Example

```
P00>>>show cpu
CPU 0   CurOwner 0  Owner 0   Type Major 15, Minor  2
CPU 1   CurOwner 0  Owner 0   Type Major 15, Minor  2
CPU 2   CurOwner 0  Owner 0   Type Major 15, Minor  2
CPU 3   CurOwner 0  Owner 0   Type Major 15, Minor  2
CPU 0   CurOwner 0  Owner 0   Type Major 15, Minor  2
CPU 1   CurOwner 0  Owner 0   Type Major 15, Minor  2
CPU 2   CurOwner 0  Owner 0   Type Major 15, Minor  2
CPU 3   CurOwner 0  Owner 0   Type Major 15, Minor  2
P00>>>
```

show device

Displays information for devices on the system.

Syntax

sh[ow] dev[ice] [*dev_name*]

Options

None

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>>>show dev
 dka0.0.0.2002.0            DKA0              COMPAQ BD0366349C  3B06
 dka100.1.0.2002.0          DKA100            COMPAQ BD036635C5  B017
 eia0.0.0.2004.1            EIA0              00-02-A5-89-9D-36
 eib0.0.0.2005.1            EIB0              00-02-A5-89-9D-37
 fwa0.0.0.3001.0            FWA0              08-00-2B-B9-1B-7D
 pga0.0.0.1.1               PGA0        WWN 1000-0000-c929-4dbc
 pgb0.0.0.2.2               PGB0        WWN 1000-0000-c927-2ebd
 pka0.7.0.2002.0            PKA0                SCSI Bus ID 7
 pkb0.7.0.1.2               PKB0                SCSI Bus ID 7
 pkc0.7.0.101.2             PKC0                SCSI Bus ID 7
P00>>>
```

show *envar*

Displays the current state of the specified environment variable.

Syntax

sh[ow] *envar*
or
sh[ow] *

Options

None

Arguments

envar	An environment variable name. 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

Show the status of all environment variables that begin with "boot."

```
P00>>>show boot*
 boot_dev               dka0.0.0.2002.0
 boot_file
 boot_osflags
 boot_reset             OFF
 bootdef_dev            dka0.0.0.2002.0
 booted_dev
 booted_file
 booted_osflags
P00>>>
```

show fru

Displays the physical configuration of field replaceable units (FRUs).

Syntax

sh[**ow**] **fru**

Options

None

Arguments

None

FRU Acronyms Used In Display

CAB	System, I/O, or Power cabinet
DRW	System, I/O, or Power drawer
COCP	Cabinet operator control panel
DOCP	Drawer operator control panel
SBB	System Building Block backplane
DUO	Dual CPU Module
CPU	Alpha CPU chip
CMM	CPU Module Manager module
RIMM	Rambus Memory Module
MBM	Backplane Manager module
FAN	System or PCI box blower
PS	Individual DC power supplies
PWR	Main power module
PCI	PCI I/O backplane
IOR	I/O Riser module
SLOT	individual PCI module
PBM	PCI backplane manager module
AGP	Individual AGP module

Example

```
P00>>>show fru
Fru Name                  E Part #             Serial #         Model/Other
 CAB0                     00 -                  -                -
 CAB0.COCP                00 cab                srm_8p           -
 CAB0.DRW0                00 -                  -                -
 CAB0.DRW0.DOCP           00 ???????????????    ??????????       -
 CAB0.DRW0.SBB            00 -                  -                -
 CAB0.DRW0.DUO0           00 54-30252-04.A01    AY14605499       -
 CAB0.DRW0.DUO0.CMM       00 ???????????????    ??????????       -
 CAB0.DRW0.DUO0.CPU0      00 -                  -                -
 CAB0.DRW0.DUO0.RIMM00    00 20-1C872-01        SRM_8P000L       -
 CAB0.DRW0.DUO0.RIMM10    00 20-1C872-01        SRM_8P001L       -
 CAB0.DRW0.DUO0.RIMM20    00 20-1C872-01        SRM_8P002L       -
 CAB0.DRW0.DUO0.RIMM30    00 20-1C872-01        SRM_8P003L       -
 CAB0.DRW0.DUO0.RIMM50    00 20-1C872-01        SRM_8P005L       -
 CAB0.DRW0.DUO0.RIMM60    00 20-1C872-01        SRM_8P006L       -
 CAB0.DRW0.DUO0.RIMM70    00 20-1C872-01        SRM_8P007L       -
 CAB0.DRW0.DUO0.RIMM80    00 20-1C872-01        SRM_8P008L       -
 CAB0.DRW0.DUO0.CPU1      00 -                  -                -
 CAB0.DRW0.DUO0.RIMM01    00 20-1C872-01        SRM_8P00AL       -
```

```
CAB0.DRW0.DUO0.RIMM11    00 20-1C872-01        SRM_8P00BL       -
CAB0.DRW0.DUO0.RIMM21    00 20-1C872-01        SRM_8P00CL       -
CAB0.DRW0.DUO0.RIMM31    00 20-1C872-01        SRM_8P00DL       -
CAB0.DRW0.DUO0.RIMM51    00 20-1C872-01        SRM_8P00FL       -
CAB0.DRW0.DUO0.RIMM61    00 20-1C872-01        SRM_8P00GL       -
CAB0.DRW0.DUO0.RIMM71    00 20-1C872-01        SRM_8P00HL       -
CAB0.DRW0.DUO0.RIMM81    00 20-1C872-01        SRM_8P00IL       -
CAB0.DRW0.MBM            00 54-30284-02.D01    SW1280029        -
CAB0.DRW0.DUO1           00 54-30252-03.B02    AY13905275       -
CAB0.DRW0.DUO1.CMM       00 ???????????????    ??????????       -
CAB0.DRW0.DUO1.CPU0      00 -                  -                -
CAB0.DRW0.DUO1.RIMM00    00 20-1C872-01        ??????????       -
CAB0.DRW0.DUO1.RIMM10    00 20-1C872-01        ??????????       -
CAB0.DRW0.DUO1.RIMM20    00 20-1C872-01        ??????????       -
CAB0.DRW0.DUO1.RIMM30    00 20-1C872-01        ??????????       -
CAB0.DRW0.DUO1.RIMM50    00 20-1C872-01        ??????????       -
CAB0.DRW0.DUO1.RIMM60    00 20-1C872-01        ??????????       -
CAB0.DRW0.DUO1.RIMM70    00 20-1C872-01        ??????????       -
CAB0.DRW0.DUO1.RIMM80    00 20-1C872-01        ??????????       -
CAB0.DRW0.DUO1.CPU1      00 -                  -                -
CAB0.DRW0.DUO1.RIMM01    00 20-1C872-01        SRM_8P01AL       -
CAB0.DRW0.DUO1.RIMM11    00 20-1C872-01        SRM_8P01BL       -
CAB0.DRW0.DUO1.RIMM21    00 20-1C872-01        SRM_8P01CL       -
CAB0.DRW0.DUO1.RIMM31    00 20-1C872-01        SRM_8P01DL       -
CAB0.DRW0.DUO1.RIMM51    00 20-1C872-01        SRM_8P01FL       -
CAB0.DRW0.DUO1.RIMM61    00 20-1C872-01        SRM_8P01GL       -
CAB0.DRW0.DUO1.RIMM71    00 20-1C872-01        SRM_8P01HL       -
CAB0.DRW0.DUO1.RIMM81    00 20-1C872-01        SRM_8P01IL       -
CAB0.DRW0.DUO2           00 54-30252-03.B02    AY13905206       -
CAB0.DRW0.DUO2.CMM       00 ???????????????    ??????????       -
CAB0.DRW0.DUO2.CPU0      00 -                  -                -
CAB0.DRW0.DUO2.RIMM00    00 20-1C872-01        SRM_8P020L       -
CAB0.DRW0.DUO2.RIMM10    00 20-1C872-01        SRM_8P021L       -
CAB0.DRW0.DUO2.RIMM20    00 20-1C872-01        SRM_8P022L       -
CAB0.DRW0.DUO2.RIMM30    00 20-1C872-01        SRM_8P023L       -
CAB0.DRW0.DUO2.RIMM50    00 20-1C872-01        SRM_8P025L       -
CAB0.DRW0.DUO2.RIMM60    00 20-1C872-01        SRM_8P026L       -
CAB0.DRW0.DUO2.RIMM70    00 20-1C872-01        SRM_8P027L       -
CAB0.DRW0.DUO2.RIMM80    00 20-1C872-01        SRM_8P028L       -
CAB0.DRW0.DUO2.CPU1      00 -                  -                -
CAB0.DRW0.DUO2.RIMM01    00 20-1C872-01        SRM_8P02AL       -
CAB0.DRW0.DUO2.RIMM11    00 20-1C872-01        SRM_8P02BL       -
CAB0.DRW0.DUO2.RIMM21    00 20-1C872-01        SRM_8P02CL       -
CAB0.DRW0.DUO2.RIMM31    00 20-1C872-01        SRM_8P02DL       -
CAB0.DRW0.DUO2.RIMM51    00 20-1C872-01        SRM_8P02FL       -
CAB0.DRW0.DUO2.RIMM61    00 20-1C872-01        SRM_8P02GL       -
CAB0.DRW0.DUO2.RIMM71    00 20-1C872-01        SRM_8P02HL       -
CAB0.DRW0.DUO2.RIMM81    00 20-1C872-01        SRM_8P02IL       -
CAB0.DRW0.DUO3           00 54-30252-03.B02    AY13905253       -
CAB0.DRW0.DUO3.CMM       00 ???????????????    ??????????       -
CAB0.DRW0.DUO3.CPU0      00 -                  -                -
CAB0.DRW0.DUO3.RIMM00    00 20-1C872-01        SRM_8P030L       -
CAB0.DRW0.DUO3.RIMM10    00 20-1C872-01        SRM_8P031L       -
CAB0.DRW0.DUO3.RIMM20    00 20-1C872-01        SRM_8P032L       -
CAB0.DRW0.DUO3.RIMM30    00 20-1C872-01        SRM_8P033L       -
CAB0.DRW0.DUO3.RIMM50    00 20-1C872-01        SRM_8P035L       -
CAB0.DRW0.DUO3.RIMM60    00 20-1C872-01        SRM_8P036L       -
CAB0.DRW0.DUO3.RIMM70    00 20-1C872-01        SRM_8P037L       -
CAB0.DRW0.DUO3.RIMM80    00 20-1C872-01        SRM_8P038L       -
CAB0.DRW0.DUO3.CPU1      00 -                  -                -
CAB0.DRW0.DUO3.RIMM01    00 20-1C872-01        SRM_8P03AL       -
CAB0.DRW0.DUO3.RIMM11    00 20-1C872-01        SRM_8P03BL       -
CAB0.DRW0.DUO3.RIMM21    00 20-1C872-01        SRM_8P03CL       -
```

```
CAB0.DRW0.DUO3.RIMM31    00 20-1C872-01        SRM_8P03DL       -
CAB0.DRW0.DUO3.RIMM51    00 20-1C872-01        SRM_8P03FL       -
CAB0.DRW0.DUO3.RIMM61    00 20-1C872-01        SRM_8P03GL       -
CAB0.DRW0.DUO3.RIMM71    00 20-1C872-01        SRM_8P03HL       -
CAB0.DRW0.DUO3.RIMM81    00 20-1C872-01        SRM_8P03IL       -
CAB0.DRW0.FAN0           00 -                   -                -
CAB0.DRW0.FAN1           00 -                   -                -
CAB0.DRW0.PSA0           00 -                   -                -
CAB0.DRW0.PSA1           00 -                   -                -
CAB0.DRW0.PSA2           00 -                   -                -
CAB0.DRW0.PWR0           00 -                   -                -
CAB0.DRW8                00 -                   -                -
CAB0.DRW8.DOCP           00 ??????????????     ??????????       -
CAB0.DRW8.PS0            00 30-56245-01.AX03   4I22301617       -
CAB0.DRW8.PS1            00 30-56245-01.AX03   4I22301602       -
CAB0.DRW8.FAN0           00 -                   -                -
CAB0.DRW8.FAN1           00 -                   -                -
CAB0.DRW8.FAN2           00 -                   -                -
CAB0.DRW8.PCI            00 54-30658-01.A01    SW14500037       -
CAB0.DRW8.IOR0           00 ??????????????     ??????????       -
CAB0.DRW8.PCI0.SLOT1     00 -                   -                -
CAB0.DRW8.PCI0.SLOT2     00 -                   -                -
CAB0.DRW8.PCI1.SLOT1     00 -                   -                -
CAB0.DRW8.PCI2.SLOT1     00 -                   -                -
CAB0.DRW8.PCI2.SLOT2     00 -                   -                -
CAB0.DRW8.PCI2.SLOT3     00 -                   -                -
CAB0.DRW8.AGP            00 -                   -                -
CAB0.DRW8.PBM            00 54-30284-01.D01    SW12300013       -
P00>>>
```

show memory

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 RIMMs is given in addition to the board information.

Arguments

None

Example

```
P00>>>show mem
 System Memory 4 GB
                          RIMMs
 PID   Cab Drw CPU  0123456789     Size     Address
 0     0   0   0   PPPP.PPPP.     1 GB            0   Non-Striped
 1     0   0   1   PPPP.PPPP.     1 GB    400000000   Non-Striped
 2     0   0   2   PPPP.PPPP.     1 GB    800000000   Non-Striped
 3     0   0   3   PPPP.PPPP.     1 GB    c00000000   Non-Striped
P00>>>
```

show pal

Displays the versions of *Tru64 UNIX* and *OpenVMS* PALcode.

Syntax

sh[**ow**] **pal**

Options

None

Arguments

None

Example

```
P00>>>show pal
 pal                     OpenVMS PALcode X2.11-0, Tru64 UNIX PALcode X2.08-0
P00>>>
```

show_status

Displays information on system exercisers and diagnostic firmware running in the background.

Syntax

show_status

Options

None

Arguments

None

Example

```
P00>>>show_status
 ID       Program      Device       Pass  Hard/Soft Bytes Written  Bytes Read
 -------- ------------ ------------ ------ --------- ------------- -------------
 00000001         idle system            0    0    0             0             0
 000001ed      memtest memory            2    0    0    1227620352    1227620352
 00000206      memtest memory            2    0    0    1073266688    1073266688
 0000021f      memtest memory            2    0    0    1073266688    1073266688
 00000228      memtest memory            2    0    0    1073266688    1073266688
P00>>>
```

show version

Shows the version of the SRM console firmware code.

Syntax

sh[**ow**] **version**

Options

None

Arguments

None

Example

```
P00>>>show version
 version                X6.3-9195 Aug 16 2002 13:59:21
P00>>>
```

sys_exer

Tests the entire system, including memory, disks, tapes, serial ports, parallel port, network, and VGA.

Syntax

sys_exer [**-t** *runtime*]

Options

None

Arguments

-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
 Default zone extended at the expense of memzone.
 Use INIT before booting
 Exercising the Memory
 memtest -bs 1c0000 -rb -p 0 &
 memtest -bs 1c0000 -rb -p 0 &
 memtest -bs 1c0000 -rb -p 0 &
 memtest -bs 1c0000 -rb -p 0 &
 memtest -sa 400074000 -ea 440000000 -z -p 0 &
 memtest -sa 800074000 -ea 840000000 -z -p 0 &
 memtest -sa C00074000 -ea C40000000 -z -p 0 &
 memtest -sa 2000074000 -ea 2040000000 -z -p 0 &
 memtest -sa 2400074000 -ea 2440000000 -z -p 0 &
 memtest -sa 2800074000 -ea 2840000000 -z -p 0 &
 memtest -sa 2C00074000 -ea 2C40000000 -z -p 0 &
 Exercising the DK* Disks (read-only)
 Testing the VGA (Alphanumeric Mode only)
 Exercising the EI* Network
Type "show_status" to display testing progress
 Type "cat el" to redisplay recent errors
 Type "init" in order to boot the operating system
P00>>>
```

test

Tests the entire system.

Syntax

test

Options

None

Arguments

None

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
 No DZ* Disks available for testing
 No DY* Disks available for testing
 Testing the DK* Disks (read only)
 No DU* Disks available for testing
 No DR* Disks available for testing
 No DQ* Disks available for testing
 No DF* Disks available for testing
 No MK* Tapes available for testing
 No MU* Tapes available for testing
 Testing the VGA (Alphanumeric Mode only)
 Testing the EI* Network
 P00>>>
```

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** I **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** I **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** I **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** and **wwidmgr.ps** at

ftp://ftp.digital.com/pub/Digital/Alpha/firmware/readmes/*vn.n*/doc/ (where *vn.n* is the latest firmware version).

For example, for version 6.3 it would be
ftp://ftp.digital.com/pub/Digital/Alpha/firmware/readmes/v6.3/doc/

Index

P

R

S

T