Sun™ StorEdge™ A7000 ROM Monitor Reference Manual

THE NETWORK IS THE COMPUTER™

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900 USA
650 960-1300 Fax 650 969-9131

Part No. 805-4901-10
September 1998, Revision A

Send comments about this document to: `docfeedback@sun.com`

Contents

Code Samples

Tables

Preface

Sun StorEdge A7000 ROM Monitor Reference Manual describes the commands available with the ROM Monitor software on the Sun StorEdge A7000 Intelligent Storage Server System. The following information is included in this manual:

- Descriptions of starting and using the ROM Monitor
- Descriptions and formats of the commands in the various command groups
- Examples of user interaction with the ROM Monitor

How This Book Is Organized

Chapter 1 "Starting and Using the ROM Monitor" shows the console display at power up and reset. It describes the ROM Monitor's command line interpreter and the command line syntax format.

Chapter 2 "Node Control Commands" describes each node control command. It shows the command line format with examples of command variations.

Chapter 3 "Node Configuration Commands" describes node configuration commands stored in NVRAM on the Processor Module. Three node configuration commands affect DRAM. Each of these commands and their arguments are described in detail along with examples of command variations.

Chapter 4 "Memory Commands" describes each memory command and provides the command format with examples.

Chapter 5 "Miscellaneous Commands" describes the software development commands, disk/tape commands, the `test` command, and the remote support command. It shows each command format and examples.

Typographic Conventions

TABLE P-1 Typographic Conventions

Typeface or Symbol	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output.	Use the `help` command to display a list of commands. `Automatic test enabled`
`AaBbCc123`	What you type, when contrasted with on-screen computer output.	`ROM >>`**`readmem`** *address*
AaBbCc123	Book titles, new words or terms, words to be emphasized. Command-line variable; replace with a real name or value.	Read Chapter 2 in the *Console Operations Manual*. `ROM >>`**`readconfig slot`** *n* **`all`**
|	In command line syntax and system output displays, a vertical line indicates that a choice can be made between the values.	In the following example, either `enabled` or `disabled` will be displayed: `autotest {enabled|disabled}`
[ ]	In command line syntax or system output examples, brackets indicate optional values. If several values are placed inside brackets, any or none of them can be entered or displayed. Brackets are also used in system prompts to enclose the response choices.	In the following example, displaying the slot number is optional: `[Slot` *n*`:]`

( )	In command line syntax or system output examples, parenthesis around several values indicate that the enclosed values *must* be entered. If the values are separated by a vertical line then only one of the values must entered. If they are not separated by a vertical line, then all the values must be entered.	In the following example, either **`set`** or **`reset`** must be entered:
		`(`**`set`**`|`**`reset`**`)`
		In the following example, all the values must be entered:
		`(`*bus chan target lun*`)`

TABLE P-1 Typographic Conventions *(Continued)*

Typeface or Symbol	Meaning	Examples
`...`	In command line syntax and running text, a horizontal ellipsis indicates repetition or omission.	In the following example, one or more levels (*lvl*) can be entered or displayed: `levels selected` *lvl*`...`*lvl*

.
.
.
	The vertical ellipsis indicates continuation of system output.	In the following example, additional information would be displayed in place of the vertical ellipsis:
		`rms`
		`.`
		`.`
		`.`

`< >`	In examples of command input, an item surrounded by a greater than and less than sign must be replaced with an action. The greater than and less than signs are omitted when performing the action.	`<CR>` means press the Return key.

Related Documentation

For more information on the operation of the console, refer to the *Console Operations Manual* provided with your system.

Sun Documentation on the Web

The `docs.sun.com` web site enables you to access Sun technical documentation on the Web. You can browse the `docs.sun.com` archive or search for a specific book title or subject at:

```
http://docs.sun.com.
```

Sun Welcomes Your Comments

We are interested in improving our documentation and welcome your comments and suggestions. You can email your comments to us at:

`smcc-docs@sun.com`.

Please include the part number of your document in the subject line of your email.

CHAPTER **1**

Starting and Using the ROM Monitor

The Processor Module board includes a ROM Monitor with a built-in interactive user and operator console display interface. This interface lets you access control and utility functions to configure, test, and boot your system. You control these functions through command line entries.

The chapter topics are presented in the following order:

- Starting the ROM Monitor—page 1-1
- ROM Monitor Command Line Interpreter—page 1-7

Starting the Rom Monitor

The ROM Monitor starts automatically upon power up. It may also be restarted by either pressing the reset switch on the CPU front panel or using the `reset node` command.

Reset / Powerup

When a Processor Module board powers up or is reset, a series of confidence tests is always run. The Console then displays the monitor name, copyright statement, and several Processor Module board parameters. (You can also display these parameters after power up by using the `readconfig` command.)

After this display, you have the option to run the next level of tests provided the tests are enabled. If the tests are enabled, the following message is displayed:

```
Automatic test enabled, 3 seconds to break . . .
```

Press any key to bypass the built-in automatic self-test portion of the power up sequence.

If you do not press a key, the monitor executes the tests and displays results before proceeding.

If autoboot is enabled, you have the option to boot, or not, when the following message is displayed:

```
You have 7 seconds to terminate the autoboot sequence . . .
```

Press any key to terminate the autoboot sequence. If you do not terminate the autoboot sequence, the `sysboot` utility boots automatically. For more information on `sysboot`, refer to your operating system's *Software Support* or *Administrator Guide*.

If autoboot is disabled, control passes to the Command Line Interpreter (CLI) and the Console displays the following ROM Monitor prompt:

```
ROM >>
```

Note – Power up clears expansion memory contents.

A secondary boot device may be defined and enabled in case the primary boot device fails. You can also boot from a VME SCSI disk with a two step boot procedure using `setconfig netboot`. (See the `setconfig netboot` command in Chapter 3 for more information.)

You can also boot directly from a VME SCSI disk by setting the proper parameters using the `setconfig bdevice` command. (See the `setconfig bdevice` command in Chapter 3 for more information.)

Reset/Power Up Display

CODE EXAMPLE 1-1 shows what the console typically displays after the Processor Module board resets, the system powers up, or the operator issues a `reset node` command. The example assumes the autoboot capability is enabled.

The example system includes no expansion memory and four CPUs, with no errors detected. The actual sequence and display vary between systems and monitor revisions. If autoboot is enabled, sysboot-specific messages display.

CODE EXAMPLE 1-1 Reset/Power Up Display

```
********* 88110 ROM Monitor **********
Copyright 1998 Sun Microsystems, Inc. All rights reserved.
It is Thu May 20 15:38:48 GMT 1998
Part: 531_103422_200, RCS: V2_00, Created: Tue May 5 14:11:18 EDT
1998
Processor Brd Serial Number: 98123456, Assembly ID 0x0534,
Functional Rev B
Adaptor Brd Serial Number: 93210417, Assembly ID 0x0534,
Functional Rev A
Memory Brd 1 Serial Number: 96510541, Assembly ID 0x210, Assembly
Rev H00
Ethernet Address: 08:00:4c:a1:08:38
Total Processor Brd memory 32 Mbytes
Total Adaptor Brd memory 0 Mbytes
Total expansion memory 512 Mbytes
Common Console Interface not present
<<< Running with Instruction Cache Enabled >>>
Starting cpu 1
Starting cpu 2
Starting cpu 3
Clearing Processor Brd memory...
Clearing expansion memory...
Searching for Scsi Devices on Channel A:
......................................
Device: 0 0 5 0, disk drive
Device: 0 0 6 0, disk drive

Automatic test enabled, 3 seconds to break
```

Note – Pressing any key at this point bypasses the built-in self tests.

```
Slot 1: Running Built In Tests

Test levels selected:  core  qmem  slots

Running Tests
Starting EPROM checksum test:
    Testing new PROMs
        CPU 0, Read checksum 29b81b8e calculated 29b81b8e...PASS
Starting Parameter prom checksum test:
        CPU 0, calc checksum fffffb7e, stored fffffb7e...PASS
Starting DRAM Parity test:
        Bypassed for this configuration
        - no Adaptor Memory present
Starting SRAM Parity test:
...PASS
Starting TS80 X-cpu int test:
        CPU 1 interrupting: CPU 0, CPU 1, CPU 2, CPU 3...PASS
        CPU 3 interrupting: CPU 0, CPU 1, CPU 2, CPU 3...PASS
        CPU 2 interrupting: CPU 0, CPU 1, CPU 2, CPU 3...PASS
        CPU 0 interrupting: CPU 0, CPU 1, CPU 2, CPU 3...PASS
Starting TS80 xint test:
        CPU 1 testing xint 0, 1, 2, 3, 4, 5, 6, 7, ...PASS
        CPU 0 testing xint 0, 1, 2, 3, 4, 5, 6, 7, ...PASS
        CPU 3 testing xint 0, 1, 2, 3, 4, 5, 6, 7, ...PASS
        CPU 2 testing xint 0, 1, 2, 3, 4, 5, 6, 7, ...PASS
Starting TS80 int cpu test:
        CPU = 1, Timer: 0, 1, 2, 3, 4, 5...PASS
        CPU = 0, Timer: 0, 1, 2, 3, 4, 5...PASS
        CPU = 3, Timer: 0, 1, 2, 3, 4, 5...PASS
        CPU = 2, Timer: 0, 1, 2, 3, 4, 5...PASS
Starting TS80 Timer int test:
        CPU: 1, timer 0, 1, 2, 3, 4, 5...PASS
        CPU: 0, timer 0, 1, 2, 3, 4, 5...PASS
        CPU: 3, timer 0, 1, 2, 3, 4, 5...PASS
        CPU: 2, timer 0, 1, 2, 3, 4, 5...PASS
Starting Ethernet controller chip test:
...PASS
Starting Ethernet internal loopback test:
...PASS
Starting Ethernet external loopback test:
...PASS
Starting VME intr bsy test:
...PASS
Starting VME interrupt test:
        CPU 0 testing vme interrupt: 1 2 3 4 5 6 7 ...PASS
```

```
Starting SCC internal loopback test:
    port: 0 ....RESERVED (console port)
    port: 1 ....PASS
    port: 2 ....PASS
    port: 3 ....PASS
Starting SCSI regs test:
CPU 0 reading icode istat dmastat stat0 stat2 ID ...PASS
Starting System Bus Parity test:
...PASS

Starting DRAM DATA=ADDR test:
    Bypassed for this configuration
    - no Adaptor Memory present
Starting DRAM address line test:
    Bypassed for this configuration
    - no Adaptor Memory present
Starting Multi CPU fairness test:
    Bypassed for this configuration
    - no Adaptor Memory present
Starting Multi Data Size test:
    Bypassed for this configuration
    - no Adaptor Memory present
Starting DRAM DATA=ADDR test:
    DRAM DATA=ADDR 3e100000-40000000 with 4 cpus
        cpu 0 testing 8126464 bytes starting at 3e100000
        cpu 1 testing 8126464 bytes starting at 3e100000
        cpu 2 testing 8126464 bytes starting at 3e100000
        cpu 3 testing 8126464 bytes starting at 3e100000
        fill, chk...PASS
Starting DRAM address line test:
    DRAM address line BYTE, from 3e100000-40000000...PASS
    DRAM address line WORD, from 3e100000-40000000...PASS
    DRAM address line LONG, from 3e100000-40000000...PASS
Starting Multi CPU fairness test:
    CPU 0, 99984, CPU 1, 100000, CPU 2, 100000, CPU 3, 100000, PASS
Starting Multi Data Size test:
    Multi Data Size 3e100000-3e200000 with 4 cpus
    (lfill-bchk), (bfill-lchk), (lfill-wchk), (wfill-lchk),
    (lfill-dchk), (dfill-lchk), (bfill-wchk), (wfill-bchk),
    (bfill-dchk), (dfill-bchk), (wfill-dchk), (dfill-wchk)...PASS

Testing Memory Cards
Starting Expansion Memory Parameter prom checksum test:
...PASS
```

```
Starting DRAM DATA=ADDR test:
    DRAM DATA=ADDR 40000000-60000000 with 4 cpus
        cpu 0 testing 134217728 bytes starting at 40000000
        cpu 1 testing 134217728 bytes starting at 48000000
        cpu 2 testing 134217728 bytes starting at 50000000
        cpu 3 testing 134217728 bytes starting at 58000000
        fill, chk...PASS
Starting DRAM address line test:
    DRAM address line BYTE, from 40000000-60000000...PASS
    DRAM address line WORD, from 40000000-60000000...PASS
    DRAM address line LONG, from 40000000-60000000...PASS
Starting VBUS DATA=ADDR test:
    VBUS DATA=ADDR 40000000-60000000 with 4 cpus
     Master Page: 2
        cpu 0 testing 134217728 bytes starting at c0000000
        cpu 1 testing 134217728 bytes starting at c8000000
        cpu 2 testing 134217728 bytes starting at d0000000
        cpu 3 testing 134217728 bytes starting at d8000000
        fill, chk...PASS
Starting Multi CPU fairness test:
    CPU 0, 99992, CPU 1, 100000, CPU 2, 100000, CPU 3, 100000, PASS
Starting Multi CPU Vbus fairness test:
    CPU 0, 99983, CPU 1, 100000, CPU 2, 100000, CPU 3, 100000, PASS
Starting Expansion Memory Parity test:
    Bypassed for this configuration
    No expansion memory board configured.
Starting Expansion Memory Interleave test:
    Bypassed for this configuration
    No expansion memory board configured.
Starting Multi Data Size test:
    Multi Data Size 40000000-40100000 with 4 cpus
    (lfill-bchk), (bfill-lchk), (lfill-wchk), (wfill-lchk),
    (lfill-dchk), (dfill-lchk), (bfill-wchk), (wfill-bchk),
    (bfill-dchk), (dfill-bchk), (wfill-dchk), (dfill-wchk)...PASS

Testing Backplane Cards
No extended diagnostics configured
You have 7 seconds to terminate the autoboot sequence.  .  .
ROM >>
```

Note – Press any key to bypass the autoboot sequence.

ROM Monitor Command Line Interpreter

The user interface includes a Command Line Interpreter (CLI) that displays the following prompt after power up or reset:

```
ROM >>
```

The CLI accepts commands from six ROM Monitor command groups. Each group and its related commands are listed in the following tables:

- TABLE 1-1 lists the Node Control commands and the `help` command
- TABLE 1-2 lists the Node Configuration commands
- TABLE 1-3 lists the Memory commands
- TABLE 1-4 lists the Miscellaneous commands

TABLE 1-1 Node Control Commands and `help` Command (see Chapter 2)

Command	Function
`altboot`	Execute node boot procedure from alternate device
`boot`	Execute node boot procedure from default device
`burnin`	Read, enable, or disable node burnin status flag
`date`	Read or set date
`errorlog`	Clear, show, or report memory-based error log
`get`	Transfer file from a network host to memory
`help`	Display all commands or specific command format
`listimages`	List node images
`loadimages`	Load a specific node image
`nvlog`	Clear, show, or report node's nonvolatile random-access memory (NVRAM)-based error log
`nvram`	Accept | default | save | restore node configuration data in NVRAM
`port`	Set or read state of specified serial port

TABLE 1-1 Node Control Commands and `help` Command (see Chapter 2) *(Continued)*

Command	Function
`rawboot`	Execute node boot procedure from specified device
`reset`	Reset node or SCSI channel
`transparent`	Communicate over console serial port

TABLE 1-2 Node Configuration Commands (see Chapter 3)

Command	Function
`altbdevice`	Set or read alternate autoboot configuration device
`altbimage`	Set or read alternate autoboot image name
`altnetboot`	Set or read alternate netboot state
`autoboot`	Set or read autoboot device
`autotest`	Set or read autotest state
`bdevice`	Set or read autoboot configuration device
`bimage`	Set or read autoboot image name
`boardid`	Read Processor Module board identification (ID)
`btimeout`	Set or read autoboot timeout value
`clockcal`	Set or read time-of-day clock calibration value
`cpu`	Set or read node CPU configuration
`eadrs`	Read node Ethernet address
`errport`	Set or read error port number
`estate`	Set or read environmental monitor state flag
`memconfig`	Set or read expansion memory configuration values (this command contains subarguments for base addresses, interleave factor, and other information)
`msize`	Read node's DRAM size
`netboot`	Set or read netboot state
`printerr`	Set or read error port state
`revision`	Read Processor Module board and Adaptor board revision levels (this command is also used to set the Processor board's revision level)
`scsibus`	Read SCSI bus device types and logical unit numbers

TABLE 1-2 Node Configuration Commands (see Chapter 3) *(Continued)*

Command	Function
`scsichanA/B`	Set or read on-board SCSI channel
`scsidelay`	Set or read SCSI bus device initialization delay time
`scsiid`	Set or read SCSI bus channel ID
`scsimaskA/B`	Set or read SCSI masked out devices
`scsiresetA/B`	Set or read issuance of SCSI resets
`scsitimeout`	Set or read SCSI timeout value
`scsistart`	Manually initialize the SCSI subsystem
`serial`	Set or read Processor Module board serial ports
`show`	Read current system configuration and status information
`slot`	Set or read slot configuration values (this command contains subarguments for board type, board state, control/status registers, diagnostics, and other information)
`snumber`	Set or read Processor Module board serial number
`sysid`	Set or read system ID number
`tlevel`	Set or read autotest level
`version`	Read ROM Monitor firmware version from ROM
`vmeconfig`	Set or read VME values (this command contains subarguments for address modifier bits, master/slave page bits, and other information)

TABLE 1-3 Memory Commands (see Chapter 4)

Command	Function
`comparemem`	Compare two memory ranges
`copymem`	Copy memory from range to range
`debug`	Special memory sequence debugger
`disassemble`	Disassemble memory
`fillmem`	Fill memory
`interleave`	Configure memories in interleave mode
`memtest`	Run built-in memory tests on a range
`modmem`	Modify memory

TABLE 1-3 Memory Commands (see Chapter 4) *(Continued)*

Command	Function
`readmem`	Read memory
`search`	Search a memory range for a pattern
`string`	Search a memory range for a character string
`uninterleave`	Reconfigure the memories uninterleaved

TABLE 1-4 Miscellaneous Commands (see Chapter 5)

Command	Function	Type
`go`	Send specified processor to address	Software Development
`mgo`	Send all processors to address	Software Development
`srecord`	Load S-records	Software Development
`readblock`	Read logical block(s)	Disk/Tape
`rewind`	Rewind tape drive	Disk/Tape
`scsirwc`	SCSI read/write/compare test	Disk/Tape
`skipfilem`	Skip file mark(s)	Disk/Tape
`taperwc`	Tape read/write/compare test	Disk/Tape
`writeblock`	Write logical block(s)	Disk/Tape
`test`	Execute built-in self tests (BIST)	Test
#	Comment command	Remote Support

Command Line Format

All commands consist of a text string followed by a carriage return, in the general form:

```
command argument argument [options]<CR>
```

The ROM Monitor checks all entries for validity; invalid entries produce an error message.

Command line entries are subject to the following rules:

- The CLI requires only enough input to avoid ambiguity. If less is supplied, a "`_____ is ambiguous`" message appears, along with a list of possible choices. In the formats given in this manual, minimum input required is shown in **`boldface type`**.
- Command arguments can be required or optional, according to the command. Required arguments follow the command and are separated by a space.
- Tokens on the command line must be separated by white space, which consists of spaces, tabs, or newlines.
- The carriage return (or Enter) key (<CR>) enters the command and terminates input.
- Numeric values are in decimal unless preceded by 0x (hex) or 0b (binary).

Arguments Entered as Expressions

Numeric arguments on the command line can be entered as an expression. The expression can be any group of decimal or hexadecimal argument values enclosed in parentheses and separated by the operator `+`, `-`, `*`, `/`, `<`, or `>`.

The operators act on values traditionally as shown in TABLE 1-5.

TABLE 1-5 Argument Value Operators

Operator	Action
(A) + (B)	add B to A
(A) - (B)	subtract B from A
*(A) * (B)*	multiply A by B

TABLE 1-5 Argument Value Operators *(Continued)*

Operator	Action
(A) / (B)	divide A by B
(A) < (B)	shift A left B places
(A) > (B)	shift A right B places

Recalling Previously-entered Commands

There are two ways to recall previously entered commands. The first way is from a buffer that stores the most recently entered eight commands. You can recall and display the last command entered to the console display by typing Control-L at the `ROM >>` prompt. Typing Control-L again displays the next-to-last command. Repeating this action displays up to the last eight commands, and cycles through the same eight commands again.

Note – If you have entered fewer than eight commands, the empty spaces in the buffer display as blank lines.

The second way is to execute the last command continuously from the command line. This is done by preceding the command with an asterisk and space.

```
* command <CR>
```

CODE EXAMPLE 1-2 Executing the Last Command Continuously

```
ROM >>* readm l 0x40000000<CR>
 0x40000000: 00010203

Depress escape key to terminate!
 0x40000000: 00010203

Depress escape key to terminate!
 0x40000000: 00010203

Depress escape key to terminate!
 0x40000000: 00010203

Depress escape key to terminate!
 0x40000000: 00010203

Depress escape key to terminate!
 0x40000000: 00010203

Depress escape key to terminate!
 0x40000000: 00010203

Depress escape key to terminate!
 0x40000000: 00010203

Depress escape key to terminate!
 0x40000000: 00010203

Depress escape key to terminate!
 0x40000000: 00010203

Depress escape key to terminate!

ROM >>ESC

ROM >>
```

Interrupting Command Execution

You can interrupt any Monitor operation by typing Control-C. The operation in progress terminates, the CPU controlling I/O returns the `ROM >>` prompt, and the other CPUs are forced to the idle state. Control-C also clears the commands buffer.

Caution – Use this feature carefully; you could lose data from an I/O transfer.

Acknowledging Commands

Every issued command returns at least a prompt. Some commands also return response data. In the examples in this manual, the acknowledgment is the `ROM >>` prompt unless control passes to another program, such as a diagnostic. In that case, command acknowledgment is program-dependent.

CHAPTER **2**

Node Control Commands

This chapter describes each node control command listed in TABLE 2-1. Chapter 1, *Starting and Using the ROM Monitor,* describes command line entry conventions. Typing `help` or `?` at the `ROM >>` prompt displays all currently available and valid commands. Entering `help` *commandname*, where *commandname* is a valid command, displays the command format.

TABLE 2-1 Node Control Commands

Command	Function
`altboot`	Execute the node boot procedure from the alternate device
`boot`	Execute the node boot procedure from the default device
`burnin`	Read, enable, or disable the node burnin status flag
`date`	Read or set the date
`errorlog`	Clear, show, or report about the memory-based error log
`get`	Transfer a file from a network host to memory
`help`	Display all commands or a specific command format
`listimage`	List node images
`loadimage`	Load a specific node image
`nvlog`	Clear, show, or report about the node's nonvolatile random-access memory (NVRAM)-based error log
`nvram`	Accept, default, save or restore node configuration data in NVRAM
`port`	Open or close a serial port
`rawboot`	Execute the node boot procedure from the specified device
`reset`	Reset a node or SCSI channel
`transparent`	Communicate over the console serial port

`altboot` - Alternate Boot Node

Executes the altboot procedure from the secondary device. This command is useful if a failure occurs on the primary device. If you specify a valid image, the system boots from it. If no image name is specified, the system boots from the default alternate boot image (normally `sysboot`). The response is to pass control to the boot program. For more information on `sysboot`, refer to your operating system's *Software Support* or *Administrator Guide*.

The altboot process can load the image from a local SCSI disk, a VME SCSI disk, or the console disk over the administration bus (ethernet). See `setconfig` parameters `altbdevice` and `altnetboot` in Chapter 3, "Node Configuration Commands," for more information.

If an error is detected during booting, the following message and operator option appear:

```
WARNING: an error has been detected during system initialization
Continue with boot operation? (y/n)
```

Respond by entering `y` to continue or `n` to abort the boot process.

Format

`altbo``ot` [*imagespec*]

Arguments

Argument	Description
imagespec	ASCII string representing the image name (sysboot, sysex,...)

Examples

```
ROM >>altboot <CR>
Auto boot (sysboot.110) from 0 0 6 0
Header Partition Image Rev 3.3.0 for OS: Umax 4.2

  Loading COFF file from root image at 0 0 6 0 0 /unix

NOTICE: vsi_init: master_page 2, will be forced to 3

NOTICE: lbus hold time set to 40ns

NOTICE: VME Requester Mode RWD-Not Fair, Priority, Level 3, Master
Page 0x3

NOTICE: 2GB Localbus support present
```

When an `altboot` command is issued with `netboot` enabled, the following message appears.

```
ROM >>altboot <CR>
Auto altnetboot (sysboot.110) from 0x3fc00000
```

`boot` - Boot Node

Execute boot procedure from the default device. If you specify a valid image, the system boots from it. If no image name is specified, the system boots from the default boot image (normally sysboot). The response is to pass control to the boot program. For more information on sysboot, refer to your operating system's *Software Support* or *Administrator Guide*.

The boot process can load the image from a local SCSI disk, a VME SCSI disk, or the console disk over the administration bus (ethernet). See `setconfig` parameters `bdevice` and `netboot` in Chapter 3, "Node Configuration Commands," for more information.

If an error was detected during system initialization prior to booting, the following message and operator option appear:

```
WARNING: an error has been detected during system initialization
Continue with boot operation? (y/n)
```

Respond by entering `y` to continue or `n` to abort the boot.

Format

`bo``ot` [*imagespec*]

Arguments

Argument	Description
imagespec	ASCII string representing the image name (sysboot, sysex,...)

Examples

```
ROM >>**boot** <CR>
Auto boot (sysboot.110) from 0 0 6 0
Header Partition Image Rev 3.3.0 for OS: Umax 4.2

  Loading COFF file from root image at 0 0 6 0 0 /unix

NOTICE: vsi_init: master_page 2, will be forced to 3

NOTICE: lbus hold time set to 40ns

NOTICE: VME Requester Mode RWD-Not Fair, Priority, Level 3, Master
Page 0x3

NOTICE: 2GB Localbus support present
Memory card in slot 3 assumed to be EDRAM

NOTICE: UNIX(R) System V Release 3.6.0.11U3.2: unix Version
3.6.0.11U3.2

NOTICE: Machine 88k Total real memory 131072 KB

Starting CPU #1
Starting CPU #3
Starting CPU #2
```

When a `boot` command is issued with `netboot` enabled, this message appears:

```
ROM >>**boot** <CR>
Auto boot (sysboot.110) from 0x3fc00000
```

burnin - Read, Enable, or Disable Node Burn In Status Flag

Set or reset a flag that software uses to tell whether a node is being burnt in. When the burnin flag is set (enabled), test command execution loops until a key is pressed.

Type burnin<CR> to read the current node state.

Format

burnin [**e**nable | **d**isable]

Arguments

Argument	Description
enable	Set the burnin flag; loop tests until a key is pressed
disable	Reset the burnin flag; tests do not loop

Examples

```
ROM >>burnin<CR>
system burnin: enabled
ROM >>

ROM >>burnin disable<CR>
system burnin: disabled
ROM >>
```

`date` - Set or Read Date

Set or read the real-time clock's date and time. Type `date` without arguments to read the current date and time in 24 hour format, relative to Greenwich mean time (GMT).

Format

`da``te` [*year month day hour*:*minute* [:*seconds*]]

Arguments

Argument	Description
year	Two-digit number for the current year; valid range is 00 to 99
month	Two-digit number for the current month (January = 01, February = 02,...); valid range is 01 through 12
day	Two-digit number for the day of the month; valid range is 00 through 31 (as appropriate for length of the *month* selected)
hour:*minute*:*seconds*	*hour* is the two-digit number for current hour; valid range is 00 through 23 (00 = midnight, 23 = 11:00 PM) *minute* is the two-digit number for current minutes past the hour; valid range is 00 through 59 *second* is the two-digit number for current seconds past the minute; valid range is 00 through 59

Examples

```
ROM >>date<CR>
Tue Apr 04 04:35:21 GMT 1995
ROM >>

ROM >>date 95 05 03 16:24<CR>
ROM >>date<CR>
Wed Apr 05 16:24:06 GMT 1995
ROM >>
```

`errorlog` - Clear, Show, or Report Memory-based Error Log

Clear, show, or report memory-based error log values. The `errorlog` command with the `report` option prints a screenful of messages from the error log until the error messages are exhausted. You are prompted to continue after each screenful is presented. This action also causes the last inspected field of subsequent error messages to be updated so the operator can detect whether messages have scrolled by. This can be done by comparing the *error_log_index* and the *last_inspected_field* shown as follows:

```
Error[error_log_index-last_inspected_field]: Slot 1 CPU 0: message
```

In the following example the operator detects that 10 messages have been logged since the last inspection by the `errorlog` command. After this inspection takes place, the *last_inspected_field* in the example would be 100—the index of the last error message logged.

```
Error[100-90]: Slot 1 CPU 0: Device 1 6 0 failed read capacity
operation
```

Format

er`rorlog` (**c**`lear` | **s**`how` | **r**`eport`) [*index*]

Arguments

Argument	Description
c`lear`	Erase or clear the error log, returning the monitor prompt
sh`ow`	Display the number of entries in the error log
r`eport`	Show the specified entry or all entries in the error log
index	A numerical index of the error log; default index number is 1. All messages after this index are displayed.

Examples

```
ROM >>errorlog show<CR>

Serial Number 93253f3ec     Revision G04
Log Initialized Mon Mar 13 16:28:20 GMT 1995
5 Log Entries     5 Errors Reported

ROM >>
ROM >>errorlog report<CR>

Serial Number 93253f3ec     Revision G04
Log Initialized Mon Mar 13 16:28:26 GMT 1995
5   Log Entries     5 Errors Reported
Error 1  Node 1  Slot 1  Mon Mar 13 16-28-28 GMT 1995
Device 1 3 0 failed read capacity operation

Error 2  Node 1  Slot 1  Mon Mar 13 16-28-30 GMT 1995
Device 1 4 0 failed read capacity operation

Error 3  Node 1  Slot 1  Mon Mar 13 16-28-32 GMT 1995
Device 1 5 0 failed read capacity operation

Error 4  Node 1  Slot 1  Mon Mar 13 16-28-34 GMT 1995
Device 1 6 0 failed read capacity operation

More ? (Y/N)  y

Error 5  Node 1  Slot 1  Mon Mar 13 16-28-39 GMT 1995
Failed to find sysboot on device 0 6 0, (use listimage?)
ROM >>errorlog report 2<CR>

Serial Number 93253f3ec     Revision G04
Log Initialized Mon Mar 15 16:28:41 GMT 1993
5 Log Entries     5 Errors Reported

Error 2  Node 1  Slot 1  Mon Mar 13 16-28-43 GMT 1995
Couldn’t start CPU 2
```

```
Error 3  Node 1  Slot 1  Mon Mar 13 16-28-45 GMT 1995
Couldn’t start CPU 2-

Error 4  Node 1  Slot 1  Mon Mar 13 16-28-47 GMT 1995
Couldn’t start CPU 3
Error 5  Node 1  Slot 1  Mon Mar 13 16-28-49 GMT 1995
Couldn’t start CPU 3-


ROM >>errorlog clear<CR>
ROM >>
```

`get` - Get File from Network Host

Transfer a named file from a network host machine to memory. The UMAX V `tftp` utility performs the transfer. The command response is the completion status and number of bytes received.

Note – The `tftp` utility may vary between UMAX V and other UNIX systems. Consult the `tftp` man page on the transferring host for details.

Format

get *hostIPaddr filename loadaddr*

Arguments

Argument	Description
hostIPaddr	IP address of the system storing the file
filename	ASCII name of the file
loadaddr	Starting memory address to load the file into

Example

The following example is for a system start.

```
ROM >>get 129.154.44.59 mc4.image 0x3f600000<CR>
RARP successful, IPaddr 129.154.044.112
........
Transfer Complete: 275328 bytes received
```

The following example is for a system reset.

```
ROM >>get 129.154.044.112 mc4.image 0x3f600000
........
Transfer Complete: 275328 bytes received
```

`help` - Display Command Help

Display all commands or the format of a specific command. Type `help` or `?` to display all commands alphabetically. Type `help` *command* to display the format of command name *command*.

If you enter an invalid command name, `help` displays all commands alphabetically.

Format

`help` [*command*]

`?`

Arguments

Argument	Description
command	Valid command name

Examples

```
ROM >>help<CR>
Available Commands:
(an alphabetical list of all commands is displayed)
ROM >>
ROM >>help setconfig bdevice<CR>
Command format: setconfig bdevice bus chan target lun [partindex]
ROM >>
```

`listimage` - List Node Images

List accessible node images resident in the header partition of the specified device. The command response is a list of valid images.

Format

`listimage` [*bus chan target lun*]

Arguments

Argument	Description	
bus	Index or a hex value of a valid VME short address	
	Index Number	**Description**
	0	On-board SCSI
	1	VME Controller 0x6800
	2	VME Controller 0x7000
	3	VME Controller 0x7800
	.	
	.	
	.	
	15	VME Controller 0xd800
chan	SCSI channel 0 or 1	
target	SCSI target device 0 through 7	
lun	SCSI logical device 0 through 7	

Note – Only channel 0 is supported when using VME SCSI boot. Make sure the boot device is on channel 0 under this configuration.

Examples

```
ROM >>listimage 0 0 6 0<CR>
Name: sysboot, Image entry: 3fc013f0
Name: sysboot.110, Image entry: 3fc014c0
Name: escondiag, Image entry: 3f600000
Name: mc4diag, Image entry: 3f600000
ROM >>

ROM >>listimage<CR>
Name: sysboot, Image entry: 3fc013f0
Name: sysboot.110, Image entry: 3fc014c0
Name: escondiag, Image entry: 3f600000
Name: mc4diag, Image entry: 3f600000
ROM >>
```

`loadimage` - Load Node Image

Load node image from the header partition of the specified device. Response is a completion status message. If no device is specified, the `bdevice` parameters are used.

Format

`lo``adimage` (*imagespec*) [*bus chan target lun*]

Arguments

Argument	Description	
imagespec	ASCII string representing the image name (rmsdiag, sysboot, sysex,...)	
bus	Index or a hex value of a valid VME short address	
	Index Number	**Description**
	0	On-board SCSI
	1	VME Controller 0x6800
	2	VME Controller 0x7000
	3	VME Controller 0x7800
	.	
	.	
	.	
	15	VME Controller 0xd800
chan	SCSI channel 0 or 1	
target	SCSI target device 0 through 7	
lun	SCSI logical device 0 through 7	

Note – Only channel 0 is supported when using VME SCSI boot. Make sure the boot device is on channel 0 under this configuration.

Example

```
ROM >>**loadimage qbnmcdiag 0 1 6 0**<CR>
qbmcdiag: entry point 3f600000
File addr = 29f000 Load addr = 3f600000 Size = 56392
File addr = 2ace00 Load addr = 3f6026d0 Size = 493552
ROM >>
```

`nvlog` - Clear, Show, or Report Node NVRAM-based Error Log

Clear, show, or report node-based error log values.

Format

nv`log (`**c**`lear | `**s**`how | `**r**`eport) [`*index*`]`

Arguments

Argument	Description
c`lear`	Erase or clear the error log, returning the monitor prompt
r`eport`	Show the specified entry or all entries in the error log
s`how`	Display the number of entries in the error log
index	Numerical index of the error log; default index number is 1

Examples

```
ROM >>nvlog show<CR>
Serial Number 96A10057   Revision D00
Non Volatile Ram Log Initialized Tue Aug 04 13:07:48 GMT 1998
3 Log Entries     3 Errors Reported
ROM >>nvlog report<CR>
Serial Number 96A10057   Revision D00
Non Volatile Ram Log Initialized Tue Aug 04 13:07:48 GMT 1998
3 Log Entries     3 Errors Reported
Error 1  Node 1  Slot 1  Tue Aug 04 13-07-48 GMT 1998
Device 1 6 0, failed read capacity operation

More ? (Y/N)  y
```

```
Error 2  Node 1  Slot 1  Tue Aug 04 13-07-48 GMT 1998
Failed to find sysboot on device 0 6 0, (use listimage?)

More ? (Y/N)  y
Error 3  Node 1  Slot 1  Tue Aug 04 13-07-48 GMT 1998
CPU 0: MBUS data exception, address 0x0003000C
ROM >>nvlog report 2<CR>
Serial Number 96A10057   Revision D00
Non Volatile Ram Log Initialized Tue Aug 04 13:07:48 GMT 1998
3 Log Entries     3 Errors Reported

Error 2  Node 1  Slot 1  Tue Aug 04 13-07-48 GMT 1998
Failed to find sysboot on device 0 6 0, (use listimage?)
ROM >>nvlog clear<CR>
```

`nvram` - Accept | Default | Restore | Save Node Configuration Data in NVRAM

Save or change configuration data. You can abort this command and accept previously stored values. After you enter the command and arguments, the monitor prompts you to confirm that you want to make the requested change. Press `y` to confirm the modification; press `n` or any other key to abort before saving.

Caution – If `nvram accept` is not executed, changes made by the configuration command `setconfig` are valid only until the next power up sequence or system reset. Refer to Chapter 3, *Node Configuration Commands*, for more information about `setconfig`.

Note – After the NVRAM is modified, the monitor verifies the checksum and displays the following warning if the checksum is not valid:

```
WARNING: bad nvram checksum detected
```

Format

nvram (**a**ccept | **d**efault | **r**estore | **s**ave) [*bus* | *chan* | *targ* | *lun*]

Arguments

Argument	Description	
accept	Initialize NVRAM with the current configuration values	
default	Initialize NVRAM with the default configuration values	
restore	Initialize NVRAM with configuration values previously saved to tape	
save	Save current NVRAM configuration values to tape	
bus	Index or a hex value of a valid VME short address	
	Index Number	**Description**
	0	On-board SCSI
	1	VME Controller 0x6800
	2	VME Controller 0x7000
	3	VME Controller 0x7800
	.	
	.	
	.	
	15	VME Controller 0xd800
chan	SCSI channel 0 or 1	
target	SCSI target device 0 through 7	
lun	SCSI logical device 0 through 7	

Note – Only channel 0 is supported when using VME SCSI boot. Make sure the boot device is on channel 0 under this configuration.

Examples

```
ROM >>nvram accept <CR>
Are you sure you want to save the configuration? (y/n) y
nvram modified
ROM >>
ROM >>nvram default <CR>
Are you sure you want to erase/default the configuration? (y/n) y
nvram modified
ROM >>
ROM >>nvram save 0 0 3 0 <CR>
Nvram successfully saves to 0 0 3 0
ROM >>
ROM >>nvram restore 0 0 3 0 <CR>
Nvram successfully restored from 0 0 3 0
ROM >>
```

`port` - Serial Port State

Set or read the state of the serial port specified.

Caution – The state of serial port 0 (the console) can not be modified. If an attempt is made to issue a command to a closed serial port, an error message is generated and the command aborted.

Format

`port` *port#* `[`**`o`**`pen |`**`c`**`lose]`

Arguments

Argument	Description
port#	Serial port number. Valid range is 1 through 3.
`o``pen`	Enable port configured and enabled (this is the default)
`c``lose`	Serial port deconfigured and disabled

Examples

```
ROM >>port 1<CR>
serial port 1: open
ROM >>

ROM >>port 1 close<CR>
ROM >>port 1<CR>
serial port 1: closed
ROM >>
```

`rawboot` - Rawboot Node

Execute boot procedure from a specified disk. If you specify a valid image, the system boots from it. If no image name is specified, the system boots from the default boot image (normally sysboot). The response is to pass control to the boot program. For more information on sysboot, refer to your operating system's *Software Support* or *Administrator Guide*.

If an error was detected during system initialization prior to booting, the following message and operator option appear:

```
WARNING: an error has been detected during system initialization
Continue with boot operation? (y/n)
```

Respond by typing `y` to continue or `n` to abort the boot.

Format

`ra``wboot` *bus chan target lun* [*imagespec*]

Arguments

Argument	Description	
bus	Index or a hex value of a valid VME short address	
	Index Number	**Description**
	0	On-board SCSI
	1	VME Controller 0x6800
	2	VME Controller 0x7000
	3	VME Controller 0x7800
	.	
	.	
	.	
	15	VME Controller 0xd800

Argument	Description
chan	SCSI channel 0 or 1
target	SCSI target device 0 through 7
lun	SCSI logical device 0 through 7
imagespec	ASCII string representing the image name (sysboot, sysex,...).

Note – Only channel 0 is supported when using VME SCSI boot. Make sure the boot device is on channel 0 under this configuration.

Examples

```
ROM >>rawboot 0 0 6 0 sysboot.110<CR>
Manual boot (sysboot.110) from 0 0 6 0
Header Partition Image Rev 3.3.0 for OS: Umax 4.2

  UMAX V Boot Loader 4.17: Fri Jun 5 11:56:58 EDT 1998

Sysboot->
```

When `rawboot` is issued with `netboot` enabled, this message appears:

```
ROM >>rawboot 1 0 6 0 sysboot.110<CR>
Manual netboot (sysboot.110) from 0x3fc00000
```

`reset` - Reset Node or SCSI Channels

Reset node or SCSI channels. The `reset node` command resets the system and is equivalent to powering on the system; the same information is displayed in either case. Refer to Chapter 1, *Starting and Using the ROM Monitor*, for an example of the console display resulting from reset or power up.

The `reset scsi` command issues a SCSIbus reset and reconfiguration based on devices found connected to the SCSIbus. Issue a `reset scsi` command if you install a SCSI device while the system is on-line or anytime after the power up configuration check.

Format

reset [**n**ode **|** **sc**si]

Arguments

Argument	Description
node	Nodal reset option
scsi	SCSI bus reset option

Examples

```
ROM >>reset scsi <CR>
Searching for Scsi Devices on Channel A:
................................
Device: 0 0 3 0, tape drive
Device: 0 0 5 0, disk drive
Device: 0 0 6 0, disk drive

ROM >>
```

`transparent` - Communicate Over Console Serial Port

Allow access to the serial line connected to the specified console port. Commands entered at the console are sent to the serial line; characters received from the serial line are displayed on the console.

A sample use for this command is to send commands to configure a modem connected to a serial port.

Leave transparent mode by entering **`~.`** (tilde period).

Format

`trans``parent` *port#*

Arguments

Argument	Description
port#	Serial port number. Valid range is 0 through 3.

Examples

```
ROM >>transparent 2<CR>
Transparent Mode Active
Enter ~. to terminate
ROM >> ~.
```

CHAPTER 3

Node Configuration Commands

The Processor Module board contains battery-backed nonvolatile random-access memory (NVRAM) containing permanent node configuration and diagnostic information. At node power up or reset, the ROM Monitor copies the contents of NVRAM into the Processor Module's dynamic RAM (DRAM). The DRAM temporarily stores the node configuration until the next power up or reset, when NVRAM again overwrites DRAM with node configuration information.

The node configuration commands that affect DRAM are: `setconfig`, `readconfig` and `show`. Use `setconfig` to change and temporarily store node configuration information in DRAM. Use `readconfig` to read and display the configuration information stored in DRAM. Use `show` to read and display the configuration information that was used to initialize the CPU.

Note – The `nvram` command, described in Chapter 2, lets you permanently store `setconfig` changes in NVRAM.

All three commands share many of the same arguments; in the case of `readconfig` and `show`, the arguments are read-only. Where arguments differ is noted.

This chapter describes each `setconfig` and `readconfig` command argument with format and examples listed in alphabetical order. The `show` command is listed separately at the beginning. See TABLE 3-1 for valid `setconfig` and `readconfig` command arguments.

The memconfig, slot and vmeconfig commands have their own sets of individual arguments and are listed separately at the end of this chapter.

TABLE 3-1 setconfig and readconfig Command Arguments

Argument	Description
autoboot	Set or read autoboot device
autotest	Set or read autotest state
bdevice	Set or read autoboot configuration device
bimage	Set or read autoboot image name
boardid	Read Processor Module board identification (ID)
btimeout	Set or read autoboot timeout value
clockcal	Set or read time-of-day clock calibration value
cpu	Set or read node CPU configuration
eadrs	Read node Ethernet address
errlog	Set or read error logging and reporting
errport	Set or read error port number
estate	Set or read environmental monitor state flag
memconfig	Set or read expansion memory configuration values (this command contains subarguments for base addresses, interleave factor, and other information)
msize	Read node's DRAM size
printerr	Set or read error port state
revision	Read Processor Module board and adaptor board revision levels (this command is also used to set the processor board revision level)
scsibus	Read SCSI bus device types and logical unit numbers
scsichanA/B	Set or read on-board SCSI channel
scsidelay	Set or read SCSI bus device initialization delay time
scsiid	Set or read SCSI bus channel ID
scsimaskA/B	Set or read SCSI masked out devices
scsiresetA/B	Set or read issuance of SCSI resets
scsitimeout	Set or read timeout value
serial	Set or read Processor Module board serial ports
slot	Set or read slot configuration values (this command contains subarguments for board type, board state, control and status registers, diagnostics, and other information)

TABLE 3-1 `setconfig` and `readconfig` Command Arguments

Argument	Description
`snumber`	Read Processor Module board serial number
`sysid`	Set or read system ID number
`tlevel`	Set or read autotest level
`version`	Read ROM Monitor firmware version from ROM
`vmeconfig`	Set or read VME values (this command contains subarguments for address modifier bits, master/slave page bits, and other information)

Configuration Change Warning

Values changed by the `setconfig` command apply to any subsequent reboot (caused by power up, reset, `boot` or `rawboot` command). However, if the ROM Monitor detects a difference between the NVRAM-stored configuration and the actual configuration, any autoboot operation aborts and the following message is displayed on the console:

```
WARNING: CPU / MEMORY / SCSI Configuration changed, Autoboot
aborted
Must accept new configuration to autoboot.
```

Use the `nvram accept` command to store the configuration. Chapter 2, *Node Control Commands*, describes the `nvram` command.

Also, only the ROM Monitor, extended diagnostics, and `sysboot` use configuration information stored in NVRAM. Operating system software may not use this information and could override the sysbooted configuration.

Monitor Response to Commands

Pressing `<CR>` after entering a `setconfig` command with arguments returns only the ROM Monitor prompt (`ROM >>`). Pressing `<CR>` after entering a `readconfig` command with arguments reads and displays the requested data.

Entering `readconfig` or `setconfig` without arguments returns a list of available configuration values.

Entering `show` without arguments returns a list of available arguments.

Command Descriptions

Chapter 1, *Starting and Using the ROM Monitor,* describes command line entry conventions. Typing `help` or `?` at the `ROM >>` prompt displays all currently available and valid commands. Typing `help` *commandname*, where *commandname* is a valid command, displays the command format.

`show` Command

The `show` command displays the current configuration and status information of the CPU devices. This information may differ from that displayed using the `readconfig` command, which displays the values that are stored in NVRAM.

Format

show [**v**me | **sl**ots | **c**onfig | **m**emconfig | **sc**si | **se**rial]

Arguments

Argument	Description
vme	Display current VMEbus configuration
slots	Display current slot configuration
config	Display current system configuration
memconfig	Display current memory configuration
scsi	Display current SCSI configuration
serial	Display current serial port configuration and status

Examples

```
ROM >>show config<CR>

autoboot state:             disabled
netboot state:              disabled
altnetboot:                 disabled
boot image:                 sysboot.110
altboot image:              sysboot.110
ip boot address:            none
ip altboot address:         none
default autoboot device:    bus 0 channel 0, target 6, lun 0, part 0
alternate autoboot device:  bus 0 channel 0, target 6, lun 0, part 0
autoboot timeout:           7 sec.
autotest:                   disabled
autotest levels selected:   core, qmem, slots,
cpus' configured:           0, 1, 2, 3
processor brd serial num:   93s12345
adaptor brd serial num:     93s19684
assembly id:                0x504
processor brd func rev:     A
adaptor brd func rev:       C
processor brd mem size:     128Mb
adaptor brd mem size:       64Mb
More ? (Y/N)y
ethernet address:           08:00:4c:00:ff:97
scsi delay:                 0 sec.
scsi timeout:               30 sec.
bus addr 0 channel 0 scsi id: 0
bus addr 0 channel 1 scsi id: 0
clock calibration:          +00
environmental monitor:      disabled
error state:                disabled
burnin state:               disabled
error port:                 3
board set id:               7        (88110 Processor Brd. Set)
Common Console Interface:   not present
system id:                  0
Part: 531_103422_006, RCS: V1_06, Created: Tues Oct 3 22:03:50 EDT
1995
```

```
ROM >>show vme<CR>

arbiter scheduling:           priority
requester protocol:           release when done (not fair)
service interrupt level:      3
requester interrupt level:    3
address modifier bits:        0xd         (supervisory data access)
master page bits:             2           (0x40000000 - 0x5FFFFFFF)
slave page bits:              0           (0x00000000)
local page bits:              0           (0x00000000 - 0x3FFFFFFF)
window page bits:             00          (0x00000000)
standard space:               enabled
extended space:               enabled
extended window:              disabled
owner timeout:                15
i/o global snooping:          disabled
vme global snooping:          disabled


ROM >>show slots<CR>

Slot      type              state                     eimage
----------------------------------------------------------------

1         adaptor           configured                none
2         processor         configured                none
3         edram             configured                none
4         undefined         undefined                 none
5         undefined         undefined                 none
6         undefined         undefined                 none
7         undefined         undefined                 none
8         undefined         undefined                 none
9         undefined         undefined                 none

More ? (Y/N)


ROM >>show mem<CR>

slot      lbus addr         vbus addr    size  ileave  pos  isize
-------------------------------------------------------------------
2         0x3e000000        none         32Mb  none    -    -----
3         0x40000000        0x40000000   512Mb none    -    -----
```

```
ROM >>show scsi<CR>

Device      type            Product Id     Vendor     Rev
----------------------------------------------------------------
0 0 3 0     tape drive      TDC 3800       TANDBERG   =02:
0 0 4 0     disk drive      94181-15       CDC        0293
0 0 5 0     disk drive      94171-9        CDC        7975
0 0 6 0     disk drive      94181-15       CDC        0293

ROM >>

ROM >>show serial<CR>

Port     speed    stop     parity     state
-------------------------------------------
0        9600     1        none       open
1        9600     1        none       open
2        9600     1        none       closed
3        9600     1        none       closed

ROM >>
```

`setconfig`/`readconfig` Commands

This section describes in detail all the parameters for the `setconfig` and `readconfig` commands.

`altbdevice` - Alternate Autoboot Specified Device

Set or read the alternate autoboot SCSI device storing the autoboot image software.

Format

`set`config **`altbd`**evice *bus chan target lun* [*partindex*]
`readc`onfig **`altbd`**evice

Arguments

Argument	Description	
bus	Index or hex value of a valid VME short address	
	Index #	**Description**
	0	On-board SCSI
	1	VME Controller 0x6800
	2	VME Controller 0x7000
	3	VME Controller 0x7800
	.	
	.	
	.	
	15	VME Controller 0xd800
chan	SCSI channel 0 or 1	

Argument	Description
target	SCSI target device 0 through 7
lun	SCSI logical device 0 through 7
partindex	Boot disk partition index

Note – Only channel 0 is supported when using VME SCSI boot. Make sure the boot device is on channel 0 under this configuration.

Examples

```
ROM >>setconfig altbdevice 0 0 6 0 0<CR>
ROM >>
ROM >>readconfig altbdevice<CR>
autoboot device: bus <on-board SCSI> channel 0 target 6 lun 0
partindex 0
ROM >>
```

`altbimage` - Alternate Autoboot Image Name

Set or read the alternate autoboot image name.

Format

`set`config **`altb`**image *string*
`readc`onfig **`altb`**image

Arguments

Argument	Description
string	Alternate autoboot image name (such as sysboot, umax.image, rmsinit,...)

Examples

```
ROM >>readconfig altbimage<CR>
altboot image: rmsinit
ROM >>
ROM >>setconfig altbimage sysboot<CR>
ROM >>
ROM >>readconfig altbimage<CR>
altboot image: sysboot
ROM >>
```

`altnetboot` - Alternate Netboot State

Setting `altnetboot` when autobooting allows retrieval from a secondary boot server located at IP address *altipaddr* when the primary boot device fails. *altipaddr* is the image defined by `altbimage` for a network load into the address defined by *altloadaddr*. Typically the `altbimage` is `sysboot`, which is usually directed to a secondary drive defined by `altbdevice`. If `altnetboot` is `enabled` the image is `sysboot.image`.

Format

`set``config` **`altnetb`**`oot [`**`d`**`isable] [`**`e`**`nable` *altipaddr altloadaddr*`]`
`readc``onfig` **`altnetb`**`oot`

Arguments

Argument	Description
`d``isable`	Disable the alternate netboot feature
`e``nable`	Enable the alternate netboot feature
altipaddr	Internet address of the alternate boot server in dot notation (a.b.c.d)
altloadaddr	Memory address of where the alternate image is to be loaded

Examples

The following example is for a system configured with 64 Megabytes.

To enable this feature:

```
ROM >>setconfig altnetboot enable 129.91.120.104 0x3fc00000<CR>
ROM >>setconfig altbimage sysboot.image<CR>
ROM >>setconfig altbdevice 1 0 6 0<CR>
ROM >>nvram accept
Are you sure you want to save the configuration (y/n)?y
ROM >>reset node
```

To disable this feature:

```
ROM >>setconfig altnetboot disable<CR>
ROM >>nvram accept<CR>
are you sure you want to save the configuration (y/n)?y
ROM >>
```

`autoboot` - Autoboot State

Set or read the autoboot state. Sun StorEdge A7000 systems have autoboot enabled as a default condition.

Format

set`config` **autob**`oot (`**e**`nable` **|** **d**`isable)`
readc`onfig` **autob**`oot`

Arguments

Argument	Description
e`nable`	Executes an autoboot sequence if no errors are detected during the power up sequence
d`isable`	Does not execute an autoboot sequence after power up

Examples

```
ROM >>readconfig autoboot<CR>
autoboot: disabled
ROM >>
ROM >>setconfig autoboot enable<CR>
ROM >>
ROM >>readconfig autoboot<CR>
autoboot: enabled
ROM >>
```

`autotest` - Autotest State

Set or read the power up or reset autotest state. Typically, system default power up or reset is with tests enabled. Pressing any key at this message during power up or reset disables the tests:

```
Automatic test enabled, 3 seconds to break . . .
```

Format

setconfig **autot**est (**e**nable **|** **d**isable)
readconfig **autot**est

Arguments

Argument	Description
enable	Executes built-in self tests and extended diagnostics on power up or reset
disable	Skips built-in self tests and extended diagnostics on power up or reset

Examples

```
ROM >>setconfig autotest enable<CR>
ROM >>
ROM >>readconfig autotest<CR>
autotest: enabled
ROM >>
```

`bdevice` - Autoboot Specified Device

Set or read the autoboot SCSI device storing the auto boot image software.

Format

set`config` **bd**`evice` *bus chan target lun* [*partindex*]
readc`onfig` **bd**`evice`

Arguments

Argument	Description	
bus	Index or hex value of a valid VME short address	
	Index #	**Description**
	0	On-board SCSI
	1	VME Controller 0x6800
	2	VME Controller 0x7000
	3	VME Controller 0x7800
	.	
	.	
	.	
	15	VME Controller 0xd800
chan	SCSI channel 0 or 1	
target	SCSI target device 0 through 7	
lun	SCSI logical device 0 through 7	
partindex	Boot disk partition index	

Note – Only channel 0 is supported when using VME SCSI boot. Make sure the boot device is on channel 0 under this configuration.

Examples

```
ROM >>setconfig bdevice 0 0 6 0 0<CR>
ROM >>
ROM >>readconfig bdevice<CR>
autoboot device: bus <on-board SCSI> channel 0 target 6 lun 0
partindex 0
ROM >>
```

`bimage` - Autoboot Image Name

Set or read the autoboot image name

Format

set`config` **bi**`mage` *string*
readc`onfig` **bi**`mage`

Arguments

Argument	Description
string	Autoboot image name (such as sysboot, umax.image, rmsinit,...)

Examples

```
ROM >>readconfig bimage<CR>
boot image: sysboot
ROM >>
ROM >>setconfig bimage umax.image<CR>
ROM >>
ROM >>readconfig bimage<CR>
boot image: umax.image
ROM >>
```

`boardid` - Board ID

Read the major and minor processor board ID levels.

Format

readc`onfig` **bo**`ardid`

Examples

```
ROM >>readconfig boardid<CR>
Board Set ID = 7, (88110 Processor Brd. Set)
ROM >>
```

`btimeout` - Autoboot Timeout Value

Set or read the autoboot timeout value. This value is displayed during the power up sequence if autoboot is enabled. It sets the time the operator has to press a key to bypass the autoboot sequence, as in the following message:

```
You have 5 seconds to terminate the autoboot sequence ...
```

Format

set`config` **bt**`imeout` *value*
readc`onfig` **bt**`imeout`

Arguments

Argument	Description
value	Timeout in seconds

Examples

```
ROM >>setconfig btimeout 7<CR>
ROM >>
ROM >>readconfig btimeout<CR>
autoboot timeout: 7 sec.
ROM >>
```

`clockcal` - Clock Calibration Value

Set or read the time-of-day (TOD) clock calibration value. Use this command to correct the TOD clock if it runs fast or slow.

Format

set`config` **cl**`ockcal` *value*
readc`onfig` **cl**`ockcal`

Arguments

Argument	Description
value	+/- 0 through 31 (+ makes clock run faster, - makes clock run slower)

Note – If *value* is not preceded by an operator, *value* is assumed to be +.

Examples

```
ROM >>setconfig clockcal +10<CR>
ROM >>
ROM >>readconfig clockcal<CR>
clock calibration: +10
ROM >>
```

`cpu` - CPU Configuration

Set or read the node CPU configuration. CPUs 1 through 7 are configured and deconfigured by an odd number in the command line. CPU 0 is always configured.

Format

```
setconfig cpu value
readconfig cpu
```

Arguments

Argument	Description							
value	*x x x x x x x* `1` = 0x01 - 0xff (odd numbers only, can also be decimal)							
	x	*x*	*x*	*x*	*x*	*x*	*x*	`1`
	CPU 7	CPU 6	CPU 5	CPU 4	CPU 3	CPU 2	CPU 1	CPU 0
								(always configured)
	x = 1	configure						
	x = 0	deconfigure						

Note – If an even value is specified, the monitor returns a warning message. However, the command executes normally but CPU 0 remains configured.

Examples

```
ROM >>setconfig cpu 0xff<CR>
ROM >>
```

Here is an example of using an even number:

```
ROM >>setconfig cpu 12<CR>
WARNING: cpu 0 cannot be deconfigured
ROM >>
ROM >>readconfig cpu<CR>
Node 1:
cpu 0: configured
cpu 1: deconfigured
cpu 2: configured
cpu 3: configured
cpu 4: deconfigured
cpu 5: deconfigured
cpu 6: deconfigured
cpu 7: deconfigured
ROM >>
```

`eadrs` - Node Ethernet Address

Read the node Ethernet address.

Format

`readc``onfig` **`ea`**`drs`

Examples

```
ROM >>readconfig eadrs<CR>
Ethernet Address: 08-00-4c-00-2c-91
ROM >>
```

`errlog` - Enable or Disable Errorlog Reporting

Enable or disable errorlog reporting. The error log reporting must be turned off for all of DRAM to be tested. The recommendation is to have error log reporting disabled, and then perform an `nvram accept` and manually perform a `test lmem` before booting the operating system. For more information on error log reporting see the `errorlog` command in chapter 2.

Format

```
setconfig errlog  [enable | disable]
readconfig errlog
```

Arguments

Argument	Description
`enable`	Enable error logging and reporting
`disable`	Disable error logging and reporting

Examples

```
ROM >>setconfig errlog disable<CR>
ROM >>
ROM >>setconfig errlog enable<CR>
ROM >>
ROM >>readconfig errlog<CR>
errlog enabled
ROM >>
```

`errport` - Error Port

Set or read the error port number. Ports 1-3 may be used as an error port. The error port is enabled and disabled by the `setconfig printerr` command. If the error port is enabled all error messages are sent to both the console port and the error port. Port 0 is the console port.

Format

`set``config` **`er`**`rport` *port#*
`readc``onfig` **`er`**`rport`

Arguments

Argument	Description
port #	Valid ports are 1-3

Examples

```
ROM >>setconfig printerr enable<CR>
ROM >>readconfig printerr<CR>
error port: enabled
ROM >>
ROM >>setconfig errport 3<CR>
ROM >>
ROM >>readconfig errport<CR>
error port: 3
ROM >>
```

`estate` - Environmental Monitor State

Set or read environmental monitor state flag.

Format

```
setconfig estate (enable | disable)
readconfig estate
```

Arguments

Argument	Description
enable	Allow reading the environmental sensors
disable	Disallow reading the environmental sensors

Examples

```
ROM >>setconfig estate disable<CR>
ROM >>
ROM >>readconfig estate<CR>
environmental monitor: disabled
ROM >>
```

`msize` - Local Dynamic RAM Memory Size

Read the node local DRAM size from the ID PROM.

Format

readc`onfig` **msi**`ze`

Examples

```
ROM >>readconfig msize<CR>
Processor Brd Memory Size = 128Mb
Adaptor Brd Memory Size = 64Mb
ROM >>
ROM >>readconfig msize<CR>
Processor Brd Memory Size = 32Mb
Adaptor Brd Memory Size = 0Mb
ROM >>
```

`netboot` - Network Autoboot State

Setting `netboot` when autobooting retrieves from a boot server located at IP address *ipaddr*, the image defined by `bimage` for a network load into the load address defined by *loadaddr*. Typically the `bimage` is `sysboot.image`, which is usually directed to a primary drive defined by `bdevice`.

Note – This interface relies on the `tftp` service. This means that the `sysboot.image` file must be available on the disk of the boot server in the `/tftpboot` directory.

Note – The Ethernet address of the CPU board must be manually loaded into the boot server's `arp` table. (See `readconfig eadrs` and man pages for `arp`).

Format

`set``config` **`netb`**`oot [`**`d`**`isable] [`**`e`**`nable` *ipaddr loadaddr*`]`
`readc``onfig` **`netb`**`oot`

Arguments

Argument	Description
`d``isable`	Disable the **`netboot`** feature
`e``nable`	Enable the **`netboot`** feature
ipaddr	Internet address of the boot server in dot notation (*a.b.c.d*)
loadaddr	Memory address of where the image is to be loaded

Examples

The following example is for a VME disk based system configured with 64 Megabytes.

To enable this feature:

```
ROM >>setconfig netboot enable 129.91.120.104 0x3fc00000<CR>
ROM >>setconfig bimage sysboot.image<CR>
ROM >>setconfig bdevice 1 0 6 0<CR>
ROM >>nvram accept<CR>
are you sure you want to save the configuration (y/n)?y
ROM >>reset node<CR>
ROM >>
```

To disable this feature:

```
ROM >>setconfig netboot disable<CR>
ROM >>nvram accept<CR>
Are you sure you want to save the configuration (y/n)y
ROM >>
```

`printerr` - Error Port State

Set or read the error port state. When the error port is enabled, all error messages are sent to both the console port and the error port.

Format

```
setconfig printerr (enable | disable)
readconfig printerr
```

Arguments

Argument	Description
`enable`	Allow error message output to the error port
`disable`	Disallow error message output to the error port

Examples

```
ROM >>readconfig printerr<CR>
error port: enabled
ROM >>
ROM >>setconfig printerr disable>CR>
ROM >>
ROM >>readconfig printerr<CR>
error port: disabled
ROM >>
```

`revision` - Board Revision

Read the assembly ID and functional revision level of the processor board, the adaptor board, and any configured memory boards. This command can also set the functional revision level of the processor module.

The following information is displayed:

- Assembly ID, displayed numerically
- Assembly Functional Revision, displayed as an alpha character (A - Z)

Format

readc`onfig` **re**`vision`

set`config` `r`**e**`vision`

Examples

```
ROM >>readconfig revision<CR>
Processor Board Assembly ID 0x504, Functional Revision A
Adaptor Board Assembly ID 0x504, Functional Revision C
ROM >>
ROM >>setconfig rev B<CR>
the functional revision of the processor board has been set to B
ROM >>
```

`scsi`*xxx* - SCSI Configuration Commands

SCSI configuration commands are used to configure, deconfigure, and display the configuration of the SCSI subsystem. These commands are listed in TABLE 3-2.

TABLE 3-2 SCSI Configuration Commands

Command	Description
`setconfig scsiid`	Set the SCSI bus ID for the specified SCSI bus and channel
`setconfig scsidelay`	Set a delay value in seconds, a time during which the ROM Monitor waits before sizing and initializing the SCSI devices
`setconfig scsichan`*A/B*	Enable or disable the entire on-board SCSI channel. If no devices are present you can expedite initialization by disabling an unused bus.
`setconfig scsireset`*A/B*	Control the issuance of SCSI resets to the on-board SCSI bus. This option should be enabled. The default is disabled.
`setconfig scsimask`*A/B*	Used to mask out devices to be ignored and is primarily used for internal purposes
`setconfig scsitimeout`	Watchdog timeout safeguard in case a SCSI device locks up the SCSI initialization sequence at boot. The default is 30 seconds.
`readconfig scsiid`	Read the current SCSI bus ID for both channels
`readconfic scsibus`	Read a table built by the ROM Monitor software at power up or system reset and display all SCSI bus device types and logical unit numbers for both channels. The ROM Monitor builds the table by examining the devices actually attached to the SCSI channels; unconfigured positions are reported as null devices. It also compares the values found with those saved in NVRAM. Any differences are displayed on the console, indicating that a configuration change has been made.
`readconfig scsidelay`	Read the current SCSI delay value
`scsistart`	Used to manually initialize the SCSI subsystem

Note – Executing a `reset scsi` command causes a SCSIbus reset and device reconfiguration.

Command Formats

The following formats are used by the SCSI configuration commands:

```
setconfig scsiid bus channel id
setconfig scsidelay value
setconfig scsichanA [enable | disable]
setconfig scsichanB [enable | disable]
setconfig scsiresetA [enable | disable]
setconfig scsiresetB [enable | disable]
setconfig scsimaskA mask
setconfig scsimaskB mask
setconfig scsitimeout value
readconfig scsiid
readconfig scsibus
readconfig scsidelay
readconfig scsichanA
readconfig scsichanB
readconfig scsiresetA
readconfig scsiresetB
readconfig scsimaskA
readconfig scsimaskB
readconfig scsitimeout
scsistart
```

Command Arguments

Argument	Description	
bus	Index or hex value of a valid VME short address	
	Index #	**Description**
	0	On-board SCSI
	1	VME Controller 0x6800
	2	VME Controller 0x7000
	3	VME Controller 0x7800
	.	
	.	
	.	
	15	VME Controller 0xd800
chan	SCSI channel 0 or 1	
id	SCSI bus ID to be set for the specified channel. Valid range is 0 through 7.	
value	Delay value in seconds	
mask	OR'ing the values that represent the SCSI IDs	
e`nable`	Enable SCSI channel A/B or SCSI resets A/B	
disable	Disable SCSI channel A/B or SCSI resets A/B	

Note – Only channel 0 is supported when using VME SCSI boot. Make sure the boot device is on channel 0 under this configuration.

SCSI Startup Procedure

The first time a system comes up, the SCSI subsystem is uninitialized. The following message appears, indicating that the following steps are required for proper SCSI operation.

```
SCSI NOT CONFIGURED: refer to scsi startup procedure
```

1. Determine whether SCSI channel A, B, or both are in operation. SCSI channel A is the single ended channel and channel B is the differential. To enable SCSI channel A or B, type one of the following command lines:

   a. For devices on channel A: **`setc scsichanA enable`**

   b. For devices on channel B: **`setc scsichanB enable`**

2. To enable SCSI resets, type one of the following command lines:

   a. For devices on channel A: **`setc scsiresetA enable`**

   b. For devices on channel B: **`setc scsiresetB enable`**

3. To "mask" SCSI IDs so they are skipped at initialization time use the command: **`setc scsimaskA`** *mask* where *mask* is defined by OR'ing the values that represent the SCSI IDs presented in TABLE 3-3.

TABLE 3-3 SCSI Mask Values

SCSI ID	Mask Value
SCSI ID 0	0x01
SCSI ID 1	0x02
SCSI ID 2	0x04
SCSI ID 3	0x08
SCSI ID 4	0x10
SCSI ID 5	0x20
SCSI ID 6	0x40
SCSI ID 7	0x80

For example, for the devices on channel B use: **`setc scsimaskB`** *mask*

4. After execution of the commands in steps 1 through 3, type: **`nvram accept`**

5. Type: **`scsistart`**

Note – Subsequent reboots automatically initialize the SCSI subsystem

The following example initializes the SCSI devices on channel A:

```
ROM >>setconfig scsichanA enable<CR>
ROM >>setconfig scsiresetA enable<CR>
ROM >>nvram accept<CR>
Are you sure you want to save the configuration (y/n)?y
ROM >>scsistart<CR>
```

SCSI Configuration Command Execution Examples

```
ROM >>setconfig scsiid 0 0 0<CR>
ROM >>
ROM >>setconfig scsiid 0 1 0<CR>
ROM >>
ROM >>setconfig scsidelay 2<CR>
ROM >>
ROM >>readconfig scsidelay<CR>
scsi delay: 2 sec.
ROM >>
ROM >>readconfig scsiid<CR>
Node 1:
bus address: 0x0 / channel: 0 / id: 0
bus address: 0x0 / channel: 1 / id: 0
bus address: 0x8800 / channel: 0 / id: 0
bus address: 0x8800 / channel: 1 / id: 0
.
.
.
bus address: 0xa800 / channel: 0 / id: 0
bus address: 0xa800 / channel: 1 / id: 0
bus address: 0xb000 / channel: 0 / id: 0
bus address: 0xb000 / channel: 1 / id: 0
bus address: 0xb800 / channel: 0 / id: 0
bus address: 0xb800 / channel: 1 / id: 0
bus address: 0xc000 / channel: 0 / id: 0
bus address: 0xc000 / channel: 1 / id: 0
More ? (Y/N)y
bus address: 0xc800 / channel: 0 / id: 0
.
.
.
bus address: 0xf000 / channel: 0 / id: 0
bus address: 0xf000 / channel: 1 / id: 0
bus address: 0xf800 / channel: 0 / id: 0
bus address: 0xf800 / channel: 1 / id: 0
ROM >>
ROM >>readconfig scsibus<CR>
Node 1:
bus: 0 / channel: 0 / target: 4 / lun: 0 / device: tape drive
bus: 0 / channel: 0 / target: 5 / lun: 0 / device: disk drive
bus: 0 / channel: 0 / target: 6 / lun: 0 / device: disk drive
ROM >>
```

`serial` - Serial Configuration

Set or read the serial port configuration. The serial ports are set individually by separate commands. All ports are read with the `readconfig serial` command.

Note – If the console port values do not match the device configuration, console control is lost. This is true with other devices. The console must be attached to port 0.

Format

set`config` **se**`rial` *port# rate stop parity*
readc`onfig` **se**`rial`

Arguments

Argument	Description
port#	Valid ports numbered 0 through 3
rate	Port baud rate. Valid rates are 300, 1200, 2400, 3600, 4800, 7200, 9600, 19200, and 38400
stop	Port stop bits. Valid number of bits is 1, 1.5, or 2
parity	Odd, even, or no port parity

Examples

```
ROM >>setconfig serial 1 4800 2 even<CR>
ROM >>
ROM >>readconfig serial<CR>
port 0 baud: 9600, stop: 1, parity: none
port 1 baud: 9600, stop: 1, parity: none
port 2 baud: 9600, stop: 1, parity: none
port 3 baud: 9600, stop: 1, parity: none
ROM >>
```

`snumber` - Board Serial Number

Set or read the processor board serial number.

Format

readconfig **sn**umber
setconfig **sn**umber

Examples

```
ROM >>setconfig snumber 93s12345<CR>
Processor Brd Serial Number set to: 93s12345
ROM >>
ROM >>readconfig snumber<CR>
Processor Board Serial Number = 93s12345
Adaptor Board Serial Number = 93s19684
ROM >>
```

`sysid` - System ID

Set or read the system ID. The system ID number is unique to each processor board.

⚠ **Caution –** This command is to be used only by a Customer Services engineer when replacing a processor board. The system ID number must not be changed for other reasons.

Format

set`config` **sy**`sid` *id*
readc`onfig` **sy**`sid`

Arguments

Argument	Description
id	System ID of up to 12 numbers

Examples

```
ROM >>setconfig sysid 123456<CR>
ROM >>
ROM >>readconfig sysid<CR>
system id: 123456
ROM >>
```

`tlevel` - Autotest Level

Set or read the autotest level. The `setconfig` command adds or removes specified test level(s) from the autotest sequence. Levels not specified in the command remain unmodified.

Format

```
setconfig tlevel (+ | - [core | cache | ecc | lmem | lpbk | qmem
| scsi | slots])
```

```
readconfig tlevel
```

Arguments

Argument	Description
+ | **-**	Indicates that the specified options are to be added to or removed from autotest level
core	Processor and adaptor board tests
cache	Cache memory tests
ecc	ECC memory tests
lmem	Various algorithm tests of onboard and expansion memory
lpbk	External loopback tests (external hardware required)
qmem	Address line, MBUS fairness, and VBUS fairness tests of onboard and expansion memory
scsi	Tests the SCSI interface along with reads and writes to disk
slots	Any configured extended diagnostic tests

Examples

```
ROM >>readconfig tlevel<CR>
autotest levels selected: qmem lmem slots
ROM >>setconfig tlevel +core -lmem<CR>
ROM >>
ROM >>readconfig tlevel<CR>
autotest levels selected: core qmem slots
ROM >>
```

`version` - Present Monitor Version

Read the ROM Monitor version and creation date.

Note – The version listed in the example may not match your version.

Format

readc`onfig` **ve**`rsion`

Examples

```
ROM >>readconfig version<CR>
88110 ROM Monitor
Part: 531_103422_066, RCS: V1_06, Created: Tue Oct 3 22:03:50 EDT
1995
ROM >>
```

memconfig - Expansion Memory Configuration Commands

Set or read the expansion memory configuration values. Variations of the setconfig memconfig command set the values individually. Variations of the readconfig memconfig command allow the operator to read the values individually or all at once.

Memory configuration software serially configures all memory boards present in the system.

Note – The commands relating to address interleaving are valid only in systems containing two or four expansion memory boards.

Warning messages are displayed on the console for the following reasons:

- Illegal holes in memory
- Memory board found in a slot that is configured for a CPU or VME board
- A command directed at a slot not containing a memory board

The variables for the memconfig command are listed alphabetically in the following pages.

`all` - All Expansion Memory Configuration

Read all expansion memory configuration values for the specified board.

Format

readc`onfig` **mem**`config` *brd#* **a**`ll`

Arguments

Argument	Description
brd#	Expansion memory board number; valid range is 1 through 4

Examples

```
ROM >>readconfig memconfig 1 all<CR>
Board 1: local bus base address: 0x400000
Board 1: vme bus base address: 0x800000
Board 1: Interleave Factor = 4-board interleave
Board 1: Interleave Size = 16 bytes
Board 1: Local Bus Interface: enabled
Board 1: VME Bus Interface: enabled
Board 1: Interleave Position: 2
Board 1: local control parity error interrupts: enabled
Board 1: local address parity error interrupts: disabled
Board 1: vme parity interrupts: disabled
Board 1: vme berr on parity error: disabled
ROM >>
```

`berr` - VME BERR

Set or read the state of the VME BERR signal on a VME read parity error.

Format

set`config` **mem**`config` *brd#* **be**`rr (`**e**`nable | `**d**`isable)`
readc`onfig` **mem**`config` *brd#* **be**`rr`

Arguments

Argument	Description
brd#	Expansion memory board number; valid range is 1 through 4
e`nable`	Generate a BERR (board error) if a VME read parity error occurs
d`isable`	Do not generate BERR on VME read parity error

Examples

```
ROM >>readconfig memconfig 1 berr<CR>
Board 1: vme berr on parity error: enabled
ROM >>
ROM >>setconfig memconfig 1 berr disable<CR>
ROM >>
ROM >>readconfig memconfig 1 berr<CR>
Board 1: vme berr on parity error: disabled
ROM >>
```

`ifactor` - Interleave Factor

Set or read the board interleave factor (systems with two or four expansion memory boards).

Format

set`config` **mem**`config` *brd#* **if**`actor` `(`**1** `|` **2** `|` **4**`)`
readc`onfig` **mem**`config` *brd#* **if**`actor`

Arguments

Argument	Description
brd#	Expansion memory board number; valid range is 1 through 4
1	Specifies no interleave (use for systems containing 0, 1, or 3 boards)
2	Specifies 2-board interleave
4	Specifies 4-board interleave

Examples

```
ROM >>setconfig memconfig 1 ifactor 4<CR>
ROM >>
ROM >>readconfig memconfig 1 ifactor<CR>
Board 1: Interleave Factor = 4-board interleave
ROM >>
ROM >>setconfig memconfig 2 ifactor 1<CR>
ROM >>
ROM >>readconfig memconfig 2 ifactor<CR>
Board 2: Interleave Factor = no interleave
ROM >>
```

`ipos` - Interleave Position

Set or read the board interleave positions. Each board in the interleave must have a different interleave position. All boards are set with the same command so it does not matter which board number is specified, but the number of digits specified in the position values must be equal to the number of boards installed or an error message will be displayed. Each board is read individually.

Format

set`config` **mem**`config` *brd#* **ip**`os` *positionvalues*
readc`onfig` **mem**`config` *brd#* **ip**`os`

Arguments

Argument	Description
brd#	Expansion memory board number; valid range is 1 through 4
positionvalues	One digit for each board in the interleave; valid range is 1 through 4

Examples

The following example sets boards 1 and 2 to interleave positions 1 and 2, respectively.

```
ROM >>setconfig memconfig 1 ipos 12<CR>
ROM >>

ROM >>readconfig memconfig 1 ipos<CR>
Board 1: Interleave Position: 1
ROM >>
ROM >>readconfig memconfig 2 ipos<CR>
Board 2: Interleave Position: 2
ROM >>
```

`isize` - Interleave Size

Set or read the board interleave size for systems with two or four expansion memory boards. All expansion memory boards in the system are set at once to the same value; it does not matter which board is specified in the `setconfig` command.

Format

```
setconfig memconfig brd# isize (16  | 32 | 64 | 128 | 256)
readconfig memconfig brd# isize
```

Arguments

Argument	Description
brd#	Expansion memory board number; valid range is 1 through 4
16|32|64|128|256	Memory segment size (in bytes) for interleaving (all boards must be set to the same size)

Examples

```
ROM >>setconfig memconfig 1 isize 16<CR>
ROM >>
ROM >>readconfig memconfig 1 isize<CR>
Board 1: Interleave Size = 16 bytes
ROM >>
```

`laparity` - Local Address Parity

Set or read the state of the local address parity error interrupt.

Format

set`config` **mem**`config` *brd#* **la**`parity (`**e**`nable | `**d**`isable)`
readc`onfig` **mem**`config` *brd#* **la**`parity`

Arguments

Argument	Description
brd#	Expansion memory board number; valid range is 1 through 4
e`nable`	Allow response to local address parity error interrupts
d`isable`	Ignore local address parity error interrupts

Examples

```
ROM >>readconfig memconfig 1 laparity<CR>
Board 1: local address parity error interrupts: enabled
ROM >>
ROM >>setconfig memconfig 1 laparity disable<CR>
ROM >>
ROM >>readconfig memconfig 1 laparity<CR>
Board 1: local address parity error interrupts: disabled
ROM >>
```

`lbase` - Local Bus Base Address

Set or read the local bus base address.

Format

setconfig **mem**config *brd#* **lba**se *addr*
readconfig **mem**config *brd#* **lba**se

Arguments

Argument	Description
brd#	Expansion memory board number; valid range is 1 through 4
addr	Local bus base address; valid range is 0x40000000 to 0x7f000000

Examples

```
ROM >>setconfig memconfig 1 lbase 0x40000000<CR>
ROM >>
ROM >>readconfig memconfig 1 lbase<CR>
Board 1: local bus base address: 0x40000000
ROM >>
```

`lbus` - Local Bus Interface

Set or read the state of the local bus interface.

Format

setconfig **mem**config *brd#* **lbu**s (**e**nable | **d**isable)
readconfig **mem**config *brd#* **lbu**s

Arguments

Argument	Description
brd#	Expansion memory board number; valid range is 1 through 4
enable	Enable local bus interface
disable	Disable local bus interface

Examples

```
ROM >>setconfig memconfig 1 lbus enable<CR>
ROM >>
ROM >>readconfig memconfig 1 lbus<CR>
Board 1: Local Bus Interface: enabled
ROM >>
ROM >>setconfig memconfig 4 lbus disable<CR>
ROM >>
ROM >>readconfig memconfig 4 lbus<CR>
Board 4: Local Bus Interface: disabled
ROM >>
```

`lcparity` - Local Control Parity

Set or read the state of the local control parity error interrupt.

Format

setconfig **mem**config *brd#* **lc**parity (**e**nable **|** **d**isable)
readconfig **mem**config *brd#* **lc**parity

Arguments

Argument	Description
brd#	Expansion memory board number; valid range is 1 through 4
enable	Allow response to local control parity error interrupts
disable	Ignore local control parity error interrupts

Examples

```
ROM >>readconfig memconfig 1 lcparity<CR>
Board 1: local control parity error interrupts: disabled
ROM >>
ROM >>setconfig memconfig 1 lcparity enable<CR>
ROM >>
ROM >>readconfig memconfig 1 lcparity<CR>
Board 1: local control parity error interrupts: enabled
ROM >>
```

vbase - VMEbus Base Address

Set or read the VMEbus base address.

Format

setconfig **mem**config *brd#* **vba**se *addr*
readconfig **mem**config *brd#* **vba**se

Arguments

Argument	Description
brd#	Expansion memory board number; valid range is 1 through 4
addr	VMEbus base address; valid range is 0x80000000 to 0xbf000000

Examples

```
ROM >>setconfig memconfig 1 vbase 0x80000000<CR>
ROM >>
ROM >>readconfig memconfig 1 vbase<CR>
Board 1: vme bus base address: 0x80000000
ROM >>
```

`vbus` - VMEbus Interface

Set or read the state of the expansion memory board VMEbus interface.

Format

```
setconfig memconfig brd# vbus (enable | disable)
readconfig memconfig brd# vbus
```

Arguments

Argument	Description
brd#	Expansion memory board number; valid range is 1 through 4
`enable`	Enable the VMEbus interface
`disable`	Disable the VMEbus interface

Examples

```
ROM >>setconfig memconfig 4 vbus disable<CR>
ROM >>
ROM >>readconfig memconfig 4 vbus<CR>
Board 4: VME Bus Interface: disabled
ROM >>
ROM >>setconfig memconfig 1 vbus enable<CR>
ROM >>
```

`vmeparity` - VME Parity

Set or read the state of the VME parity interrupt.

Format

set`config` **mem**`config` *brd#* **vm**`eparity (`**e**`nable` **|** **d**`isable)`
readc`onfig` **mem**`config` *brd#* **vm**`eparity`

Arguments

Argument	Description
brd#	Expansion memory board number; valid range is 1 through 4
e`nable`	Allow response to VME parity error interrupts
d`isable`	Ignore VME parity error interrupts

Examples

```
ROM >>readconfig memconfig 1 vmeparity<CR>
Board 1: vme parity interrupts: enabled
ROM >>
ROM >>setconfig memconfig 1 vmeparity disable<CR>
ROM >>
ROM >>readconfig memconfig 1 vmeparity<CR>
Board 1: vme parity interrupts: disabled
ROM >>
```

`slot` - Slot Configuration Commands

Set or read slot configuration values. Two commands set or read these values. Variations of the `setconfig slot` command set the slot configuration values individually. Variations of the `readconfig slot` command allow you to read the values individually or all at once.

The variables for the `slot` command are listed alphabetically in the following pages.

`all` - All Slot Configuration Values

Read all slot configuration values.

Format

`readc``onfig` **`sl`**`ot` *slot#* **`a`**`ll`

Arguments

Argument	**Description**
slot#	Chassis slot number

Examples

```
ROM >>readconfig slot 3 all<CR>
Slot 3: board type: vme
Slot 3: board state: Configured
Slot 3: board data: no bist
Slot 3: csr 1 address: 0xfff40800
Slot 3: csr 2 address: 0xfffc0000
Slot 3: csr 3 address: 0
Slot 3: extended image: rmsdiag
ROM >>
```

`bdata` - Board-Specific Data

Set or read the board-specific data flag. This flag tells whether BIST data is available for the slot.

Format

set`config` **slot***#* **b**`data` *value*
readc`onfig` **sl**`ot` *slot#* **b**`data`

Arguments

Argument	Description
slot#	Chassis slot number
value	Built in self test setting (BIST) 1 = BIST available 0 = no BIST available

Examples

```
ROM >>setconfig slot 3 bdata 0<CR>
ROM >>
ROM >>readconfig slot 3 bdata<CR>
Slot 3: board data: no bist
ROM >>
```

`eimage` - Extended Image Name

Set or read an extended diagnostic image name. This command must be executed and the values saved in NVRAM for the diagnostic to run on power up or reset.

Format

setconfig **sl**ot *slot#* **e**image *string*
readconfig **sl**ot *slot#* **e**image

Arguments

Argument	Description
slot#	Chassis slot number
string	Extended diagnostic image name

Examples

```
ROM >>readconfig slot 3 eimage<CR>
Slot 3, extended image: mc4init
ROM >>
ROM >>setconfig slot 3 eimage mc4diag<CR>
ROM >>
ROM >>readconfig slot 3 eimage<CR>
Slot 3, extended image: mc4diag
ROM >>
```

`scsr` - Standard CSR Address

Set or read standard Control/Status Register (CSR) addresses. Three different addresses can be set individually. The `readconfig` command reads all three. The address can be entered as hex or decimal, but is always read as hex.

Format

`set`config **`sl`**ot *slot#* **`sc`**sr (**`1`** | **`2`** | **`3`**) *addr*
`readc`onfig **`sl`**ot *slot#* **`sc`**sr

Arguments

Argument	Description
slot#	Chassis slot number
`1` | **`2`** | **`3`**	1 = board-specific CSR 1 2 = board-specific CSR 2 3 = board-specific CSR 3
addr	CSR address

Examples

```
ROM >>setconfig slot 3 scsr 1 0xfff40800<CR>
ROM >>
ROM >>setconfig slot 3 scsr 2 0xfffc0000<CR>
ROM >>
ROM >>readconfig slot 3 scsr<CR>
Slot 3, csr 1 address: 0xfff40800
Slot 3, csr 2 address: 0xfffc0000
Slot 3, csr 3 address: 0x0
ROM >>
```

`state` - Board State

Set or read the board state. If the command has not been executed for a given slot and this value is read, it is reported as “undefined”.

Format

`set``config` **`sl`**`ot` *slot#* **`st`**`ate (`**`c`**`onfigured | `**`d`**`econfigured)`
`readc``onfig` **`sl`**`ot` *slot#* **`st`**`ate`

Arguments

Argument	Description
slot#	Chassis slot number
`c``onfigured`	Board is configured in NVRAM (if configured, the ROM Monitor attempts to run the extended diag specified with the `set slot ei` command)
`d``econfigured`	Board is not configured in NVRAM

Examples

```
ROM >>readconfig slot 3 state<CR>
Slot 3: board state: undefined
ROM >>
ROM >>setconfig slot 3 state configured<CR>
ROM >>
ROM >>readconfig slot 3 state<CR>
Slot 3: board state: configured
ROM >>
ROM >>setconfig slot 11 state deconfigured<CR>
ROM >>
ROM >>readconfig slot 11 state<CR>
Slot 11: board state: deconfigured
ROM >>
```

`type` - Board Type

Set or read the board type.

Format

set`config` **sl**`ot` *slot#* **t**`ype` (**processor** | **adaptor** | **exm**`em` | **edr**`am` | **v**`me` | **rmsm** |**rmss** | **mcsm** | **mcss**)

readc`onfig` **sl**`ot` *slot#* **t**`ype`

Arguments

Argument	Description
slot#	Chassis slot number
processor	CPU processor board
adaptor	CPU adaptor board
exmem	Local bus memory board
edram	ECC protected local bus memory board
vme	VME board (including the RM board)
rmsm	REFLECTIVE MEMORY master board
rmss	REFLECTIVE MEMORY slave board
mcsm	MEMORY CHANNEL master board
mcss	MEMORY CHANNEL slave board

Examples

```
ROM >>setconfig slot 3 type exmem<CR>
ROM >>
ROM >>readconfig slot 1 type<CR>
Slot 1: board type: adaptor
ROM >>
ROM >>readconfig slot 2 type<CR>
Slot 21: board type: processor
ROM >>
ROM >>readconfig slot 3 type<CR>
Slot 3: board type: exmem
ROM >>
ROM >>readconfig slot 10 type<CR>
Slot 10: board type: vme
ROM >>
```

`vmeconfig` - VME Configuration Commands

Variations of the `setconfig vmeconfig` command set the VMEbus configuration values individually. Variations of the `readconfig vmeconfig` command allow you to read the values individually or all at once.

The variables for the `vmeconfig` command are listed alphabetically in the following pages.

`all` - All VME Values

Read the current state of all VME values.

Format

readconfig **vm**econfig **a**ll

Examples

```
ROM >>readconfig vmeconfig all<CR>

arbiter scheduling: round robin
requester protocol: release when done (fair)
requester interrupt level: 2
service interrupt level: 3
address modifier bits: 0xd
master page bits: 3
slave page bits: 0
More ? (Y/N)y

local page bits: 0
standard space: disabled
extended space: enabled
extended window: disabled
owner timeout: 15
window page bits: 06
ROM >>
```

`extended` - Extended Space

Enable or disable VME extended space or read the state of same.

Format

setconfig **vm**econfig **e**xtended (**e**nable | **d**isable)
readconfig **vm**econfig **e**xtended

Arguments

Argument	Description
enable	Enable extended space
disable	Disable extended space

Examples

```
ROM >>readconfig vmeconfig extended<CR>
extended space: enabled
ROM >>
ROM >>setconfig vmeconfig extended disable<CR>
ROM >>
ROM >>readconfig vmeconfig extended<CR>
extended space: disabled
ROM >>
```

`lpage` - Local Page Bits

Set or read VME local page bits.

Format

setconfig **vm**econfig **lp**age *value*
readconfig **vm**econfig **lp**age

Arguments

Argument	Description
value	Represents local page bits; valid range is 1 through 3

Examples

```
ROM >>setconfig vmeconfig lpage 3<CR>
ROM >>
ROM >>readconfig vmeconfig lpage<CR>
local page bits: 3
ROM >>
```

`modifier` - Address Modifier

Set or read VME address modifier bits.

Format

`set`config **`vm`**econfig **`mo`**difier *value*
`readc`onfig **`vm`**econfig **`mo`**difier

Arguments

Argument	Description
value	Valid range is 0 through 15

Examples

```
ROM >>setconfig vmeconfig modifier 3<CR>
ROM >>
ROM >>readconfig vmeconfig modifier<CR>
address modifier bits: 0x3
ROM >>
```

`mpage` - Master Page Bits

Set or read VME master page bits.

Format

setconfig **vm**econfig **mp**age *value*
readconfig **vm**econfig **mp**age

Arguments

Argument	Description
value	Valid range is 0 through 7

Examples

```
ROM >>setconfig vmeconfig mpage 2<CR>
ROM >>
ROM >>readconfig vmeconfig mpage<CR>
master page bits: 2
ROM >>
```

`priority` - Requester Interrupt Level

Set or read requester interrupt level.

Format

setconfig **vm**econfig **pri**ority *value*
readconfig **vm**econfig **pri**ority

Arguments

Argument	Description
value	Valid range is 0 through 7

Examples

```
ROM >>setconfig vmeconfig priority 2<CR>
ROM >>
ROM >>readconfig vmeconfig priority<CR>
requester interrupt level: 2
ROM >>
```

`protocol` - Requester Protocol

Set or read requester protocol.

Format

`set`config **`vm`**econfig **`pro`**tocol (**`ro`**r | **`rw`**d | **`rwd`**nf)
`readc`onfig **`vm`**econfig **`pro`**tocol

Arguments

Argument	Description
`ro`r	Release on request
`rw`d	Release when done (fair)
`rwd`nf	Release when done (not fair)

Examples

```
ROM >>setconfig vmeconfig protocol ror<CR>
ROM >>
ROM >>readconfig vmeconfig protocol<CR>
requester protocol: release on request
ROM >>
ROM >>setconfig vmeconfig protocol rwd<CR>
ROM >>
ROM >>readconfig vmeconfig protocol<CR>
requester protocol: release when done (fair)
ROM >>
ROM >>setconfig vmeconfig protocol rwdnf<CR>
ROM >>
ROM >>readconfig vmeconfig protocol<CR>
requester protocol: release when done (not fair)
ROM >>
```

`schedule` - Arbiter Scheduling

Set or read arbiter scheduling type for events.

Format

`set`config **`vm`**econfig **`sc`**hedule (**`r`**rs | **`p`**ri)
`readc`onfig **`vm`**econfig **`sc`**hedule

Arguments

Argument	**Description**
`r``rs`	Set round robin arbiter scheduling
`p``ri`	Set priority arbiter scheduling

Examples

```
ROM >>setconfig vmeconfig schedule pri<CR>
ROM >>
ROM >>readconfig vmeconfig schedule<CR>
arbiter scheduling: priority
ROM >>
ROM >>setconfig vmeconfig schedule rrs<CR>
ROM >>
ROM >>readconfig vmeconfig schedule<CR>
arbiter scheduling: round robin
ROM >>
```

`service` - Service Interrupt Level

Set or read the VME service interrupt level.

Format

`set`config **`vm`**econfig **`se`**rvice *value*
`readc`onfig **`vm`**econfig **`se`**rvice

Arguments

Argument	Description
value	VME service interrupt level; valid range is 0 through 3

Examples

```
ROM >>setconfig vmeconfig service 3<CR>
ROM >>
ROM >>readconfig vmeconfig service<CR>
service interrupt level: 3
ROM >>
```

spage - Slave Page Bits

Set or read VME slave page bits.

Format

setconfig **vm**econfig **sp**age *value*
readconfig **vm**econfig **sp**age

Arguments

Argument	Description
value	Slave page bits; valid range is 0 through 3

Examples

```
ROM >>setconfig vmeconfig spage 2<CR>
ROM >>
ROM >>readconfig vmeconfig spage<CR>
slave page bits: 2
ROM >>
```

`standard` - Standard Space

Enable or disable VME standard space or read the state of same.

Format

setconfig **vm**econfig **st**andard (**e**nable **|** **d**isable)
readconfig **vm**econfig **st**andard

Arguments

Argument	Description
enable	Enable VME standard space
disable	Disable VME standard space

Examples

```
ROM >>readconfig vmeconfig standard<CR>
standard space: enabled
ROM >>
ROM >>setconfig vmeconfig standard disable<CR>
ROM >>
ROM >>readconfig vmeconfig standard<CR>
standard space: disabled
ROM >>
```

`timeout` - Owner Timeout

Set or read VME owner timeout value.

Format

setconfig **vm**econfig **t**imeout *value*
readconfig **vm**econfig **t**imeout

Arguments

Argument	Description
value	Valid range is 0 through 15

Examples

```
ROM >>setconfig vmeconfig timeout 7<CR>
ROM >>
ROM >>readconfig vmeconfig timeout<CR>
owner timeout: 7
ROM >>
```

`window` - Extended Window State

Set or read the state of the VME extended window. When enabled, off-board VME masters can access a 128 Mbyte area of VME extended space in memory. After power up or reset, the state of the window is disabled.

Format

`set`config **`vm`**econfig **`wi`**ndow (**`e`**nable | **`d`**isable)
`readc`onfig **`vm`**econfig **`wi`**ndow

Arguments

Argument	Description
`e``nable`	Enable extended space access window
`d``isable`	Disable extended space access window

Examples

```
ROM >>readconfig vmeconfig window<CR>
extended window: disabled
ROM >>
ROM >>setconfig vmeconfig window enable<CR>
ROM >>
ROM >>readconfig vmeconfig window<CR>
extended window: enabled
ROM >>
```

`wpage` - Slave Access

Set or read the state of the VME window bits.

Format

set`config` **vm**`econfig` **wp**`age` *value*
readc`onfig` **vm**`econfig` **wp**`age`

Arguments

Argument	Description
value	Window page bits; valid range is 0 through 31

Examples

```
ROM >>readconfig vmeconfig wpage<CR>
window page bits: 0
ROM >>
ROM >>setconfig vmeconfig wpage 31<CR>
ROM >>
ROM >>readconfig vmeconfig wpage<CR>
window page bits: 31
ROM >>
```

CHAPTER 4

Memory Commands

This chapter describes each command listed in TABLE 4-1. The memory commands perform a full range of memory-related functions. Chapter 1, *Starting and Using the ROM Monitor*, describes command line entry conventions.

TABLE 4-1 Memory Commands

Command	Function
`comparemem`	Compare two memory ranges
`copymem`	Copy memory from range to range
`debug`	Special memory sequence debugger
`disassemble`	Disassemble memory
`fillmem`	Fill memory
`interleave`	Configure memories in interleave mode
`memtest`	Run built-in memory tests on a range
`modmem`	Modify memory
`readmem`	Read memory
`search`	Search a memory range for a pattern
`string`	Search a memory range for a character string
`uninterleave`	Reconfigure the memories uninterleaved

`comparemem` - Compare Memory

Compares one range of memory with another, by the specified data type. Both ranges are the same size. A successful comparison returns only the monitor prompt. A failed comparison displays an error message and all bad comparisons. For a failed comparison, you can choose to continue or terminate the memory compare operation.

Format

`com``paremem` [**`b`** | **`w`** | **`l`**] *addr1 addr2* [*size*]

Arguments

Argument	Description
`b`	Compare bytes (default)
`w`	Compare words
`l`	Compare longwords
addr1	Starting address of range 1
addr2	Starting address of range 2
size	Range size (default is 1) in increments specified by the **`b`** | **`w`** | **`l`** options

Note – Failure to specify addresses on the correct boundary results in an error message. Addresses should be specified as follows:

bytes: 0x3c000000, 0x3c000001, 0x3c000002, 0x3c000003,...
words: 0x3c000000, 0x3c000002, 0x3c000004, 0x3c000006,...
longwords: 0x3c000000, 0x3c000004, 0x3c000008, 0x3c00000c,...

Examples

```
ROM >>comparemem b 0x3c000000 0x3c000010 0x10<CR>
Memory did not compare
0x3c000002 = 0x23,     0x3c000012 = 0x24
Continue ? (Y/N)
0x3c000008 = 0xab,     0x3c000018 = 0xbb
Continue ? (Y/N)
0x3c00000d = 0x04      0x3c00001d = 0x14

ROM >>

ROM >>comparemem l 0x3c000000 0x3c000010 4<CR>
Memory did not compare
0x3c000000 = 0x00010203,    0x3c000010 = 0x10111213
Continue ? (Y/N)

ROM >>

ROM >>comparemem w 0x3c000000 0x3c000010 0x04<CR>

ROM >>
```

`copymem` - Copy Memory Range

Copies one range of memory to another. The data in the source range is unchanged by the execution of this command.

Format

copymem [**b** | **w** | **l**] *src dst size*

Arguments

Argument	Description
b	Copy bytes
w	Copy words
l	Copy longwords
src	Starting address of source range
dst	Starting address of destination range
size	Size of both ranges in increments specified by the **b** | **w** | **l** option

Examples

```
ROM >>copymem b 0xfc001000 0xfc002000 0x64<CR>

ROM >>

ROM >>copymem l 0xfc001000 0xfc002000 25<CR>

ROM >>
```

Note – Both examples copy the same amount of data (25 decimal longwords is equal to 64 hexadecimal bytes).

debug - Memory Debugger Program

Activates a debugger, allowing single memory operations (read | write | compare) to be performed individually or linked together and executed as a program loop. The console responds with a menu system for entering single commands or program loops. A <CR> is not needed after the command entry.

Note – Proper operation requires a terminal with VT100 cursor control capability.

Format

debug

Examples

```
ROM >>debug<CR>
```

A menu is displayed for the operator to follow:

```
Hi-Rent Debugger
  r. Read Data
  w. Write Data
  l. Interlock
  p. Perform Loop Sequence
ESC. Exit Debugger

Enter: r, w, l, p, ESC  ==><ESC>
ROM >>
```

Note – Pressing the ESC key terminates the debugger (when allowed) and returns the ROM Monitor prompt.

In this debugger, all data and address entries are assumed to be hexadecimal so the `0x` prefix is not accepted.

Write Data

Typing a `w` (automatic <CR>) opens a dialogue with the operator. The menu prompts the operator to enter the data type followed by the address where the data is to be written and a <CR>. Finally, enter data and a <CR>. An example of this operation follows.

```
Enter: r, w, l, p, ESC ==>w
   enter data type  b  h  w  w
   enter address:  3f000100
   enter data:  1234abcd
```

The debugger echoes the address and new data just above the menu/entry line:

```
   Address:  3f000100      Data Written:  1234abcd
```

Read Data

Typing an `r` produces the same sequence as for a write, but there is no prompt to enter data. The debugger returns the address and data just above the menu/entry line.

Interlock

Typing an `l` performs a special semaphore cycle on a specified location, where it reads the data currently in the location, stores it in a CPU register, and then writes specified data to the location. It is a diagnostic tool for development or repair.

Perform Loop Sequence

Typing a `p` causes the debugger to enter a mode where a series of commands can be entered and run as a program loop. Once started, the program loops until the operator intervenes. The debugger displays the following loop menu from which to select a sequence of functions.

```
  r. Read Data  read (b/w/d) at (addr)
  w. Write Data  write (b/w/d) at (addr) (data)
  l. Lock Memory Location  interlock (b w/d) at (addr) (data)
  v. Increment Variable
     Read Address  increment (b/w/d) last read address
  x. Increment Variable
     Write Address  increment (b/w/d) last write address
  c. Compare Last Data/  compare (b/w/d) last data/address read
     Address to Value  to (value)
  p. Print Last Data Read  display last address and data
      read (display on one line or scroll)
  i. Start of If Sequence  if (eq/neq/>/<) last compare, execute
      commands in the If sequence.
      [ if sequence ]
  f. End of If Sequence
  s. Start of Loop Sequence  loop (address increment count)
      { loop sequence }
  o. End of Loop Sequence
  h. Halt Until Key Depressed suspend loop execution
  d. Delay for Time Period
     Specified  delay (time)
  e. Execute Sequence
ESC. Exit Debugger

Enter: r, w, l, v, x, c, p, i, f, s, o, h, d, e, ESC ==>
```

Note – After commands and their arguments are entered, the command letters are displayed in sequence in a single history line above the entry line. This provides a visual record of the loop that remains during execution and until returning to the loop menu.

Read Data. The `r` command prompts the operator for the data type and the choice of fixed or variable address (automatic <CR>), and the address.

Write Data. The `w` command prompts the operator for the data type and the choice of fixed or variable address (automatic <CR>), the address, and the data.

Lock Memory Location. The `l` command does a special semaphore cycle on a specified location; it reads the data currently in the location, stores it in a CPU register, and then writes the specified data to the location. It is a diagnostic tool for development or repair.

Increment Variable Address. The `v` command increments the last read address accessed by one, two or four bytes, depending on the last data type selected. The `x` command does the same thing to the last write address accessed. This works only with loops using the `s` command (with a loop count) and the `r` and `w` command with the variable option selected.

Compare Last Data or Address Read to Value. The `c` command prompts the operator to choose whether to compare a read address, write address, or data, and to supply a comparison value. The result of the comparison (equal/not equal/ greater than / less than) is used as the basis for a subsequent `if` sequence.

Print Last Data Read. The `p` command displays the address and data from the last `r` command executed. Invoking the command prompts the operator to select either a one-line display, where each display overwrites the previous one, or a scrolling display, as follows:

```
     (o)ne line print / (s)croll print ==>   type o or s
```

Note – The scroll print option is useful only for intermittent output. A steady stream of output displays too fast to be read.

Start of If Sequence. The `i` command prompts for a mathematical operator relating to the last comparison made. The start of the if sequence is indicated by `[i,` in the history line.

[if sequence]. One or more commands can be placed inside the brackets of the `if` sequence. Execution of these commands depends on whether the mathematical operator agrees with the result of the last comparison. For example, if the last unit of data read was equal to the specified comparison value and the equal operator was specified in the `if` command, the commands in the brackets are executed.

End of If Sequence. When the sequence of conditional commands has been entered, the operator enters the `f` command. The end of the `if` sequence is indicated by `f]`, in the history line. The entire `if` sequence looks like:

```
[i, cmd, cmd, cmd, f]
```

Start of Loop Sequence. The `s` command prompts the operator for a hexadecimal loop count. This count is relevant only to loops containing `r` or `w` commands in which variable addressing is specified. It determines the (hex) number of times to increment the address before it restores the address to its original value and starts over. The specification must be for n-1 iterations. For example, to read 16_{10} units of data, specify a loop count of f (decimal 16 - 1). This is the first command letter of the loop and is displayed as `{s,`.

{ loop sequence }. This is the loop program, which is a series of commands and/or `if` sequences.

End of Loop Sequence. After the last command in the loop has been entered, the operator enters the `o` command, indicating the end of the loop. This is displayed in the history line as `o},`. An entire loop sequence looks like:

{s, *cmd*, *cmd*, **[i**, *cmd*, *cmd*, **f]**, *cmd*, *cmd*, **o}**,

---

Note – If multiple loop sequences are entered on a single history line, only the first such sequence is executed.

Halt Until Key Depressed. When the `h` command is encountered in the loop sequence, program execution stops and waits for the operator to press a key.

Delay For Time Period Specified. The `d` command prompts the operator to enter a delay time value. When this command is executed, loop execution is suspended until the delay times out. This allows time to see the result of displaying several locations in sequence.

Execute Sequence. The `e` command puts the loop sequence into execution. Loop sequences run until the operator intervenes by pressing a key. Pressing a key again returns to the loop menu.

ESC. The Escape key terminates the debugger session and returns the ROM Monitor prompt.

Loop Sequence Examples

Several examples of using loop sequences are provided here. Refer to the preceding command definitions as needed.

1. The following sequence turns user LEDs 0 through 3 on and off at a rate determined by the delay value supplied with the `d` command.

Note – The addresses of the LEDs vary between processor boards. Consult the system administrator for the correct LED addresses if these do not work.

---

a. Type `w` and write 1 to a fixed byte at fff5c000 (address of LED0).

b. Type `d` for a time delay. Try 1000. The larger the number, the longer the delay.

c. Write 0 to a fixed byte at the same address.

d. Delay same time.

e. Write 1 to a fixed byte at fff5c200 (address of LED1).

f. Delay same time.

g. Write 0 to a fixed byte at the same address.

h. Delay same time.

i. Write 1 to a fixed byte at fff5c400 (address of LED2)

j. Delay same time.

k. Write 0 to a fixed byte at the same address.

l. Delay same time.

m. Write 1 to a fixed byte at fff5c600 (address of LED3).

n. Delay same time.

o. Write 0 to a fixed byte at the same address.

p. Delay same time.

q. Type `e` to execute the loop.

The history line for this loop looks like this:

```
w, d, w, d, w, d, w, d, w, d, w, d, w, d, w, d, e,
```

2. The following sequence cycles all user LEDs on and off with fewer commands than in step 1, using variable addressing. This sequence takes longer to cycle between LEDs because the LED addresses are spaced some distance apart and variable addressing increments consecutive addresses, according to the data type selected.

   a. Type `s` and set up a loop count for 400 increments (this hex number increments the address far enough to include all the LED addresses).

   b. Write 1 to a variable word at fff5c000 (address of LED 0).

   c. Delay long enough to see the state of the LED.

   d. Write 0 to a variable word. The address is automatic (variable addressing).

   e. Delay long enough to see the state of the LED.

   f. Type `x` to increment the write address.

   g. Type `o` to close the variable loop.

   h. Execute the loop.

The history line for this loop looks like this:

```
{s, w, d, w, d, x, o}, e,
```

3. The following sequence writes a data pattern to memory outside the debugger, and then uses the debugger to read and display the data until the operator stops the loop.

   a. Set up a buffer with the `fillmem` command. Use the `-i` option to write incremental data.

```
fillmem b 0x(address) 20 -i(data)<CR>
```

Note – Be sure to fill an area that is safe to write into.

   b. Enter the `debug` command and get into the loop menu.

   c. Set up a loop count for 13 increments (this is 19 decimal).

   d. Read a byte in variable addressing mode from the same address used in `fillmem`, (step a).

   e. Display the data read.

f. Delay long enough to see the data.

g. Type `v` to increment the read address.

h. Close the variable loop.

i. Execute.

The history line for this loop looks like this:

```
{s, r, p, d, v, o}, e,
```

If the arguments have been entered correctly, the address and data increment within the limits of the loop count, start over again, and run until terminated.

disassemble - Disassemble Memory

Disassembles machine language in memory, starting at *addr*, and returns assembly language. The output fills the display with addresses and assembly language, and then stops scrolling to allow visual examination. Press the space bar to continue scrolling, any other key to quit.

Format

disassemble *addr*

Arguments

Argument	**Description**
addr	Starting address of disassembly

Examples

```
ROM >>disassemble 0x3c100000<CR>
assembly language<SPACE>
more assembly language<SPACE>
more assembly language<ANY OTHER KEY>

ROM >>
```

`fillmem` - Fill Memory

Fills a range of memory with a specified value. The only response to this command is the ROM monitor prompt.

Format

`fillmem [b | w | l] addr size [-i]data`

Arguments

Argument	Description
b	Fill bytes
w	Fill words
l	Fill longwords
addr	Starting address of range
size	Size of range in increments specified by **b** | **w** | **l** option
-i	Increment data value by one **b** | **w** | **l** after each memory write
data	Data to be stored

Examples

```
ROM >>fillmem w 0x3c000004 0x10 0x5555<CR>
ROM >>
ROM >>fillmem b 0x3c002000 0x20 -ia0xa0<CR>
ROM >>
```

Note – See the last example in the `readmem` command description to see the result of the `fillmem b` example.

`interleave` - Interleave Expansion Memory

Automatically determines which configured expansion memory boards in the system can be interleaved and sets the appropriate values required to enable interleaving.

The operator can examine the new memory configuration and abort the operation before the memory configuration or NVRAM is modified.

Pressing the `y` key modifies the memory configuration and responds with a verification message and the monitor prompt. Pressing any other key aborts the operation before it modifies the memory configuration, and returns the monitor prompt.

Format

`interleave` [*size*]

Arguments

Argument	Description
size	Memory segment size in bytes for interleaving (16/32/64/128/256)

Examples

```
ROM >>interleave 16<CR>
Mem brd 1, slot 3, 64Mb, not interleaved
Mem brd 2, slot 4, 16Mb, interleave position 1, interleave size 16
bytes
Mem brd 3, slot 5, 16Mb, interleave position 2, interleave size 16
bytes
Nvram will be modified!      Continue ? (Y/N) y
nvram modified

ROM >>
```

memtest - Memory Test

Runs *loopcount* iterations of the built-in memory tests on a specific range of memory. If *loopcount* equals 0, the tests run until the `<ESC>` key is pressed.

Format

me`mtest` `[`**b** **|** **w** **|** **l**`]` *addr size loopcount*

Arguments

Argument	Description
b	Test bytes
w	Test words
l	Test longwords
addr	Starting address of test range
size	Size of range in increments specified by the **b** | **w** | **l** option
loopcount	Number of iterations desired. For continuous testing, specify 0

Examples

```
ROM >>memtest b 0xfc002000 0x100 10<CR>

Progress messages

ROM >>
ROM >>memtest w 0xfc001000 1000 0<CR>

Progress messages
     .
     .
     .
Progress messages
<ESC>

ROM >>
```

`modmem` - Modify Memory

Modifies one or more memory locations with a specified value. A list of command functions is always displayed after a <CR>. This list is followed by the first address to be modified, its current contents, and a prompt >>. At this point, any function in the list can be performed. Terminate operator input with the `<CR>` key, and end the modify session with the `<ESC>` key, at which time the normal prompt appears. When data is entered in response to the >> prompt, it is automatically echoed on the next line.

TABLE 4-2 Available Functions for the `modmem` Command

Input	Function
`<CR>`	Display next location
`+`	Display next location
`-`	Display previous location
`0x`*data*	Store hex data at current location
data	Store decimal data at current location
`?`	Display this message
`<ESC>`	Return to CLI prompt

Note – Addresses and data can be entered in hex, binary, or decimal but are always displayed in hex, as denoted by the `0x` prefix.

Maximum data entry values must be consistent with the data type specified on the command line. Exceeding these maximums produces unreliable results.

Format

modmem [**b** | **w** | **l**] *addr*

Arguments

Argument	Description
b	Modify bytes
w	Modify words
l	Modify longwords
addr	First address to be modified

Examples

```
ROM >>modmem b 0x3c000002<CR>
Modify Memory
   CR .... display next location
    + .... display next location
    - .... display previous location
 0x12 .... store hex data at current location
   12 .... store decimal data at current location
    ? .... display this message
  ESC .... return to CLI prompt

0x3c000002: 0xFF  >> <CR>      (display next location)
0x3c000003: 0x10  >> 0x55<CR>  (enter hex data)
0x3c000003: 0x55  >> -         (display previous location)
0x3c000002: 0xFF  >> 85<CR>    (enter decimal data)
0x3c000003: 0x55  >> +         (display next location)
0x3c000002: 0x55  >> ESC       (return to CLI prompt)

ROM >>
ROM >>modmem w 0x3c000004<CR>
Modify Memory
  CR ..... display next location
   + ..... display next location
   - ..... display previous location
0x12 ..... store hex data at current location
  12 ..... store decimal data at current location
   ? ..... display this message
 ESC ..... return to CLI prompt

0x3c000004: 0xFFFF  >>1234<CR> (enter decimal data)
0x3c000004: 0x04d2  >> <ESC>   (display hex equivalent)

ROM >>
ROM >>modmem 1 0x3c000008<CR>
Modify Memory
  CR ..... display next location
   + ..... display next location
   - ..... display previous location
0x12 ..... store hex data at current location
  12 ..... store decimal data at current location
   ? ..... display this message
 ESC ..... return to CLI prompt

0x3c000008: 0xFFFFFFFF  >> <CR>          (display next location)
0x3c00000C: 0x10000000  >> 0x55555555<CR>(enter hex data)
0x3c00000C: 0x55555555  >> -<CR>         (display previous location)
0x3c000008: 0xFFFFFFFF  >> 1431655765<CR>(enter decimal data)
0x3c000008: 0x55555555  >> <ESC>         (display hex equivalent)

ROM >>
```

`readmem` - Read Memory

Reads and displays a memory range. The minimum (and default) amount of data returned without using a size argument is:

- 16 bytes
- 8 words
- 4 longwords

Returned data is displayed as hex digits, with the ASCII equivalent on the right. A period (.) is inserted in any column containing an unprintable hex value.

Format

readmem [**b** | **w** | **l**] *addr* [*size*]

Arguments

Argument	Description
b	Read bytes
w	Read words
l	Read longwords
addr	Starting address of range
size	Size of range in increments specified by the **b** | **w** | **l** option

Examples

```
ROM >>readmem l 0x3c000000 8<CR>
0x3c000000: 01234567 89abcdef fedcba98 76543210.#Eg........vT2.
0x3c000010: 31323334 41424344 4c6f6f6b 686572651234ABCDLookhere

ROM >>

ROM >>readmem w 0x3c000000 0x10<CR>
0x3c000000: 0123 4567 89ab cdef fedc ba98 7654 3210
.#Eg........vT2.
0x3c000010: 3132 3334 4142 4344 4c6f 6f6b 6865 7265
1234ABCDLookhere

ROM >>

ROM >>readmem b 0x3c002000 0x20<CR>
0x3c002000: a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af
    ................
0x3c002010: b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf
    ................

ROM >>
```

Note – In each example the same amount of memory was read. Only the data presentation format changed.

`search` - Search Memory

Searches a block of memory for a specified value. Memory locations containing the search data value are displayed. However, if the search data value is immediately preceded by an exclamation point (!), then all locations in the search range that do not contain the search data value are listed.

Format

sea`rch` [**b** | **w** | **l**] *addr* *size* [**!**]*data*

Arguments

Argument	Description
b	Search for bytes
w	Search for words
l	Search for longwords
addr	Starting address of search range
size	Size of search range in increments specified by the **b** | **w** | **l** option
!	Searches for absences of the search data value
data	Search data value

Examples

```
ROM >>search b 0x3f000000 0x100 0x12<CR>
0x3f000010: 0x12
0x3f0000f0: 0x12

ROM >>

ROM >>search l 0x3f000000 0x100 !0x0<CR>
0x3f000010: 0x12ffffff
0x3f0000c0: 0xffffffff
0x3f0000f0: 0x12000000
0x3f0000fc: 0xa5a5a5a5

ROM >>
```

`string` - String Search

Searches a block of memory for a specified character string. If the string is found within the search range, it is displayed. The starting address where the string is found is displayed, followed by the hexadecimal data and the ASCII equivalent. If the string is not found, only the monitor prompt is returned.

Format

`string` *addr size string*

Arguments

Argument	Description
addr	Starting address of search range
size	Size of search range in bytes (characters)
string	Character string (79 characters maximum) terminated by a space, a colon, a comma, or a carriage return

Examples

```
ROM >>string 0x0003cd00 0x100 I_found_it!<CR>
0x0003cd90: 49 5f 66 6f 75 6e 64 5f 69 74 21 00 00 00 00 00
I_found_it!.....

ROM >>
```

`uninterleave` - Uninterleave Expansion Memory

Automatically uninterleaves all configured expansion memory boards in the system and sets the appropriate values required to disable interleaving.

The operator can examine the new memory configuration and abort the operation before the memory configuration or NVRAM is modified. Pressing the `y` key modifies the memory configuration. The system responds with a verification message and the monitor prompt. Pressing any other key aborts the operation before it modifies the memory configuration, and returns the monitor prompt.

Format

u`ninterleave`

Examples

```
ROM >>uninterleave<CR>

Mem brd 1, slot 3, 64Mb, not interleaved
Mem brd 2, slot 4, 16Mb, not interleaved
Mem brd 3, slot 5, 16Mb, not interleaved
Nvram will be modified!      Continue ? (Y/N) y
nvram modified

ROM >>
```

CHAPTER 5

Miscellaneous Commands

This chapter describes the software development commands, disk/tape commands, `test` command, and remote support command. The software development commands are listed in TABLE 5-1. The disk/tape commands are listed in TABLE 5-2.

TABLE 5-1 Software Development Commands

Command	Function
`go`	Send specified processor to address
`mgo`	Send all processors to address
`srecord`	Load S-records

TABLE 5-2 Disk/tape Commands

Command	Function
`readblock`	Read logical blocks
`rewind`	Rewind tape drive
`scsirwc`	SCSI read/write/compare test
`skipfilem`	Skip file marks
`taperwc`	Tape read/write/compare test
`writeblock`	Write logical blocks

Software Development Commands

Software development commands are listed in detail in the following pages. Refer to Chapter 1, *Starting and Using the ROM Monitor*, for command line entry rules.

go - Start CPU at Address

Starts the specified CPU on the processor board at the specified memory address. Executable code must be available at the target address. This command is normally used to start a task (such as a diagnostic program) after loading it into memory from a device. This command returns no ROM monitor prompt because it passes control to the program located at the address specified in the command. If no arguments are passed, *cpu #* defaults to 0 and *addr* defaults to `0x3f600000`.

Format

go [*cpu #*] [*addr*]

Arguments

Argument	Description
cpu #	Specified CPU; valid range is 0 through 3 on a 4-CPU configuration and 0 through 7 on an 8-CPU configuration
addr	Address at which to begin execution

Examples

For a 4-CPU configuration:

```
ROM >>go 0 0x3c040000<CR>

ROM >>
```

Executable program here, followed by a return to the CLI.

```
ROM >>go 7 0x3f600000<CR>
Invalid argument: cpu = 7

cpu value must be between 0 and 03

ROM >>
```

For an 8-CPU configuration:

```
ROM >>go 9 0x3f600000<CR>

Invalid argument: cpu = 9

cpu value must be between 0 and 07

ROM >>
```

`mgo` - Start All CPUs

Starts all CPUs at the specified memory address. Executable code must be available at the target address. This command is used when all CPUs must be started together, such as after loading the `sysboot` utility. This command returns no ROM monitor prompt because it passes control to the program located at the address specified in the command.

Format

`mg``o (`*addr*`)`

Arguments

Argument	Description
addr	Address at which to begin execution

Examples

```
ROM >>mgo 0x3c001000<CR>
```

Executable program here, followed by a return to the CLI.

`srecord` - Load Extended Exerciser Records

Loads Motorola Extended Exerciser S-records from an RS-232 serial line through the specified serial port and stores them in memory starting at the address specified. A serial input device must be attached to the specified port.

Note – The ROM monitor will wait as long as necessary for the requested data. This can give the appearance of a hung system. Use the <ESC> key or Control-C to terminate the command.

Format

sr`ecord` *port #* *addr*

Arguments

Argument	**Description**
port #	Specifies the port number from 0 through 3.
addr	Starting memory address at which to store data

Examples

```
ROM >>srecord 1 0x3c001000<CR>

ROM >>
```

Disk/Tape Commands

Disk/tape commands are listed in detail on this and the following pages. See Chapter 1, *Starting and Using the ROM Monitor*, for command line entry rules.

`readblock` - Read Logical Block

Reads logical blocks from a device and stores data in memory. The response is a prompt (and an error message, if any errors are detected).

Format

`readb``lock` (*bus*) (*chan*) (*target*) (*lun*) (*addr*) (*startblock*) (*nblocks*)

Arguments

Argument	Description	
bus	Index or a hex value of a valid VME short address	
	Index #	**Description**
	0	On-board SCSI
	1	VME Controller 0x6800
	2	VME Controller 0x7000
	3	VME Controller 0x7800
	.	
	.	
	.	
	15	VME Controller 0xd800
chan	SCSI channel 0 or 1	
target	SCSI target devices 0 through 7	
lun	SCSI logical device 0 through 7	

Argument	Description
addr	Starting memory address at which to store data
startblock	Starting block number on the device
nblocks	Number of blocks to read

Examples

```
ROM >>readblock 0 0 6 0 0x3f000000 22620 1000<CR>

ROM >>
```

Note – Only channel 0 is supported when using VME SCSI boot. Make sure the boot device is on channel 0 under this configuration.

`rewind` - Rewind Tape

Rewinds the specified tape device to the beginning of tape (BOT) marker.

Format

rew`ind` (*chan*) (*target*) (*lun*)

Arguments

Argument	Description
chan	SCSI channel 0 or 1
target	SCSI target devices 0 through 7
lun	SCSI logical device 0 through 7

Examples

```
ROM >>rewind 0 5 0<CR>

ROM >>
```

`scsirwc` - SCSI Read / Write / Compare

Reads the diagnostic tracks into memory, writes them back out to disk, compares them, and then reports drive geometry and diagnostic results to the operator.

Format

`scsirwc (`*bus*`) (`*chan*`) (`*target*`) (`*lun*`)`

Arguments

Argument	Description	
bus	Index or a hex value of a valid VME short address	
	Index #	**Description**
	0	On-board SCSI
	1	VME Controller 0x6800
	2	VME Controller 0x7000
	3	VME Controller 0x7800
	.	
	.	
	.	
	15	VME Controller 0xd800
chan	SCSI channel 0 or 1	
target	SCSI target devices 0 through 7	
lun	SCSI logical device 0 through 7	

Note – Only channel 0 is supported when using VME SCSI boot. Make sure the boot device is on channel 0 under this configuration.

Examples

```
ROM >>scsirwc 0 0 6 0<CR>
Big Endian format detected!
Header Partition Image Rev 3.0.9
Disk Name: Sun Microsystems
Layout flags = 0x1
Image area descriptor:897536 bytes of images, descriptors at byte
17920
total_blocks = 71, track_cylinder = 15
cylinders = 1931, total_avail_blocks =2031705
avail_sect_cylinder = 1065, end cylinders = 24
There are 8 partitions out of a potential 64
Part 15 size 2130,offset 2028825,type 7=Diagnostic Partition,
nameoff 3840
Part name diag
Header Partition is at offset 0, size 4260
SCSI R/W/C Diagnostic passed

ROM >>
```

`skipfilem` - Skip Filemarks

Skips filemarks on the specified tape device.

Format

sk`ipfilem` (*chan*) (*target*) (*lun*) (*nfilemarks*)

Arguments

Argument	Description
chan	SCSI channel 0 or 1
target	SCSI target devices 0 through 7
lun	SCSI logical device 0 through 7
nfilemarks	Number of filemarks to skip (0 is invalid)

Examples

```
ROM >>skipfilem 0 5 0 2<CR>

ROM >>
```

`taperwc` - Tape Read / Write / Compare

Writes a 5K test pattern to tape, reads it back into DRAM, and compares it with the original data pattern. If any of the tape commands used to perform this test fail, or if the memory comparison fails, an appropriate message is logged to the operator. Ensure that the tape used is a scratch tape, as any previous data on the tape will be destroyed. Also ensure that the tape is not write protected.

Format

taperwc (*chan*) (*target*) (*lun*)

Arguments

Argument	Description
chan	Tape channel 0 or 1
target	Tape target devices 0 through 7
lun	Tape logical device 0 through 7

Examples

```
ROM >>taperwc 1 3 0<CR>
rewinding tape
writing test pattern to tape
rewinding tape
reading test pattern from tape
comparing memory:  ok
rewinding tape
TAPE R/W/C Diagnostic passed.
ROM >>
```

`writeblock` -Write Logical Block

Writes logical blocks to a device. The source of write data is main memory.

Format

`writeblock` (*bus*) (*chan*) (*target*) (*lun*) (*addr*) (*startblock*) (*nblocks*)

Arguments

Argument	Description	
bus	Index or a hex value of a valid VME short address	
	Index #	**Description**
	0	On-board SCSI
	1	VME Controller 0x6800
	2	VME Controller 0x7000
	3	VME Controller 0x7800
	.	
	.	
	.	
	15	VME Controller 0xd800
chan	SCSI channel 0 or 1	
target	SCSI target devices 0 through 7	
lun	SCSI logical device 0 through 7	
addr	Starting memory address from which to read data	
startblock	Starting block number on the device	
nblocks	Number of blocks to write	

Note – Only channel 0 is supported when using VME SCSI boot. Make sure the boot device is on channel 0 under this configuration.

Examples

```
ROM >>writeblock 0 0 6 0 0x3f000000 22620 1000<CR>

ROM >>
```

`test` Command

The `test` command is used only in the ROM Monitor. Refer to Chapter 1, *Starting and Using the ROM Monitor*, for command line entry rules.

`test` - Test Various Components

The ROM Monitor allows testing various parts of the system, either automatically at power up/reset or on command. The `test` command executes built-in self tests at the specified levels. Any combination of levels may be specified. If no level is specified, the current autotest level, as previously set by the `setconfig tlevel` command, is used. The response to this command is a set of completion status messages.

Note – Setting (enabling) the burnin flag causes tests to loop until a key is pressed. See the `burnin` command description.

Format

test [**ca**che | **co**re | **e**cc | **lm**em | **lp**bk | **q**mem | **sc**si | **sl**ots | **ed**ramecc]

Arguments

Argument	Description
cache	Cache memory tests
core	Processor board components test
ecc	ECC memory tests
lmem	Various algorithm tests of on-board and expansion memory
lpbk	All external loopback tests (external hardware required)
qmem	MBUS fairness tests of onboard and expansion memory

Argument	Description
scsi	SCSI interface tests along with tests of reads and writes to disk
slots	Any configured extended diagnostics
edramecc	EDRAM ECC memory tests

Examples

```
ROM >>test<CR>

Test levels selected:  core  qmem  slots

Testing CPU Card
Starting EPROM checksum test:
    Testing new PROMs
    CPU 0, Read checksum 29b81b8e calculated 29b81b8e...PASS
Starting Parameter prom checksum test:
    CPU 0, calc checksum fffffb7e, stored fffffb7e...PASS
Starting DRAM Parity test:
    Bypassed for this configuration
    - no Adaptor Memory present
Starting SRAM Parity test:
...PASS
Starting TS80 X-cpu int test:
    CPU 1 interrupting: CPU 0, CPU 1, CPU 2, CPU 3...PASS
    CPU 3 interrupting: CPU 0, CPU 1, CPU 2, CPU 3...PASS
    CPU 2 interrupting: CPU 0, CPU 1, CPU 2, CPU 3...PASS
    CPU 0 interrupting: CPU 0, CPU 1, CPU 2, CPU 3...PASS
Starting TS80 xint test:
    CPU 1 testing xint 0, 1, 2, 3, 4, 5, 6, 7, ...PASS
    CPU 0 testing xint 0, 1, 2, 3, 4, 5, 6, 7, ...PASS
    CPU 3 testing xint 0, 1, 2, 3, 4, 5, 6, 7, ...PASS
    CPU 2 testing xint 0, 1, 2, 3, 4, 5, 6, 7, ...PASS
Starting TS80 int cpu test:
    CPU = 1, Timer: 0, 1, 2, 3, 4, 5...PASS
    CPU = 0, Timer: 0, 1, 2, 3, 4, 5...PASS
    CPU = 3, Timer: 0, 1, 2, 3, 4, 5...PASS
    CPU = 2, Timer: 0, 1, 2, 3, 4, 5...PASS
Starting TS80 Timer int test:
    CPU: 1, timer 0, 1, 2, 3, 4, 5...PASS
    CPU: 0, timer 0, 1, 2, 3, 4, 5...PASS
    CPU: 3, timer 0, 1, 2, 3, 4, 5...PASS
    CPU: 2, timer 0, 1, 2, 3, 4, 5...PASS
Starting Ethernet controller chip test:
...PASS
Starting Ethernet internal loopback test:
...PASS
Starting Ethernet external loopback test:
...PASS
Starting VME intr bsy test:
...PASS
```

```
Starting VME interrupt test:
   CPU 0 testing vme interrupt: 1 2 3 4 5 6 7 ...PASS
Starting SCC internal loopback test:
   port: 0 ....RESERVED (console port)
   port: 1 ....PASS
   port: 2 ....PASS
   port: 3 ....PASS
Starting SCSI regs test:
CPU 0 reading icode istat dmastat stat0 stat2 ID ...PASS
Starting System Bus Parity test:
...PASS
Starting DRAM DATA=ADDR test:
   Bypassed for this configuration
   - no Adaptor Memory present
Starting DRAM address line test:
   Bypassed for this configuration
   - no Adaptor Memory present
Starting Multi CPU fairness test:
   Bypassed for this configuration
   - no Adaptor Memory present
Starting Multi Data Size test:
   Bypassed for this configuration
   - no Adaptor Memory present
Starting DRAM DATA=ADDR test:
   DRAM DATA=ADDR 3e100000-40000000 with 4 cpus
      cpu 0 testing 8126464 bytes starting at 3e100000
      cpu 1 testing 8126464 bytes starting at 3e8c0000
      cpu 2 testing 8126464 bytes starting at 3f080000
      cpu 3 testing 8126464 bytes starting at 3f840000
      fill, chk...PASS
Starting DRAM address line test:
   DRAM address line BYTE, from 3e100000-40000000...PASS
   DRAM address line WORD, from 3e100000-40000000...PASS
   DRAM address line LONG, from 3e100000-40000000...PASS
Starting Multi CPU fairness test:
   CPU 0, 99984, CPU 1, 100000, CPU 2, 100000, CPU 3, 100000, PASS
Starting Multi Data Size test:
   Multi Data Size 3e100000-3e200000 with 4 cpus
      (lfill-bchk), (bfill-lchk), (lfill-wchk), (wfill-lchk),
      (lfill-dchk), (dfill-lchk), (bfill-wchk), (wfill-bchk),
    (bfill-dchk), (dfill-bchk), (wfill-dchk), (dfill-wchk)...PASS

Testing Memory Cards
Starting Expansion Memory Parameter prom checksum test:
...PASS
```

```
Starting DRAM DATA=ADDR test:
    DRAM DATA=ADDR 40000000-60000000 with 4 cpus
        cpu 0 testing 134217728 bytes starting at 40000000
        cpu 1 testing 134217728 bytes starting at 48000000
        cpu 2 testing 134217728 bytes starting at 50000000
        cpu 3 testing 134217728 bytes starting at 58000000
        fill, chk...PASS
Starting DRAM address line test:
    DRAM address line BYTE, from 40000000-60000000...PASS
    DRAM address line WORD, from 40000000-60000000...PASS
    DRAM address line LONG, from 40000000-60000000...PASS
Starting VBUS DATA=ADDR test:
    VBUS DATA=ADDR 40000000-60000000 with 4 cpus
     Master Page: 2
        cpu 0 testing 134217728 bytes starting at c0000000
        cpu 1 testing 134217728 bytes starting at c8000000
        cpu 2 testing 134217728 bytes starting at d0000000
        cpu 3 testing 134217728 bytes starting at d8000000
        fill, chk...PASS
Starting Multi CPU fairness test:
    CPU 0, 99992, CPU 1, 100000, CPU 2, 100000, CPU 3, 100000, PASS
Starting Multi CPU Vbus fairness test:
    CPU 0, 99983, CPU 1, 100000, CPU 2, 100000, CPU 3, 100000, PASS
Starting Expansion Memory Parity test:
    Bypassed for this configuration
    No expansion memory board configured.
Starting Expansion Memory Interleave test:
    Bypassed for this configuration
    No expansion memory board configured.
Starting Multi Data Size test:
    Multi Data Size 40000000-40100000 with 4 cpus
    (lfill-bchk), (bfill-lchk), (lfill-wchk), (wfill-lchk),
    (lfill-dchk), (dfill-lchk), (bfill-wchk), (wfill-bchk),
    (bfill-dchk), (dfill-bchk), (wfill-dchk), (dfill-wchk)...PASS
```

```
PASSES THROUGH TEST LIST = 1    ERRORS = 0

Testing Backplane Cards
Slot 3: Loading extended image mc4diag
Extended Memory Channel IV Diagnostic
Diagnostic Revision: 1.15
MCA VME CSR = 0x00000000
EDRAM VME CSR = 0x00000000
EDRAM VME Base Addr = 0x40000000
EDRAM LBus Base Addr = 0x40000000
Slot: 3, Board CSR Addr: 0x00000000 0x000007ff
Board Id: 0x00000000 (???), Major Rev: 0x00000001, Minor Rev:
0x00000001
EDRAM Id: 0x00000210, Major Rev: 0x00000008, Minor Rev: 0x00000000
On Board Memory Size: 512mb, Physical Node Id: 0x0
PASS
Slot 4: Loading extended image atmdiag
Extended VATM Diagnostic
Diagnostic Revision: 1.1
Slot: 4, Board CSR Addr: 0xfff4f000 - 0xfff4f7ff
Type SONET 155 Mbps OC3 (6)
Firmware Version and Date : 5215 V/ATM A06   06/21/98 10:30
CPU Type = 0x68040    Flash Size = 0x20000
VRAM Size = 0x400000
Number of VCs = 0x800   Jumper Field = 0x967f

Searching for Scsi Devices on Channel A:
....................................
Device: 0 0 3 0, tape drive
Device: 0 0 5 0, disk drive
Device: 0 0 6 0, disk drive

You have 7 seconds to terminate the autoboot sequence.  .  .
```

Press any key to bypass the automatic autoboot sequence.

Remote Support Command

The following is a remote support command.

# - Comment Command

The comment command allows a user to write messages while at the `ROM >>` prompt without affecting system usage. This command would mainly be used by remote support engineers to convey messages to the console at a remote site.

Format

`#` *comments*

Examples

```
ROM >># This is the message<CR>
ROM >># and this is more of the message<CR>
```