Alpha SRM Console for Alpha Microprocessor Evaluation Boards

User's Guide

Order Number: EC–QK8DC–TE

Revision/Update Information: This document supersedes the *Alpha SRM Console for Alpha Microprocessor Evaluation Boards User's Guide* (EC–QK8DB–TE).

Software Version: Alpha SRM Console Version V4.1–1 or higher

Operating System: Digital UNIX Version 3.2B and OpenVMS Version 6.2

Digital Equipment Corporation
Maynard, Massachusetts

September 1995

Contents

Preface

Overview

This document describes the Alpha SRM Console for the following Alpha microprocessor evaluation board systems running the Digital UNIX and OpenVMS operating systems:

- Alpha 21064 and Alpha 21064A PCI Evaluation Board (EB64+)
- AlphaPC64 Evaluation Board (AlphaPC64)
- Alpha 21066A Evaluation Board (EB66+)
- Alpha 21164 Evaluation Board (EB164)

This document also describes how to install the Alpha SRM Console and use the Alpha SRM Console commands.

Intended Audience and Prerequisites

This document provides Alpha SRM Console information for users to bootstrap the Digital UNIX and OpenVMS operating systems, and for system designers to diagnose hardware problems on their Alpha microprocessor based designs. For information specific to installing and booting the OpenVMS and Digital UNIX operating systems, see the OpenVMS and Digital UNIX installation guides.

Before you use this document, be familiar with your system's hardware configuration, and read your evaluation board user's guide.

Document Organization

This document is organized as follows:

- Chapter 1 describes the Alpha SRM Console.
- Chapter 2 describes how to install the Alpha SRM Console.
- Chapter 3 describes some basic Alpha SRM Console commands, associated environment variables, and the ISA configuration and firmware update utilities.
- Chapter 4 describes how to run the diagnostic firmware to test and debug various system components.
- Appendix A lists technical support and ordering information.

Document Conventions and Terms

In this document, the term "evaluation board" refers to an Alpha microprocessor evaluation board, and the term "evaluation board system" refers to all of the hardware and software components associated with the evaluation board.

The following conventions are used in this document:

Convention	Description
<angle_brackets>	Angle brackets indicate a placeholder in which the user must specify a value.
`boldface type`	Boldface type indicates user input or a menu selection.
`{braces, commas}`	Braces containing items separated by commas indicate mutually exclusive items.
`{braces | vertical | lines}`	Braces containing items separated by vertical lines indicate that more than one item may be selected.

(Continued on the next page)

(Continued)

Convention	Description
Ctrl/A	A slash between two key names indicates that the two keys must be pressed simultaneously.
Italic type	Italic type emphasizes important information and indicates complete titles of manuals.
Note:	Notes provide additional information about a topic.
`Monospaced type`	Text that the system displays is shown in monospaced type.
`[square_brackets]`	Square brackets enclose optional parameters, qualifiers, and values.

Related Documentation

See Appendix A for information about ordering related documentation.

Chapter 1
Introduction

Overview

This chapter describes the Alpha SRM Console and its features.

What Is the Alpha SRM Console?

Description

The Alpha SRM Console firmware provides the following service functions:

- Power-up diagnostics and initialization
- Operator interface
- Operating system bootstrap and restart

All evaluation board systems that run the Digital UNIX or OpenVMS operating system require the Alpha SRM Console firmware. This firmware resides in either a flash ROM or a UVPROM on the motherboard.

Features

The Alpha SRM Console provides the following features:

- Event-driven executive, providing:
  - Process management
  - Memory management
  - Symmetric multiprocessor support
  - Memory-resident file system
  - Interrupt and exception handling and error reporting
- UNIX style I/O and file system, providing:
  - Generic model for byte streamed devices or files
  - Local or remote console connections via serial ports or Ethernet

- SCSI, FAT, MOP, and BOOTP protocol (class) drivers
- Device (port) drivers:
  * Ethernet controllers: DECchip 21040 Ethernet LAN Controller for PCI, DE205 (ISA)
  * SCSI disk controllers: NCR810 (PCI), QLogic ISP1020 (PCI)
  * Storage: NVRAM, flash ROM, physical memory, virtual memory, GPRs, IPRs

• Console shell, providing:
  - Shell command parser with command line editing and recall
  - boot, set, show, examine, deposit, start, stop, and continue shell commands
  - Online help

• Diagnostics, exercisers, and test scripts, providing:
  - CPU, cache, and memory initialization and configuration
  - Device-specific self-test within the driver initialization
  - Exercisers for memory, buses, and devices
  - Test scripts for the evaluation board system, subsystems, and individual devices
  - Script creation and simple editing

Chapter 2

Installing the Alpha SRM Console

Overview

Depending on the type of evaluation board, the firmware is stored in either a flash ROM or a UVPROM. For evaluation boards that have a flash ROM, functions in the Windows NT ARC Firmware, the Debug Monitor firmware, and some versions of the Alpha SRM Console firmware can be used to update the flash ROM with the Alpha SRM Console firmware. For evaluation boards that have a UVPROM, a new UVPROM containing the Alpha SRM Console needs to be obtained and installed on the evaluation board. After the Alpha SRM Console firmware has been installed, the evaluation board system is restarted to activate the new firmware.

Use the following table to determine the update procedure for your evaluation board:

To update firmware on an...	See this section...
AlphaPC64	Updating Firmware in a Flash ROM
EB66+	Updating Firmware in a Flash ROM
EB164	Updating Firmware in a Flash ROM
EB64+	Updating Firmware in a UVPROM

Updating Firmware in a Flash ROM

On evaluation board systems that have a flash ROM, the Windows NT ARC Firmware and the Debug Monitor firmware are factory installed. You can use either to add the Alpha SRM Console firmware to the flash ROM. Systems running the Alpha SRM Console firmware Version V4.1 –1 or higher can also be used to update the flash ROM with new Alpha SRM Console firmware.

To update the flash ROM, the update enable/disable jumper must be in the enable position, which is the default. See the evaluation board user's guide for more information about jumper positions.

If your system is running...	**Then see this section...**
Windows NT	Updating the Flash ROM from Windows NT
The Debug Monitor	Updating the Flash ROM from the Debug Monitor
The Alpha SRM Console	Updating the Flash ROM from the Alpha SRM Console

Updating the Flash ROM from Windows NT

Windows NT ARC Firmware Conventions

The currently selected option in a Windows NT ARC Firmware menu is highlighted. To select and choose different options in the menus, use the following keys:

Key	Description
Arrow	The arrow keys are used to select different options.
Enter	The Enter key is used to choose the highlighted option.
Escape	The Escape key is used to close a menu or cancel an operation.

When you power up your evaluation board system, the firmware displays a blue screen on the monitor and initializes the firmware drivers. If autoboot is enabled, cancel it by pressing the Escape key before the timeout period expires. This allows you to interact with the firmware.

Using the Windows NT ARC Firmware to Install the Alpha SRM Console Firmware

To update the firmware on an evaluation board that has a flash ROM, use the following table to determine the installation diskette that corresponds with your evaluation board:

If you have an...	Use the installation diskette labeled...
AlphaPC64	AlphaPC64 Alpha SRM Console Firmware − Installation Floppy
EB66+	EB66+ Alpha SRM Console Firmware − Installation Floppy
EB164	EB164 Alpha SRM Console Firmware − Installation Floppy

After you have determined the installation diskette for your evaluation board, follow this procedure:

1. Insert the installation diskette into drive A.
2. Restart your evaluation board system.
3. From the Boot menu, choose **`Supplementary menu...`**.
4. From the Supplementary menu, choose **`Install new firmware`**.
5. From the Firmware update menu, choose **`Update SRM Console`**.

   Note: If the Update SRM Console option does not exist, see the Missing Windows NT Menu Selections section in this chapter and then return to step 1 of this procedure.

6. When the system prompts you to continue the update, choose **`Yes`**.
7. When the update has completed, restart your evaluation board system.

Switching to the Alpha SRM Console Firmware from Windows NT

To switch to the Alpha SRM Console firmware from the Windows NT ARC Firmware, follow this procedure:

1. From the Boot menu, choose **`Supplementary menu...`**.
2. From the Supplementary menu, choose **`Set up the system...`**.

3. From the Setup menu, choose **`Machine specific setup…`**.

4. From the Machine specific setup menu, choose one of the following operating systems:

   - **`Switch to OpenVMS`**
   - **`Switch to Digital UNIX`**

   Note: On some evaluation board systems, the **`Switch to Digital UNIX`** option appears as **`Switch to OSF`**.

   If these options do not exist, see the Missing Windows NT Menu Selections section in this chapter and then return to step 1 of this procedure.

5. Restart your evaluation board system.

6. Observe the Alpha SRM Console prompt (>>>) on the terminal attached to the COM1 serial port and on the graphics display unit.

   Note: If you do not see the Alpha SRM Console prompt (>>>), press the Enter key on the terminal attached to COM1 or on the console keyboard.

7. To specify the default console device, use the following Alpha SRM Console commands:

To use the...	Enter these commands...
Terminal attached to the COM1 serial port	>>>**`set console serial`** >>>**`init`**
Graphics display unit	>>>**`set console graphics`** >>>**`init`**

See Chapter 3 for more information about commands and environment variables.

Missing Windows NT Menu Selections

Certain Windows NT ARC Firmware menu selections required for installing or switching to the Alpha SRM Console firmware may be missing. These menu selections can be enabled by setting an environment variable.

Problem

The following menu selections do not appear:

Menu	Missing Selection
Firmware update	**`Update SRM Console`**
Machine specific setup	**`Switch to OpenVMS`**
Machine specific setup	**`Switch to Digital UNIX`**

Note: On some evaluation board systems, the **`Switch to Digital UNIX`** option appears as **`Switch to OSF`**.

Solution

Follow this procedure:

1. Restart your evaluation board system.
2. From the boot menu, choose **`Supplementary menu...`**.
3. From Supplementary menu, choose **`Set up the system...`**.
4. From the Setup menu, choose **`Edit environment variables`**.
5. Enter **`HAVE_SRM_CONSOLE`** when you are prompted for a name.
6. Type **`TRUE`** in uppercase and press the Enter key when you are prompted for a value.
7. Press the Escape key to return to the Setup menu.
8. From the Setup menu, choose **`Supplementary menu, and save changes...`**.
9. Restart the procedure that was missing the menu selections.

Updating the Flash ROM from the Debug Monitor

Using the Debug Monitor to Install the Alpha SRM Console Firmware

To update the firmware on an evaluation board that has a flash ROM, use the following table to determine the file name and installation diskette that corresponds with your evaluation board:

If you have an...	Use this file...	On this installation diskette...
AlphaPC64	apc64srm.rom	AlphaPC64 Alpha SRM Console Firmware − Installation Floppy
EB66+	eb66psrm.rom	EB66+ Alpha SRM Console Firmware − Installation Floppy
EB164	eb164srm.rom	EB164 Alpha SRM Console Firmware − Installation Floppy

After you have determined the file and installation diskette to install the Alpha SRM Console on your evaluation board, follow this procedure:

1. Insert the installation diskette into drive A.
2. At the Debug Monitor prompt, enter the following:

   flload <*filename.rom*> 300000

3. At the Debug Monitor prompt, enter the following:

   flash 300000 8

4. At the confirmation prompt, enter **Y** to proceed with firmware changes.

Switching to the Alpha SRM Console Firmware from the Debug Monitor

To switch to the Alpha SRM Console firmware from the Debug Monitor firmware, follow this procedure:

1. At the Debug Monitor prompt, enter **`bootopt OSF`** to switch to Digital UNIX or **`bootopt VMS`** to switch to OpenVMS.
2. Power down the evaluation board system.
3. Verify that the BOOT_OPTION jumper is inserted to enable your bootopt command. See the evaluation board user's guide for more information about jumper positions.
4. Restart the evaluation board system.
5. Observe the Alpha SRM Console prompt (>>>) on the terminal attached to the COM1 serial port and on the graphics display unit.

   Note: If you do not see the Alpha SRM Console prompt (>>>), press the Enter key on the terminal attached to COM1 or on the console keyboard.

6. To specify the default console device, use the following Alpha SRM Console commands:

To use the...	Enter these commands...
Terminal attached to the COM1 serial port	>>>**set console serial** >>>**init**
Graphics display unit	>>>**set console graphics** >>>**init**

See Chapter 3 for more information about commands and environment variables.

Updating the Flash ROM from the Alpha SRM Console

Minimum Alpha SRM Console Firmware Version

Only Alpha SRM Console firmware Version V4.1–1 or higher can be used to install the Alpha SRM Console firmware into the flash ROM.

Using the Update Utility to Install the Alpha SRM Console Firmware

To update the firmware on an evaluation board that has a flash ROM, use the following table to determine the file name and installation diskette that corresponds with your evaluation board:

If you have an...	Use this file...	On this installation diskette...
AlphaPC64	apc64srm.rom	AlphaPC64 Alpha SRM Console Firmware – Installation Floppy
EB66+	eb66psrm.rom	EB66+ Alpha SRM Console Firmware – Installation Floppy
EB164	eb164srm.rom	EB164 Alpha SRM Console Firmware – Installation Floppy

After you have determined the file and installation diskette to install the Alpha SRM Console on your evaluation board, follow this procedure:

1. Insert the installation diskette into drive A.
2. At the Alpha SRM Console prompt, enter the following:

   ```
   >>>update -path fat:<filename.rom>/dva0 -target \
   _>srmrom
   ```

3. At the confirmation prompt, enter **Y** to proceed with firmware changes.

4. Restart the evaluation board system.
5. Observe the Alpha SRM Console prompt (>>>) on the terminal attached to the COM1 serial port and on the graphics display unit.

   Note: If you do not receive the Alpha SRM Console prompt (>>>), press the Enter key on the terminal attached to COM1 or on the console keyboard.

6. To specify the default console device, use the following Alpha SRM Console commands:

To use the...	Enter these commands...
Terminal attached to the COM1 serial port	`>>>`**`set console serial`** `>>>`**`init`**
Graphics display unit	`>>>`**`set console graphics`** `>>>`**`init`**

See Chapter 3 for more information about commands, environment variables, and the update utility.

Updating Firmware in a UVPROM

Files for Programming the UVPROM

The Alpha SRM Console firmware is provided only on a diskette; a UVPROM is not provided. To program your own UVPROM, use the .ROM file or the .SR file on the installation diskette. The .ROM file is the actual image that resides in the UVPROM. The .SR file is the .ROM file translated into Motorola S–record format.

Use the following table to determine the file name and the installation diskette that corresponds with your evaluation board:

If you have an...	Use one of these files...	On the installation diskette labeled...
EB64+	eb64psrm.rom eb64psrm.sr	EB64+ Alpha SRM Console Firmware − Installation Floppy

Replacing the UVPROM

To update the firmware on an evaluation board after you have a programmed UVPROM, follow this procedure:

1. Turn off the power for the evaluation board system.
2. Locate and remove one of the UVPROMs from the evaluation board, noting the correct orientation of the UVPROM.
3. Insert the Alpha SRM Console firmware UVPROM, using the correct orientation.
4. Select the Alpha SRM Console firmware UVPROM device with the UVPROM select jumper as described in the evaluation board user's guide.
5. Turn on the power for the evaluation board system.
6. Observe the Alpha SRM Console prompt (>>>) on the terminal attached to the COM1 serial port and on the graphics display unit.

Note: If you do not receive the Alpha SRM Console prompt (>>>), press the Enter key on the terminal attached to COM1 or on the console keyboard.

7. To specify the default console device, use the following Alpha SRM Console commands:

To use the...	**Enter these commands...**
Terminal attached to the COM1 serial port	>>>**set console serial** >>>**init**
Graphics display unit	>>>**set console graphics** >>>**init**

See Chapter 3 for more information about commands and environment variables.

Chapter 3

Alpha SRM Console Commands

Overview

This chapter describes the Alpha SRM Console commands necessary to set environment variables, to configure the system, to bootstrap the operating system, and to update the firmware. This chapter is divided into the following sections:

- Alpha SRM Console Conventions and Special Keys
- Basic Alpha SRM Console Command Descriptions
- Environment Variables for Alpha SRM Console Commands
- ISA Configuration Utility
- Firmware Update Utility

Alpha SRM Console Conventions and Special Keys

Conventions

The following table lists the console conventions that apply to all Alpha SRM Console commands:

Convention	Description
Backslash (\) at the end of a line	Continuation symbol to continue long commands on the next line.
>>>	Console prompt
_>	Console prompt for line continuation
Maximum command length	255 characters
Multiple contiguous spaces or tabs	Treated as a single space
Command abbreviations	Allowed, if not ambiguous
Command qualifiers or options	Prefix with a space and a dash (–)
Numbers	Hexadecimal, unless otherwise specified. (Registers, such as R0 –R31, are shown in decimal notation.)

Special Keys

The following table lists Alpha SRM Console special keys and their functions. These special keys provide command recall, line editing, and basic input/output control flow.

To do this function...	Use one of these keys or key sequences...
Terminate the command line input.	Enter
Delete one character to the left of the cursor.	Delete
Toggle insert/overstrike mode. (Overstrike is the default.)	Ctrl/A
Recall previous commands. (The last 16 commands are stored.)	Ctrl/B Up arrow Down arrow
Terminate the foreground process.	Ctrl/C
Move the cursor one position to the left.	Ctrl/D Left arrow
Move the cursor to the end of the line.	Ctrl/E
Move the cursor one position to the right.	Ctrl/F Right arrow
Move the cursor to the beginning of the line.	Ctrl/H
Suppress/resume (toggle) console output.	Ctrl/O
Resume the flow (XON) of data to the console.	Ctrl/Q
Retype the current command line.	Ctrl/R
Stop the flow (XOFF) of data to the console.	Ctrl/S
Delete the entire line.	Ctrl/U

Basic Alpha SRM Console Command Descriptions

This section describes the basic Alpha SRM Console commands that are necessary to boot the Digital UNIX and OpenVMS operating systems. This section describes the following commands:

- boot
- deposit
- examine
- set
- show

The Alpha SRM Console offers many different commands. For a complete list of Alpha SRM Console commands, enter **`help`** at the Alpha SRM Console prompt (>>>).

boot

The boot command initializes the processor, loads a program image from the specified boot device, and transfers control to the loaded image.

Syntax

boot [-file <filename>] [-flags <longword>[,<longword>]] [-protocols <enet_protocol>] [-halt] [<boot_device>]

Arguments

<boot_device>
A device path or list of devices from which the firmware will attempt to boot. Use the **set bootdef_dev** command to set an environment variable that specifies a default boot device.

Options

-file <filename>
Specifies the name of a file to load into the system. Use the **set boot_file** command to set the environment variable that specifies a default boot file.

-flags <longword>[,<longword>]
Specifies additional information for the operating system. For systems with OpenVMS, root number and boot flags are specified here. For Digital UNIX systems, the following values may be used:

i = Interactive boot
s = Boot to single user
a = Autoboot to multiuser

Use the **set boot_osflags** command to set an environment variable that specifies a default boot flag value.

-protocols <enet_protocol>
Specifies the Ethernet protocols that will be used for a network boot. Values may be mop or bootp.

–halt
Forces the bootstrap operation to halt and invoke the console program after the image is loaded and the page tables and other data structures are set up.

Description

The boot command initializes the processor, loads a program image from the specified boot device, and transfers control to that image. If you do not specify a boot device in the command line, the default boot device is used. The default boot device is determined by the value of the bootdef_dev environment variable.

If you specify a list of devices, a bootstrap is attempted from each device in the order in which the device is listed. Then control passes to the first successfully booted image. In a list, always enter network devices last, because network bootstraps terminate only if a fatal error occurs or if an image is successfully loaded.

The –flags option can pass additional information to the operating system about the boot that you are requesting. On an OpenVMS system, the –flags option specifies the system root number and boot flags. If you do not specify a boot flag qualifier, the default boot flags value specified by the boot_osflags environment variable is used.

The –protocols option allows selection of either the DECnet MOP or the TCP/IP BOOTP network protocols. The keywords mop and bootp are valid arguments for this option. It is possible to set the default protocol for a port by setting the environment variable ewa0_protocols or era0_protocols to the appropriate protocol.

Explicitly stating the boot flags or the boot device overrides the current default value for the current boot request, but does not change the corresponding environment variable.

See the Environment Variables section in this chapter for more information about environment variables.

Examples

The following boot command boots the system from the default boot device. The console program returns an error message if a default boot device has not been set.

```
>>>boot
```

The following boot command boots the system from Ethernet port ewa0:

```
>>>boot ewa0
```

The following boot command boots the system, using the file named dec2.sys from Ethernet port ewa0: :

```
>>>boot –file dec2.sys ewa0
```

The following boot command boots the system, using the TCP/IP BOOTP protocol from Ethernet port ewa0:

```
>>>boot –protocol bootp ewa0
```

The following boot command boots the system from the default boot device, using flag settings 0,1:

```
>>>boot –flags 0,1
```

The following boot command loads the image from disk dka0, but remains in console mode. Subsequently, you can enter the continue command to transfer control to the operating system.

```
>>>boot –halt dka0
```

deposit

The deposit command writes data to an address that you specify. The address can be a memory location, register, device, or file.

Syntax

deposit `[-{b,w,l,q,o,h}] [{physical, virtual, gpr, fpr, ipr}] [-n <count>] [-s <step>] [<device>:]<address> <data>`

Arguments

`<device>:`

The optional device name or address space selects the device to access. The following platform-independent devices are supported:

- pmem: physical memory.
- vmem: virtual memory. All access and protection checking occur. If the access is not allowed to a program running with the current processor status (PS), the console issues an error message. If memory mapping is not enabled, virtual addresses are equal to physical addresses.

The following platform-dependent devices are supported:

`<address>`
An address that specifies the offset within a device into which data is deposited. The address may be any legal symbolic address.

Valid symbolic addresses are:

`gpr-name`	Represents general-purpose register.
`ipr-name`	Represents internal processor register.
`PC`	Program counter.
`+`	The location immediately following the last location referenced by an examine or deposit operation.
`-`	The location immediately preceding the last location referenced by an examine or deposit operation.
`*`	The location last referenced by an examine or deposit operation.
`@`	The location addressed by the last location referenced by an examine or deposit operation.

`<data>`
The data to be deposited.

Options

-b
Specifies the data type is byte.

-w
Specifies the data type is word.

-l
Specifies the data type is longword.

-q
Specifies the data type is quadword.

-o
Specifies the data type is octaword.

-h
Specifies the data type is hexword.

–physical
References physical address space.

–virtual
References virtual address space.

–gpr
References general-purpose register address space.

–fpr
References floating-point register address space.

–ipr
References internal processor register address space.

–n `<count>`
Specifies the number of consecutive locations to examine.

–s `<step>`
Specifies the address increment as a hexadecimal value. This option allows you to override the increment that is normally derived from the data size.

Description

The deposit command writes data to the address specified, such as a memory location, register, device, or file. The defaults for address space, data size, and address are the last specified values. After initialization, the default for address space is physical memory; for data size, the default is a quadword; and for address, the default is zero.

An address or device can be specified by concatenating the device name with the address. For example, use **pmem:0** and specify the size of the address space to be written. If a conflicting device, address, or data size is specified, the console ignores the command and issues an error response.

Examples

The following deposit command clears the first 512 bytes of physical memory:

```
>>>d -n 1ff pmem:0 0
```

The following deposit command writes the value 5 into four longwords, starting at physical memory address 1234:

```
>>>d -l -n 3 pmem:1234 5
```

The following deposit command loads GPRs R0 through R8 with −1:

```
>>>d -n 8 r0 ffffffff
```

The following deposit command writes the value 8 in the first longword of the first 17 pages in physical memory:

```
>>>d -l -n 10 -s 200 pmem:0 8
```

examine

The examine command displays the contents of an address you specify. The address can be a memory location, register, device, or file.

Syntax

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

Arguments

<device>:
The optional device name (or address space) selects the device to access.

<address>
The address specifies the first location to examine within the current device. The address can be any legal address specified.

Options

–b
Specifies the data type is byte.

–w
Specifies the data type is word.

–l
Specifies the data type is longword.

–q
Specifies the data type is quadword.

–o
Specifies the data type is octaword.

–h
Specifies the data type is hexword.

–d
Specifies the data displayed is the decoded macro instruction. The Alpha instruction decode (–d) does not recognize machine-specific PALcode instructions.

–physical
References physical address space.

–virtual
References virtual address space.

–gpr
References general-purpose register address space.

–fpr
References floating-point register address space.

–ipr
References internal processor register address space.

–n `<count>`
Specifies the number of consecutive locations to examine.

–s `<step>`
Specifies the address increment as a hexadecimal value. This option allows you to override the increment that is normally derived from the data size.

Description

The examine command displays the contents of the specified address, such as a memory location, register, device, or file. The defaults for address space, data size, and address are the last specified values. After initialization, the default for address space is physical memory; for data size, the default is a quadword; and for address, the default is zero.

An address or device can be specified by concatenating the device name with the address. For example, use **pmem:0** and specify the size of the address space to be displayed. If a conflicting device, address, or data size is specified, the console ignores the command and issues an error response.

The display line consists of the device name, the hexadecimal address (or offset within the device), and the examined data, also in hexadecimal.

The examine command supports the same options as the deposit command. Additionally, the examine command supports instruction decoding with the –d option, which disassembles instructions beginning at the current address.

Examples

The following examine command displays the contents of R0, using a symbolic address:

```
>>>e r0
gpr:  0 (R0)  0000000000000002
```

The following examine command displays the contents of R0, using address space:

```
>>>e -g 0
gpr:  0 (R0)  0000000000000002
```

The following examine command displays the contents of R0, using a device name:

```
>>>e gpr:0
gpr:  0 (R0)  0000000000000002
```

The following examine command displays the contents of R7 and the next five registers: `>>>examine -n 5 r7`

```
gpr:  38 (R7)  0000000000000000
gpr:  40 (R8)  0000000000000000
gpr:  48 (R9)  0000000000000000
gpr:  50 (R10) 000000007FFBF800
gpr:  58 (R11) 000000007FF781A2
gpr:  60 (R12) 0000000000000000
```

The following examine command displays the contents of internal processor register 11:

```
>>>examine ipr:11
ipr:  11 (KSP) FFFFFFFF8228DFD0
```

set

The set command sets or modifies the value of an environment variable.

Syntax

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

Arguments

<envar>
The environment variable to be assigned a new value.

<value>
The value that is assigned to the environment variable. It can be either a numeric value or an ASCII string.

Options

-default
Restores an environment variable to its default value.

-integer
Creates an environment variable as an integer.

-string
Creates an environment variable as a string.

Description

The set command is used to set or modify the value of an environment variable. Environment variables are used to pass configuration information between the console and the operating system. See the Environment Variables section in this chapter for more information about environment variables.

Examples

The following set command modifies the default boot device to ewa0:

```
>>>set bootdef_dev ewa0
```

The following set command attempts to boot the console following an error, halt, or power-up:

```
>>>set auto_action boot
```

The following set command modifies the default boot flags to 0,1:

```
>>>set boot_osflags 0,1
```

The following set command creates an environment variable called foobar and gives it a value of 5:

```
>>>set foobar 5
```

The following set command sets up the system to start the Windows NT ARC Firmware after the next power cycle:

```
>>>set os_type nt
```

show

The show command displays the current value of a specified environment variable or information about the system.

Syntax

show [{config, device [device_name], iobq, hwrpb, map, memory, pal, version, <envar>...}]

Arguments

config
Displays the current memory configuration, PCI logical slots, and ISA logical slots (based on the ISA configuration utility input to the configuration database).

device [device name]
Displays the devices and controllers in the system. Specifying a device name returns information on that device only.

iobq
Displays the input/output counter blocks.

hwrpb
Displays the hardware restart parameter block.

map
Displays the system virtual memory map.

memory
Displays the memory module configuration.

pal
Displays the version of Digital UNIX and OpenVMS PALcode.

version
Displays the version of the console firmware.

<envar>
Displays the current value of a specified environment variable.

Description

The show command displays information about the system and the current value of a specified environment variable.

For more information about environment variables, see the Environment Variables section in this chapter.

Examples

The following show command lists device information, such as system designation, drive model, or Ethernet address:

```
>>>show device
dka0.0.0.6.0   DKA0   RZ26L  441A
dka400.4.0.6.0 DKA400 RRD43  3213
dva0.0.0.0.1   DVA0
ewa0.0.0.12.0  EWA0 08-00-2B-E2-1C-25
pka0.7.0.6.0   PKA0   SCSI Bus ID 7
```

The following show command lists system random-access memory (RAM) size:

```
>>>show memory
48 Meg of System Memory
```

The following show command lists all environment variables and their settings:

```
>>>show *
```

The following show command lists all environment variables, beginning with boot:

```
>>>show boot*
```

Environment Variables for Alpha SRM Console Commands

This section describes environment variables that are used to define system operational state and to pass information between the firmware and the operating system. Environment variables are classified as either Alpha SRM Console architecture required or system defined.

Environment Variable Descriptions

SRM-Required Environment Variables

The following table shows common Alpha SRM Console architecture required environment variables and their descriptions. For a complete list, enter **`show *`** at the Alpha SRM Console prompt.

Variable	Description
`auto_action`	When used with the set or show command, this variable modifies or displays the console action that follows an error, halt, or power-up. The action can be halt, boot, or restart. The default is halt.
`boot_file`	When used with the set or show command, this variable modifies or displays the file name to be used when a bootstrap requires a file name. The default is null.
`boot_osflags`	When used with the set or show command, this variable modifies or displays the additional parameters to be passed to system software. For OpenVMS software, these parameters are the system root number and boot flags. The default is 0.
`bootdef_dev`	When used with the set or show command, this variable modifies or displays the default device or device list from which the system will attempt to boot. If the system software is preloaded, the variable is preset to point to the device containing the preloaded software. The default is null.

System-Defined Environment Variables

The following table shows common system-defined environment variables and their descriptions. For a complete list, enter **show *** at the Alpha SRM Console prompt.

Variable	Description
console	When used with the set command, this variable modifies the console output to either the serial port or the graphics controller.
control_scsi_term	This variable is unused in the evaluation board system.
ewa0_mode	This variable determines whether either the AUI (ThinWire) or the twisted-pair Ethernet ports will be enabled. AUI is the default. (Autosensing is not supported.)
os_type	When used with the set or show command, this variable modifies or displays the specified firmware that will be loaded on the next power cycle. The possible values are: **nt** = Select the Windows NT ARC Firmware. **osf** = Select the Alpha SRM Console firmware. **vms** = Select the Alpha SRM Console firmware.
pci_parity	This variable controls PCI parity checking. The possible values are: **on** = Parity checking is enabled. **off** = Parity checking is disabled; this is the default. **sniff** = Parity checking is enabled or disabled depending on the PCI device.
oem_string	When used with the set or show command, this variable modifies or displays a text string that identifies the product name in the Alpha SRM Console banner.

(Continued on the next page)

(Continued)

<table>
<tr><th>Variable</th><th>Description</th></tr>
<tr><td>language n</td><td>The language environment variable assigns language n to the system (where n is the option number of a language listed below). Use the following procedure to select the language:
1. At the Alpha SRM Console prompt, enter the following:

>>>set language 0
>>>init

The following menu and prompt are displayed:
<table>
<tr><th>n Language</th><th>n Language</th></tr>
<tr><td>0 none (display menu)</td><td>40 Francais (Suisse Romande)</td></tr>
<tr><td>30 Dansk</td><td>42 Italiano</td></tr>
<tr><td>32 Deutsch</td><td>44 Nederlands</td></tr>
<tr><td>34 Deutsch (Schweiz)</td><td>46 Norsk</td></tr>
<tr><td>36 English (American)</td><td>48 Portugues</td></tr>
<tr><td>38 English (British/Irish)</td><td>4A Suomi</td></tr>
<tr><td>3A Espanol</td><td>4C Svenska</td></tr>
<tr><td>3C Francais</td><td>4E Vlaams</td></tr>
<tr><td>3E Francais (Canadian)</td><td></td></tr>
</table>

(1..16):

2. Enter the number that corresponds to the language you want to use. The following example shows how to assign the American English language to the system:

(1..16):36

3. When you receive a message to reset the system, power cycle the system.</td></tr>
</table>

ISA Configuration Utility

This section describes how to configure Industry Standard Architecture (ISA) options before using a new option module with an evaluation board system.

isacfg

The ISA configuration (isacfg) utility allows you to enter installation information about ISA option modules.

Syntax

isacfg [-init] [-slot <slot#>] [-dev <device#>] [-all|-rm|-mk|-mod] [-<field> <value>]...

Arguments

None

Options

–init
Initializes the configuration table to the default settings.

–slot <slot#>
Allows you to enter a unique slot number for each ISA adapter. You may assign the numbers in any order. The slot number does not relate to a physical ISA adapter position. Slot 0 is reserved for the motherboard devices.

–dev <device#>
Optional; defaults to 0 if not entered. On a multifunction or multiport adapter, this specifies the device on the adapter.

–all
Shows the entire configuration table. Overrides all other commands.

–rm
Deletes an entry from the table.

–mk
Adds an entry to the table.

–mod
Modifies an entry in the table.

The following are the optional field names and associated values:

–<dmachan{0–3}> `<value>`
Allows you to specify up to four direct memory access (DMA) channels for the device. The valid values are 0,1,2,3.

–<dmamode{0–3}> `<value>`
Allows you to specify the DMA type for –dmachan{0-3}. The following values specify the DMA modes:

1 = Block
2 = Demand
4 = Single
8 = Cascade

–<enadev> `<value>`
Allows you to specify whether an entry is enabled or disabled. Disabled devices are not used in resource allocation calculations. The following values specify whether an entry is enabled or disabled:

0 = No (disabled)
1 = Yes (enabled)

–<etyp> `<value>`
Defines an entry type. The following values specify an entry type:

0 = Causes the entry to be deleted
1 = Single option
2 = Embedded multiport device
3 = Multiport option device

–<handle> `<string>`
Binds a name to the driver (up to 15 characters).

–<iobase>{0–5}> `<value>`
Specifies up to six I/O base registers (in hexadecimal) for a particular device entry.

–<irq{0–3}> `<value>`
Allows you to assign up to four interrupt request (IRQ) channels to the device (use decimal IRQ levels).

–<membase{0–2}> `<value>`
Specifies up to three read/write ISA memory regions.

–<memlen{0–2}> <value>
Specifies the length of a memory region for one of three read/write memory regions.

–<rombase> <value>
Specifies an address for the ISA BIOS on an external ROM.

–<romlen> <value>
Specifies the length of the ROM.

–<totdev> <value>
Allows you to record the total number of devices for a particular slot.

Description

ISA devices are not capable of being probed for configuration information by the Digital UNIX or OpenVMS operating systems. Therefore, you must enter ISA option information manually by using the ISA configuration utility (isacfg). You must use this utility before installing a new ISA option module on an evaluation board system that is running the Digital UNIX or OpenVMS operating systems.

Adding ISA Options

To add a supported ISA option to an evaluation board system running the Digital UNIX or OpenVMS operating system, use the following procedure:

Step	Action	Result
1	Perform operating system configuration tasks, if any. Refer to your operating system installation guide and release notes.	The operating system is prepared for the ISA option.
2	Shut down the operating system.	The system displays the Alpha SRM Console prompt (>>>).

(Continued on the next page)

(Continued)

Step	Action	Result
3	Enter **`isacfg <options>`** at the Alpha SRM Console prompt (>>>).	The new ISA option is added to the Alpha SRM Console configuration table.
4	Enter **`init`** at the Alpha SRM Console prompt (>>>).	The Alpha SRM Console is reinitialized.
5	Refer to the ISA option documentation to configure the ISA option.	The ISA option is properly configured.
6	Refer to the evaluation board documentation to turn off the system and install the ISA option.	The ISA option is properly installed.
7	Turn on and boot the system.	The operating system boots and recognizes the new ISA option.

Examples

The following example shows how to use the isacfg utility to enter configuration information for the DE205 Ethernet controller option into the configuration database. Examples to display, modify, and remove table entries are included as well. In some cases, there are scripts available to issue the proper isacfg command. Script commands are preceded by an add_ prefix.

The following isacfg command uses the built-in script to add the DE205 option:

```
>>>add_de205
```

The following isacfg command, which performs the same function as the add_de205 command in the previous example, adds the DE205 option:

```
>>>isacfg -slot 1 -dev 0 -mk -handle DE200-LE \
_>-irq0 5 -iobase0 300 -etyp 1 -enadev 1
```

The following isacfg command displays the configuration database:

```
>>>isacfg -all
```

The following isacfg command modifies the irq0 entry of an option:

```
>>>isacfg -mod -slot 1 -irq0 14
```

The following isacfg command removes an entry:

```
>>>isacfg -rm -slot 1 -dev 0
```

Firmware Update Utility

This section describes the update utility for writing new firmware into the flash ROM. See Chapter 2 for more information about using the update utility to install the Alpha SRM Console.

update

The update utility writes new firmware into the flash ROM.

Syntax

```
update [-file <filename>] [-protocol <transport>]
[-device <source_device>] [-target <target_name>]
[-path <full_pathname>] [-ignore]
```

Arguments

None

Options

-file `<filename>`
Specifies the name of the new flash ROM image.

-protocol `<transport>`
Specifies the source transport protocol.

-device `<source_device>`
Specifies the device from which to load the new flash ROM image.

-target `<target_name>`
Specifies the flash ROM device to be updated. The following target names may be used:

arcrom = Windows NT ARC Firmware ROM

srmrom = Alpha SRM Console firmware ROM

-path `<full_pathname>`
Used to fully specify the file, protocol, and device names; this option overrides the file, protocol, and device options.

-ignore
Ignores the image signature and consistency checks. If specified, the flash ROM update occurs regardless of the built-in consistency checks.

Description

The update utility loads a flash ROM image from the specified source device, using the specified transport protocol, and writes this image to the flash ROM device.

The following table describes the flash ROM update process:

Phase	Description
1	The image is loaded from the source device into memory.
2	Consistency checks are applied to the image. Specifically, checksums, signatures, and sizes must match. The update utility will display error messages if any of the consistency checks fail. The update utility will then exit unless the –ignore option is specified.
3	The update utility prompts the user to confirm the flash ROM update with the message "`Do you really want to continue [Y/N]?`" If **`n`** is entered in response to this prompt, the update will be aborted with a return to the console prompt. If **`y`** is entered, the flash ROM update will commence.
4	The update utility writes the flash ROM with the loaded image. At this point, it is imperative that the update utility *not* be halted or interrupted for any reason. If interrupted, the flash ROM will most likely be left in a corrupt or inoperable state. If the update was successful, a success message is printed to the console. Note that it is necessary to reset or power cycle the system to unload the new image from the flash ROM into memory. Until a power down or reset occurs, the old image is still being executed.

Examples

The following update command writes the flash ROM with the image cfw_e42.sys, which is loaded from the dva0 diskette device, using the FAT file system protocol:

```
>>>update -file cfw_e42.sys -protocol fat -device dva0
```

The following update command writes the flash ROM with the image, cfw_e42.sys, which is loaded from the dva0 diskette device, specifying a fully qualified path name:

```
>>>update -path fat:cfw_e42.sys/dva0
```

DEC EtherWORKS 3 Configuration Utility

This section describes how to configure the DE205 ISA network interface module.

ewrk3_config

The DEC EtherWORKS 3 configuration utility programs the DE205 onboard EEROM.

Syntax

```
ewrk3_config [-curaddr <io_base>] [-ioaddr <io_base>]
[-bufsize <mode>] [-memaddr <memory_base>][-irq
<irq_line>] [-fbus] [-ena16] [-default] [-show]
```

Arguments

None

Options

–curaddr <io_base>
Specifies the current hex I/O base address (physical bus address) where the network interface module is located.

–ioaddr <io_base>
Specifies the hex I/O base address (physical bus address) for the current network interface module. The default address range is 300 (hex) through 31F (hex).

–bufsize <mode>
Specifies the memory mode and the amount of actual memory utilized and owned by the network interface module. The possible settings are 2K, 32K, or 64K. In most cases, Digital recommends that the 2K setting be used, leaving a maximum amount of high memory for other application programs.

–memaddr <memory_base>
Specifies the base memory address for the network interface module. Depending upon the memory buffer size and mode selected, the network interface module can be in any unused high memory area. In 2KB mode, this includes segments A through F on any 2KB boundary; 32KB mode allows segments A through F on any 32KB boundary; 64KB mode allows segments A through F on a 64KB boundary.

–irq <irq_line>
Specifies the IRQ line for the network interface module to interrupt the CPU. The default for the interrupt line is IRQ5. The possible values for the IRQ lines are:

5 = IRQ5

10 = IRQ10

11 = IRQ11

15 = IRQ15

–fbus
Enables the fast bus option for bus clock speeds greater than the standard ISA 8.33 MHz.

–ena16
Enables 16-bit transfer mode, and indicates that 16-bit memory transfers should be used. The default mode is 16-bit transfer.

–default
Sets the network interface module to the default settings.

–show
Displays the entire contents of the EEROM of the network interface module.

Description

The DEC EtherWORKS 3 configuration utility programs the DE205 onboard EEROM, and is similar to the isacfg utility. In order to use multiple network interface modules in a system, the modules must be inserted one at a time to change, at a minimum, their I/O base address to avoid address conflicts. The system must then be power cycled for the new settings to take effect.

Entering the **`ewrk3_config`** command without options causes the entire address space to be searched for DE205 modules and displays the I/O base addresses of all modules found.

Examples

The following ewrk3_config command displays the I/O base addresses of all DE205 modules in the system:

```
>>>ewrk3_config
```

The following ewrk3_config command sets the DE205 module back to its default settings. The system must be power cycled for the new settings to take effect.

```
>>>ewrk3_config -default
```

The following ewrk3_config command changes a DE205 module's I/O base address from its default setting to address 340h. The system must be power cycled for the new settings to take effect.

```
>>>ewrk3_config -curaddr 300 -ioaddr 340
```

Chapter 4

Alpha SRM Console Diagnostic Commands

Overview

This chapter describes how to run the diagnostic firmware to test and debug various system components. This chapter is divided into the following sections:

- Running the Alpha SRM Console Diagnostic Firmware
- Alpha SRM Console Diagnostic Command Descriptions
- Environment Variables for Diagnostic Commands

Running the Alpha SRM Console Diagnostic Firmware

Requirements

Some diagnostic commands exercise graphic devices that require a terminal attached to the COM1 serial port to be the default terminal device.

Bootstrapping the Diagnostic Firmware

To run diagnostic commands on your system, use the following table to determine the diagnostic diskette that corresponds with your evaluation board:

If you have an...	Use the diagnostic diskette labeled...
AlphaPC64	AlphaPC64 Alpha SRM Console Firmware Diagnostics
EB66+	EB66+ Alpha SRM Console Firmware Diagnostics
EB164	EB164 Alpha SRM Console Firmware Diagnostics
EB64+	EB64+ Alpha SRM Console Firmware Diagnostics

After you have determined the diagnostic diskette for your evaluation board, follow this procedure to bootstrap the Alpha SRM console diagnostic firmware:

Step	Action
1	Insert the diagnostic diskette into drive A.
2	At the Alpha SRM console prompt, enter the following command: `>>>`**`boot -flag 0,a0 dva0`**
3	The Alpha SRM console will restart. Observe the Alpha SRM console prompt (`>>>`) on the default console terminal device.

Alpha SRM Console Diagnostic Command Descriptions

This section describes some of the diagnostic commands that test and debug various system components. Diagnostic commands that are described in the description section as *scripts* can be customized for a particular purpose.

This section describes the following commands:

- exer
- exer_read
- exer_write
- kill_diags
- memexer
- memtest
- nettest
- show_status
- sys_exer
- test

The Alpha SRM Console offers many different commands. For a complete list of Alpha SRM Console commands, enter **help** at the Alpha SRM Console prompt (>>>).

exer

The exer command exercises one or more devices by performing read, write, and compare operations, and reports the performance statistics.

Syntax

```
exer [-sb  <start_block> ] [-eb  <end_block> ]
[-p  <pass_count> ] [-l  <blocks> ] [-bs  <block_size> ]
[-bc <block_per_io> ] [-d1  <buf1_string> ]
[-d2 <buf2_string> ] [-a  <action_string> ]
[-sec  <seconds> ] [-m] [-v] [-delay <milliseconds> ]
<device_name>...
```

Arguments

`<device_name>`
Specifies the names of the devices or file streams to be exercised.

Options

–sb `<start_block>`
Specifies the starting block number (hex) within file stream. The default is 0.

–eb `<end_block>`
Specifies the ending block number (hex) within file stream. The default is 0.

–p `<pass_count>`
Specifies the number of passes to run the exerciser. If the number of passes is set to 0, the devices will be activated continuously until halted with a Ctrl/C. The default is 1.

–l `<blocks>`
Specifies the number of blocks (hex) to exercise. The –l option has precedence over the –eb option. When reading from a device without the –l and –eb options specified, the read operation continues until an end of file is detected. When writing to a device without the –l and –eb options

specified, the write operation continues for the size of the device. The default is 1.

–bs <block_size>
Specifies the block size (hex) in bytes. The default is 200 (hex).

–bc <block_per_io>
Specifies the number of blocks (hex) for each read and write operation. On devices without a predetermined length, such as tape devices, the packet size specified for the device should be used. The default packet size is 2048. The maximum block size allowed with variable length block read operations is 2048 bytes. The default is 1.

–d1 <buf1_string>
Specifies the value of the string argument variable for the eval command to generate the buffer1 data pattern. The value of buffer1 is initialized once prior to all read and write operations. The default value of the string argument variable is all bytes set to 5A's (hex).

–d2 <buf2_string>
Specifies the value of the string argument variable for the eval command to generate the buffer2 data pattern. The value of buffer2 is initialized once prior to all read and write operations. The default value of the string argument variable is all bytes set to 5A's (hex).

–a <action_string>
Specifies an 'action string' that determines the sequence of read, write, and compare operations to various buffers. The default action string is '?r'. The possible values for the action string are:

b = Add a constant to buffer1.

c = Compare buffer1 with buffer2.

n = Write without lock from buffer1.

r = Read into buffer1.

s = Sleep for a number of milliseconds specified by the delay qualifier. If no delay qualifier is present, sleep for 1 millisecond.

> **Note:** Times reported in verbose mode will not be accurate when this action string is used.

w = Write from buffer1.

z = Zero the contents of buffer1.

B = Add a constant to buffer2.

N = Write without lock from buffer2.

R = Read into buffer2.

W = Write from buffer2.

– = Seek to file offset prior to last read or write operation.

? = Seek to a random block offset within the specified range of blocks.

–sec <seconds>
Specifies to terminate the exercise after the number of seconds have elapsed. By default, the exerciser continues until the specified number of blocks or passcount are processed.

–m
Specifies metrics mode. When the exer command has completed, a total throughput line is displayed.

–v
Specifies verbose mode. All data read is displayed on the default terminal device. This option is not applicable on write or compare operations. The default is verbose mode off.

–delay <milliseconds>
Specifies the number of milliseconds to delay when "s" appears as a character in the action string.

Description

The exer command exercises one or more devices by performing read, write, and compare operations. Two buffers, buffer 1 and buffer 2, are used to carry out these operations. 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 writes from two specified devices into two buffers and compares the contents of the buffers. A read or write operation can use either buffer, while a compare operation uses both buffers.

Options, postfix string arguments, and qualifiers are used with the exer command to specify the following:

- The address range to test within the test devices.
- The packet size, also known as the I/O size, which is the number of bytes read or written in one I/O operation.
- The number of passes to run the exer command.
- The number of seconds to run the exer command.
- The sequence of read, write, and compare operations on the test devices.

Both buffer1 and buffer2 are initialized to a data pattern before any read or write operations occur. These buffers are never reinitialized, even after completing one or more passes. The data patterns that the buffers are initialized with are either a hex 5A in every byte of each buffer or are specified via the string arguments to the optional data pattern qualifiers, –d1, –d2.

The –d1, –d2 qualifiers use a postfix string argument to initialize a buffer's contents. For each byte in the specified buffer, starting with the first byte, this postfix string is passed to the eval command, which returns a byte value that is then written to the specified buffer.

Several exer command qualifiers are used to specify the amount of device data to be processed. The qualifiers –sb, –eb, –l, –bs, and –bc specify, respectively: starting block, ending block, number of blocks, block size in bytes, and the number of blocks in a packet, where a packet is the amount of data transferred in one read or write operation.

Reading, writing, comparing buffers, and other operations can be specified to occur in various combinations and sequences. These operations are specified by a string of one-character command codes known as the 'action string'. The action string is specified as an argument to the action string qualifier, –a.

Each command code character in the action string is processed in a sequence from left to right. Each time that the exer command completes all of the operations specified by the action string, the exer command will reduce the remaining amount of device data to be processed by the size of the last packet processed by the action string. The action string is repeatedly processed until the specified amount of device data has been processed.

The lowercase action string characters, `bnrwz`, are used to specify operations that involve buffer1. The uppercase action string characters, `BNRWZ`, specify operations that involve buffer2. The action string character, 'c', involves both buffer1 and buffers. The action string characters, ' –?,' do not involve either buffer1 or buffer2.

The total number of bytes read or written on each pass of the exerciser is specified by the length in blocks or the length indicated by the starting and ending block address option arguments. If neither the ending address nor the length options are specified, then on each pass, the number of bytes processed could vary depending on whether or not the file stream is being written to or just being read. If the file stream is not being written to by the exer command, then it will read until an end-of-file is detected. If the exer command will be writing to the file (as specified in the action string), then the number of bytes processed per pass is equal to the allocation size of the file. The allocation size of the file is usually larger than the length of the file for RAM disk files, but equal to the length for disk devices.

All disk device read and write operations will fail if the block size is not equal to 1 or a multiple of 512. Partial block read and write operations are not supported so a length that is not a multiple of the block size will result in no errors, but the last partial block read and write operations of data will not occur.

Any combination of writing, reading, or comparing the buffer1 and buffer2 can be executed in the sequence as specified in the action string. Depending on the option arguments, some of the read, write, or compare operations may be omitted without affecting the execution of the other operations.

The exer command will return an error code immediately after a read, write, or compare error, if the diagnostic environment variable d_harderr is set to halt. When the diagnostic environment variable is set to continue or loop when an error is detected, then subsequent operations specified by the action string qualifier will occur except for compare operations. For example, if a read error occurs, a subsequent compare operation will be omitted because a read failure preceding a compare operation guarantees that the compare operation will fail. If subsequent block read and write operations succeed, then compare operations of those blocks will occur. When the exer command terminates completing all passes or by operator termination, then the status returned will be that of the last failed write, read, or compare operation, regardless of subsequent successful read or write operations.

Examples

The following command reads all SCSI-type disks for the entire length of each disk. It continues to read all disks concurrently until 36000 seconds (10 hours) have elapsed. Each read operation will occur at a random block number on each disk.

```
>>>exer dk*.* -p 0 -sec 36000
```

The following command writes the hex pattern 5A to every byte of blocks 1, 2, and 3 —using a packet size of 2048. The value of 2048 for the packet size is derived from multiplying the –bc value (block per I/O) of 4 by the default –bs value (block size) of 512.

```
>>>exer -sb 1 -eb 3 -bc 4 -a 'w' -d1 '0x5a' dka0
```

exer_read

The exer_read command script continuously reads random blocks of data from all online disks and displays errors.

Syntax

`exer_read` `[-sec <seconds>] [<device>...]`

Arguments

`<device>`
Specifies the device name to be exercised. The default is all online disks.

Options

`-sec` `<seconds>`
Specifies the number of seconds after which to terminate the exer_read command script. By default, this command script continues to execute until all blocks are processed or until the passcount value has been reached. The passcount value is specified by the d_passes environment variable. See the Environment Variables for Diagnostic Commands section in this chapter for more information.

Description

The exer_read command script randomly determines a block number on any online disk and reads a packet of 2048 bytes. Upon detecting an error, the exer_read command script displays an error report to the default console terminal device.

This command script continues to repeat random read operations until one of the following conditions occurs:

- All blocks on the devices have been read for the number of passes specified by the d_passes environment variable. See the Environment Variables for Diagnostic Commands section in this chapter for more information.
- The process for this command has been stopped by pressing Ctrl/C or by entering the kill_diags command.
- The amount of time specified by the –sec option has elapsed.

> **Note:** This command script will not display messages unless an error occurs.

Examples

The following exer_read command randomly reads blocks of data from all online disks and displays detected errors:

```
>>>exer_read
```

exer_write

The exer_write command script continuously reads and non-destructively writes random blocks of data on all online disks and displays errors.

Syntax

exer_write `[-sec <seconds>] [<device>...]`

Arguments

`<device>`
Specifies the device name to be exercised. The default is all online disks.

Options

-sec `<seconds>`
Specifies the number of seconds after which to terminate the exer_write command. By default, this command continues to execute until all blocks are processed or until the passcount value has been attained. The passcount value is specified by the d_passes environment variable. See the Environment Variables for Diagnostic Commands section in this chapter for more information.

Description

The exer_write command script randomly determines a block number on any online disk and reads a packet of 2048 bytes and then writes the same data back to the same location. The data from the read operation is then compared with the data on the disk for discrepancies. Upon detecting a discrepancy, the exer_write command displays an error report to the default console terminal device.

This command continues to repeat random read, write, and compare operations until one of the following conditions occurs:

- All blocks on the devices have been read for the number of passes specified by the d_passes environment variable. See the Environment Variables for Diagnostic Commands section in this chapter for more information.
- The process for this command has been stopped by pressing Ctrl/C or by entering the kill_diags command.
- The specified time has elapsed.

> **Note:** This command script will not display messages unless an error occurs.

Examples

The following exer_write command randomly reads, writes, and compares blocks of data from all disks that are online:

```
>>>exer_write
```

kill_diags

The kill_diags command script stops the execution of all diagnostic processes.

Syntax

`kill_diags`

Arguments

None

Options

None

Description

The kill_diags command script stops all executing diagnostic processes.

Examples

The following diagnostic commands start two memory exerciser processes and display the status of all executing diagnostics. A kill_diags command is then issued , which stops all memory exerciser processes. The show_status command is then issued a second time to confirm that the system is idle.

```
>>>memexer 2 &

>>>show_status

      ID   Program    Device    Pass   Hard/Soft  Bytes Written    Bytes Read
  ------   -------   -------   -----     -------  -------------    ----------
 00000001     idle    system       0      0    0              0             0
 00000351  memtest    memory       0      0    0       37748736      37748736
 00000352  memtest    memory       0      0    0       37748736      37748736

>>>kill_diags

>>>show_status

     ID    Program    Device   Pass    Hard/Soft  Bytes Written    Bytes Read
 ------    -------   -------  -----   ---------   -------------    ----------
00000001      idle    system      0       0    0              0             0

                                                              0             0

>>>
```

memexer

The memexer command script tests all available memory.

Syntax

memexer [<number>]

Arguments

<number>
Specifies the number of memory test processes to invoke. The default is 1.

Options

None

Description

The memexer command script invokes the requested number of memory tests running continuously in the background. Memory tests randomly allocate and test blocks of memory two times the size of the Bcache using all available memory.

> **Note:** This command script will not display messages unless an error occurs.

Examples

The following memexer command starts two memory tests running in the background:

```
>>>memexer 2

>>>show_status

      ID   Program     Device    Pass     Hard/Soft   Bytes Written       Bytes Read
  ------   -------    -------   -----     ---------    ------------       ----------
00000001      idle     system       0         0    0               0                0
00000107   memtest     memory      10         0    0       541065216        541065216
00000108   memtest     memory       0         0    0       541065216        541065216

>>>
```

memtest

The memtest command tests a section of memory with multiple write, read, and verify operations.

Syntax

```
memtest [-sa  <start_address> ] [-ea  <end_address> ]
[-l  <length> ] [-bs  <block_size> ] [-i  <address_inc> ]
[-d  <data_pattern> ] [-p  <pass_count> ]
[-rs <random_seed> ] [-rb ] [-f ] [-m ] [-z ]
[-h ] [-mb ] [-t <test_number>] [-ba <block_address>]
```

Arguments

None

Options

–sa <start_address>
Specifies the starting address for the test. The default is the first free space in memzone.

–ea <end_address>
Specifies the ending address for the test. The default is the start address plus the length.

–l <length>
Specifies the length of section to test in bytes. The default is the block_size, except with the –rb option, which uses the zone size. The –l option has precedence over the –ea option.

–bs <block_size>
Specifies the block (or packet) size (hex) in bytes. The default is 8192 bytes. This is only used for the random block test. For all other tests the block size equals the length.

–i <address_inc>
Specifies the address increment value in longwords. This value will be used to increment the address through the memory to be tested.

The default is 1 (longword). This is only implemented for the gray code test. An address increment of 2 tests every other longword. This option is useful for multiple CPUs testing the same physical memory.

–d <data_pattern>
Specifies to use this data pattern for testing memory. This is only used for the march test. The default pattern is all 5's.

–p <pass_count>
Specifies the number of times to execute the test. If the number of passes is set to 0, then the section of memory will be tested continuously until halted with a Ctrl/C. The default is 1.

–rs <random_seed>
Specifies the value of the random seed in order to vary the random data patterns generated. This option is only used for the random test. The default is 0.

–rb
Specifies to randomly allocate and test all of the specified memory address range. Allocations are done of block_size.

–f
Specifies fast mode. If –f is specified, the data compare is omitted. Only Error Correction Circuitry (ECC) and Error Detection Codes (EDC) errors are detected.

–m
Specifies to time the memory test. The elapsed time is displayed at the end of the test. By default the timer is off.

–z
Specifies the test will use the specified memory address without an allocation. This bypasses all checking but allows testing in addresses outside of the main memory heap. It also allows unaligned testing.

> **Warning:** This option permits testing and corrupting of any memory location.

–h
Specifies to allocate test memory from the firmware heap.

–mb
Specifies to use memory barriers after each memory access. When set, an mb (memory barrier) will be done after every memory access. This guarantees serial access to memory.

–t <test_number>
Specifies the test mask. The default is all tests.

–ba <block_address>
Specifies that data stored at this address will be used to write each block. This option is used only for the victim block test.

Description

The memtest command contains a gray code memory test, a marching 1's and 0's memory test, a random test, and a victim block test. These four tests, which are referred to as Memtest t1 through t4, can be run sequentially or individually to detect different types of memory problems. The default is for all four tests to run sequentially.

Multiple memtest commands can be issued concurrently, and can be used in conjunction with other commands to test the complete system.

> **Warning:** The memtest command will destroy the data in the section of memory under test.

Memtest t1 uses a gray code algorithm to test a specified section of memory. A gray code pattern and inverse gray code pattern are written, read, and verified for the specified address range.

Memtest t2 uses a marching 1's and 0's algorithm to test a specified section of memory. A marching pattern of 1's and 0's and inverse pattern are written, read, and verified for the specified address range.

Memtest t3 uses a random algorithm to test a specified section of memory. Random addresses, within the specified address range, can be tested with random data of random length.

Memtest t4 is a victim block test. This test will write blocks of data, victimize the data, then read and verify blocks of data for the specified address range.

The –f "fast" option can be specified to reduce the depth of testing for the memtest command. Using this option will stress the section of memory under test with a higher throughput, but will not detect address shorts. Failures can still be detected with the ECC and EDC logic.

Large sections of memory require more time for testing than small sections of memory. To make the memtest command run faster, a Ctrl/C or a kill command can only be detected while it is outside of all test loops. For this reason, a Ctrl/C or a kill command may not abort immediately.

Examples

The following memtest command example tests a section of memory starting at 0x200000 (–sa) for 0x1000 bytes (–l):

```
>>>memtest –sa 200000 –l 1000
```

The following memtest command example tests a section of memory from 0x200000 for 0x1000 bytes, but data is not verified (–f):

```
>>>memtest –sa 200000 –l 1000 –f
```

The following memtest command example has a default block size of 8192 bytes and is written from 0x300000 for 10 passes (–p):

```
>>>memtest –sa 300000 –p 10
```

The following memtest command example tests a section of memory from 0x200000 to 0x3fffff. This example also uses the –rb option, which allows every block within the range randomly allocated and prevents the reporting of errors if a block within the range cannot be allocated.

```
>>>memtest –sa 200000 –ea 400000 –rb
```

The following memtest command example has the console heap (–h) tested by randomly dynamically allocating 0x100 byte blocks (–bs):

```
>>>memtest –h –rb –bs 100
```

The following memtest command example performs tests across all of memzone (all memory excluding the HWRPB, the PAL area, the console, and the console heap). It is run in the foreground until Ctrl/C.

```
>>>memtest –rb –p 0
```

nettest

The nettest command tests the network ports by using MOP loopback.

Syntax

```
nettest   [-f <file>] [-mode <port_mode>]
[-p <pass_count>] [-sv <mop_version>] [-to <loop_time>]
[-w <wait_time>] [<port>]
```

Arguments

<port>
Specifies the Ethernet port on which to run the test.

Options

-f <file>
Specifies the file containing the list of network station addresses to which to loop messages. The default file name is lp_nodes_ewa0 for port ewa0. The default file name is lp_nodes_ena0 for port ena0. The files by default have their own station address.

-mode <port_mode >
Specifies the mode to set the port adapter (TGEC). The default is 'ex' (external loopback). Allowed values are:

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 times to run the test. If the number of passes is set to 0, the network ports will be tested continuously until halted with a Ctrl/C. The default is 1.

> **Note:** This is the number of passes for the diagnostic. Each pass will send the number of loop messages as set by the environment variables ena*_loop_count and ewa*_loop_count.

-sv <mop_version>
Specifies which MOP version protocol 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.

Description

The nettest command tests the network. This command contains options for MOP loopback tests that can test specific ports in internal loopback, external loopback or live network loopback mode.

This command can run separately or be included in a script to test the entire system.

Examples

This command performs an internal loopback test on port ewa0:

```
>>>nettest ewa0
```

This command performs a normal mode loopback test on port ewa0, using a list of nodes contained in the file ewa0_lp_nodes:

```
>>>nettest -f ewa0_lp_nodes -mode nm ewa0
```

show_status

The show_status command script shows the status of all executing diagnostic processes.

Syntax

`show_status`

Arguments

None

Options

None

Description

The show_status command script reports one line of information for each executing diagnostic process. The information includes error count, passes completed , and bytes read or written to the device being tested. The source of each line of information is an I/O block.

Examples

The following diagnostic commands start up two memory test processes and display the status of all executing diagnostic processes:

```
>>>memexer 2 &

>>>show_status

      ID   Program    Device    Pass   Hard/Soft   Bytes Written    Bytes Read
  ------   -------   -------   -----     -------   -------------    ----------
 00000001     idle    system       0       0    0              0             0
 00000351  memtest    memory       0       0    0       37748736      37748736
 00000351  memtest    memory       0       0    0       37748736      37748736

>>>
```

sys_exer

The sys_exer command script tests all subsystems and devices in the system concurrently.

Syntax

sys_exer `[-lb]`

Arguments

None

Options

`-lb`
Specifies external loopback tests to be performed on the parallel port and COM1 serial port.

Description

The sys_exer command script simultaneously tests all subsystems and system devices. All subsystems and devices are those that can be listed with the show config and show device commands, including: memory, disks, tapes, diskette, serial ports, parallel port, network, and graphic devices.

All tests will continue to execute concurrently until a kill_diags or an init command is issued. The set command, used with environment variables, can establish parameters such as whether to halt, loop, or continue on error. See the Environment Variables for Diagnostic Commands section in this chapter for more information.

The –lb option allows an external loopback test to be performed on the parallel port and serial port COM1. The sys_exer command ignores the passcount environment variable, d_passes.

Note: The sys_exer command attempts to test the entire system, including graphic devices. Commands that exercise graphic devices require that the terminal connected to the COM1 serial port be the default console terminal device.

To determine your default terminal device and to specify that the terminal attached to the COM1 serial port be your default device, use the following Alpha SRM console commands:

To do this function ...	Enter these commands...
Determine the default terminal device.	`>>>`**`show console`**
Set the terminal attached to the COM1 serial port to be the default console terminal device.	`>>>`**`set console serial`** `>>>`**`init`**

Examples

These commands run the sys_exer script to test all system components concurrently and display the status of all executing diagnostic processes.

```
>>>sys_exer &

>>>show_status

      ID    Program        Device  Pass   Hard/Soft   Bytes Written   Bytes Read
  ------     ------         -----   ---   ---------          ------        -----
00000001       idle        system     0       0    0              0            0
0000009f    memtest        memory     0       0    0      100663296    100663296
000000b4   exer_kid   dub0.0.0.1.0     0       0    0              0       444416
000000b5   exer_kid   duc0.6.0.2.0   122       0    0              0      1249280
000000b6   exer_kid   dud0.7.0.3.0   122       0    0              0      1249280
000000b7   exer_kid   dka0.0.0.0.0     0       0    0              0       256000
000000be    nettest   ewa0.0.0.6.0    13       0    0          20888        20888
000000be    nettest   era0.0.0.7.0    13       0    0          17904        17904

>>>
```

test

The test command script sequentially tests the entire system.

Syntax

test [–lb]

Arguments

–lb
Specifies which external loopback tests are performed on the parallel port and the COM1 serial port.

Options

None

Description

The test command script sequentially tests the entire system, including memory, disks, tapes, diskettes, serial ports, parallel port, network, and graphic devices. All tests will execute 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 the test command can be executed at a time. The test command can be executed as either a background or foreground process.

The set command, used with environment variables, can establish parameters such as whether to halt, loop, or continue on error. See the Environment Variables for Diagnostic Commands section in this chapter for more information.

The –lb option allows an external loopback test to be performed on the parallel port and on the COM1 serial port. The passcount environment variable, d_passes, is ignored by the test command.

Note: The test command attempts to test the entire system, including graphic devices. Commands that exercise graphic devices require the terminal connected to the COM1 serial port be the default console terminal device.

To determine your default terminal device and to specify that the terminal attached to the COM1 serial port be your default device, use the following Alpha SRM console commands:

To do this function ...	Enter these commands...
Determine the default terminal device.	>>>**show console**
Set the terminal attached to the COM1 serial port to be the default console terminal device.	>>>**set console serial** >>>**init**

Examples

These commands run the test script in the background to sequentially test all system components and display the status of all executing diagnostic processes.

```
>>> test &

>>>show_status

      ID    Program        Device   Pass   Hard/Soft   Bytes Written   Bytes Read
  ------     ------         -----    ---   ---------          ------        -----
 00000001       idle        system      0      0    0               0            0
 0000009f    memtest        memory      0      0    0       100663296    100663296
 000000b4   exer_kid   dub0.0.0.1.0     0      0    0               0       444416
 000000b5   exer_kid   duc0.6.0.2.0   122      0    0               0      1249280
 000000b6   exer_kid   dud0.7.0.3.0   122      0    0               0      1249280
 000000b7   exer_kid   dka0.0.0.0.0     0      0    0               0       256000
 000000be    nettest   ewa0.0.0.6.0    13      0    0           20888        20888
 000000be    nettest   era0.0.0.7.0    13      0    0           17904        17904

>>>
```

Environment Variables for Diagnostic Commands

This section describes environment variables that are used to control the operational state of the diagnostic test and how to respond to error conditions.

Environment Variable Descriptions

Diagnostic Environment Variables

The following table shows optional Alpha SRM Console diagnostic environment variables and their descriptions. For a complete list, enter **show *** at the Alpha SRM Console prompt.

Variable	Description
d_bell	This variable specifies whether or not to ring the terminal bell if an error is detected. The possible values are: **off** = Do not ring the bell. The default is off. **on** = Ring the bell if an error is detected.
d_cleanup	This variable specifies whether or not cleanup code is executed at the end of a diagnostic command. The possible values are: **off** = Do not execute cleanup code. **on** = Execute cleanup code. The default is on.
d_complete	This variable specifies whether or not to display a message that the diagnostic command has finished. The possible values are: **off** = Do not display a completion message. The default is off. **on** = Display a completion message.
d_eop	This variable specifies whether or not to display end-of-pass messages. The possible values are: **off** = Do not display end-of-pass messages. The default is off. **on** = Display the end-of-pass messages.

(Continued on the next page)

(Continued)

Variable	Description
d_harderr	This variable specifies the action taken following hard error detection. The possible values are: **continue** = The command attempt to continue functioning until normal completion. **halt** = The command stops functioning. The default is halt. **loop** = The command continues to function.
d_oper	This variable specifies whether or not an operator is present. The possible values are: **off** = No operator present. The default is off. **on** = An operator is present.
d_passes	When used with the exer_write or exer_read command, this variable determines the number of passes executed by the command. When set to 0, the command is executed indefinitely . The default is 1 .
d_report	This variable specifies the depth of information provided by the diagnostic error reports. The possible values are: **summary** = Provide a brief summary of all errors. **full** = Provide a full and complete error report. **off** = Do not provide an error report.

(Continued on next page)

(Continued)

Variable	Description
d_softerr	This variable specifies the action taken following hard error detection. The possible values are: **continue** = The command attempts to continue functioning until normal completion. **halt** = The command stops functioning. The default is halt. **loop** = The command continues to function.
d_startup	This variable specifies whether or not to display the diagnostic startup message. The possible values are: **off** = Do not display the startup message. The default is off. **on** = Display the startup message.
d_trace	This variable specifies whether or not to display test trace messages. The possible values are: **off** = Do not display test trace messages. The default is off. **on** = Display the test trace messages.

Appendix A

Technical Support and Ordering Information

Technical Support

If you need technical support or help deciding which literature best meets your needs, call the Digital Semiconductor Information Line:

United States and Canada	**1–800 –332 –2717**
Outside North America	**+1 –508 –628 –4760**

Ordering Digital Semiconductor Products

To order Alpha microprocessors and evaluation boards, contact your local distributor. You can order the following semiconductor products from Digital:

Microprocessor and Peripheral Chips

Product	Order Number
DECchip 21030 PCI Graphics Accelerator	21030 –AA
DECchip 21040 Ethernet LAN Controller for PCI	21040 –AA
DECchip 21050 PCI –to–PCI Bridge	21050 –AA
DECchip 21052 PCI –to–PCI Bridge	21052 –AA
DECchip 21071 Core Logic Chipset	21071 –AA
DECchip 21072 Core Logic Chipset	21072 –AA
Alpha 21064A –233 Microprocessor	21064 –BB
Alpha 21064A –275 Microprocessor	21064 –DB
Alpha 21066 –166 Microprocessor	21066 –AA
Alpha 21066A –100 Microprocessor	21066 –CB
Alpha 21066A –233 Microprocessor	21066 –AB
Alpha 21164 –266 Microprocessor	21164 –AA
Alpha 21164 –300 Microprocessor	21164 –BA

Evaluation Board Kits

Evaluation board kits include a complete design kit, Windows NT installation kit, and an accessories kit with an evaluation board.

Product	Order Number
AlphaPC64 Evaluation Board Kit –275 MHz	21A02 –03
Alpha 21066A Evaluation Board Kit –233 MHz	21A03 –03
Alpha 21164 Evaluation Board Kit –266 MHz	21A04 –01

Design Kits

Design kits include full documentation and schematics. They do not include evaluation boards or related hardware.

Product	Order Number
Alpha 21064 Evaluation Board Design Kit	QR–21A01 –13
AlphaPC64 Evaluation Board Design Kit	QR–21A02 –13
Alpha 21066A Evaluation Board Design Kit	QR–21A03 –13
Alpha 21164 Evaluation Board Design Kit	QR–21A04 –11

Ordering Associated Literature

The following tables lists some of the available Digital Semiconductor literature. For a complete list, contact the Digital Semiconductor Information Line.

Title	Order Number
Alpha Architecture Reference Manual [1]	EY–L520E–DP–YCH
DECchip 21064 and DECchip 21064A Alpha AXP PCI Evaluation Board User's Guide	EC–N0640–72
Alpha 21164 Microprocessor Hardware Reference Manual	EC–QAEQB–TE
Alpha 21064 and Alpha 21064A Microprocessors Hardware Reference Manual	EC–Q9ZUB–TE
Alpha 21066, 21066A, and 21068 Microprocessors Hardware Reference Manual	EC–QC4GA–TE
Alpha 21066/21066A Microprocessors Data Sheet	EC–QC4HA–TE
Alpha 21066A Microprocessor Evaluation Board (EB66+) User's Guide	EC–QDVCB–TE
Alpha 21066A Microprocessor Evaluation Board (EB66+) Product Brief	EC–QDVEA–TE
AlphaPC64 Evaluation Board Product Brief	EC–QEZMC–TE
Alpha Microprocessors Evaluation Board Debug Monitor User's Guide	EC–QHUVB–TE
Alpha Microprocessors Evaluation Board Software Design Tools User's Guide	EC–QHUWA–TE
AlphaPC64 Evaluation Board User's Guide	EC–QGY2C–TE

[1] To order and purchase the *Alpha Architecture Reference Manual*, call **1–800–DIGITAL** from the U.S. or Canada, or contact your local Digital office or technical or reference bookstore where Digital Press books are distributed by Prentice Hall.

Index

D

E

L

M

N

O

P

R

S