Silicon Graphics Fuel™ Visual Workstation Diagnostic Reference Manual

SGI Confidential & Proprietary Information - For Internal Recipients Only

Document Number 108-0350-002

CONTRIBUTORS
Revised by Darrin Goss
Edited by Cindi Leiser
Production by Rhonda Kunsman
Engineering contributions by Mike Brown, Nelson Crumbaker, Tim Gamerdinger, Jason Godfrey, Jeff Keopp, Kurt Kermes, Art Ordonio, Darin Oster, Jon Palm, Nadia Pavlova, Roberto Romano, Joe Sam, Lisa Steinmetz, Eric Voisard, Jim Young, and Judy Young

Record of Revision

Version	Description
001	January 2002 Original printing.
002	July 2002 Revised for the IRIX 6.5.17 and *Internal Support Tools 2.7* CD releases. This revision adds information about the `olcdrom` online diagnostic.

Contents

Figures

Tables

About This Document

The *Silicon Graphics Fuel Visual Workstation Diagnostic Reference Manual* provides information about the diagnostic software that you can use to troubleshoot Silicon Graphics Fuel visual workstations.

This document includes the following information:

- Chapter 1, “Overview,” provides an introduction to the diagnostic software that is available.
- Chapter 2, “Scan Diagnostics,” describes the scan diagnostics and how to use them.
- Chapter 3, “Power-on Diagnostics,” describes the power-on diagnostics and how to use them.
- Chapter 4, “Offline Diagnostics,” describes the offline diagnostics and how to use them.
- Chapter 5, “Online Diagnostics,” describes the online diagnostics and how to use them.
- The Appendix describes POD mode and lists the commands that you can run from it.

This document uses the following notational conventions.

Convention	Meaning
`screen display`	Fixed-space font denotes literal items such as references to commands, files, routines, path names, output, messages, and programming language structures.
variable	Italic typeface denotes variable entries and words or concepts that are being defined such as command variables and references to document titles.
`user input`	Bold, fixed-space font denotes literal items that you enter on the keyboard in interactive sessions.
...	Ellipses indicate that one or more elements have been removed and / or that the preceding element can be repeated.
`[]`	Square brackets enclose optional portions of a command.
`<>`	Pointed brackets enclose required portions of a command.
|	Dividers are used to separate more than one possible value for commands and variables.

Chapter 1

Overview

SGI provides many diagnostics that you can use to troubleshoot Silicon Graphics Fuel visual workstations. These diagnostics test the components represented in Figure 1-1 and Figure 1-2.

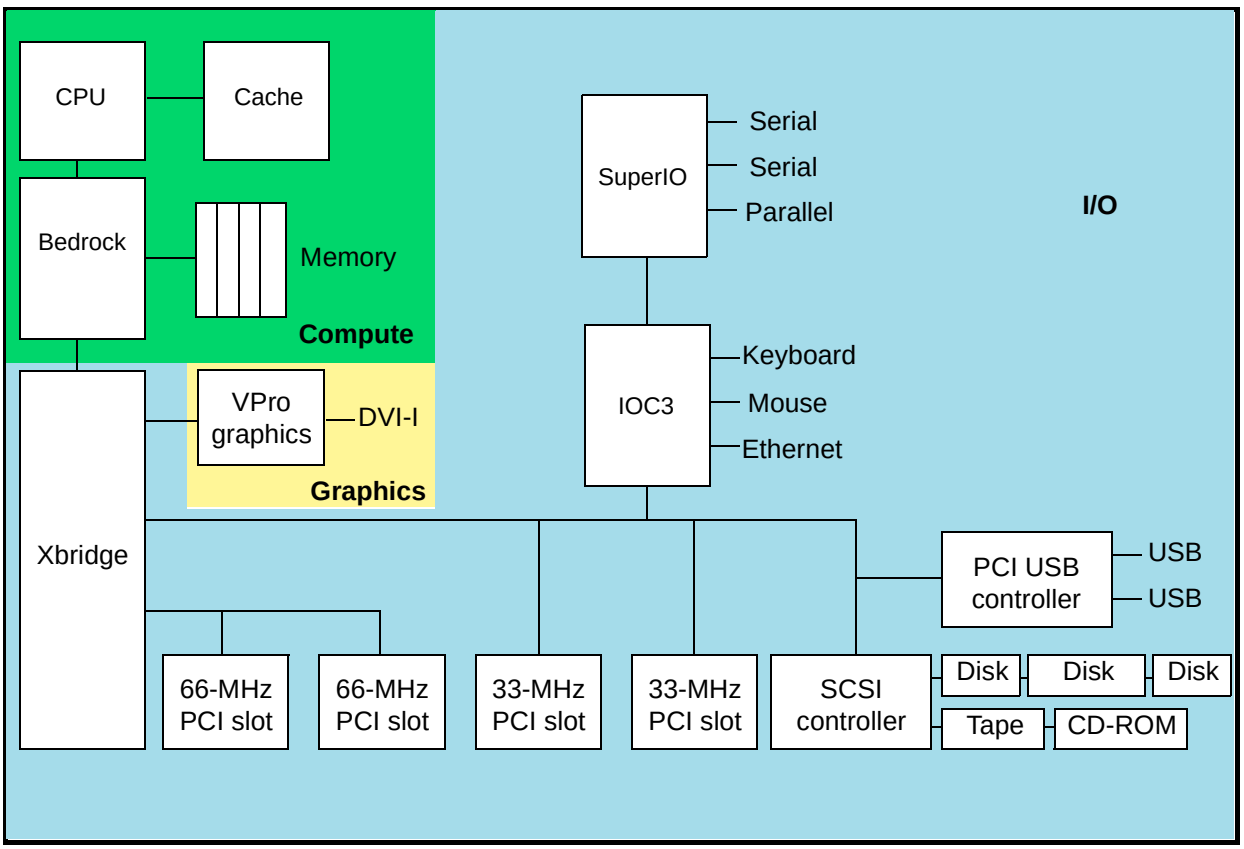

Figure 1-1 Hardware Components Tested by Diagnostics (Block Diagram View)

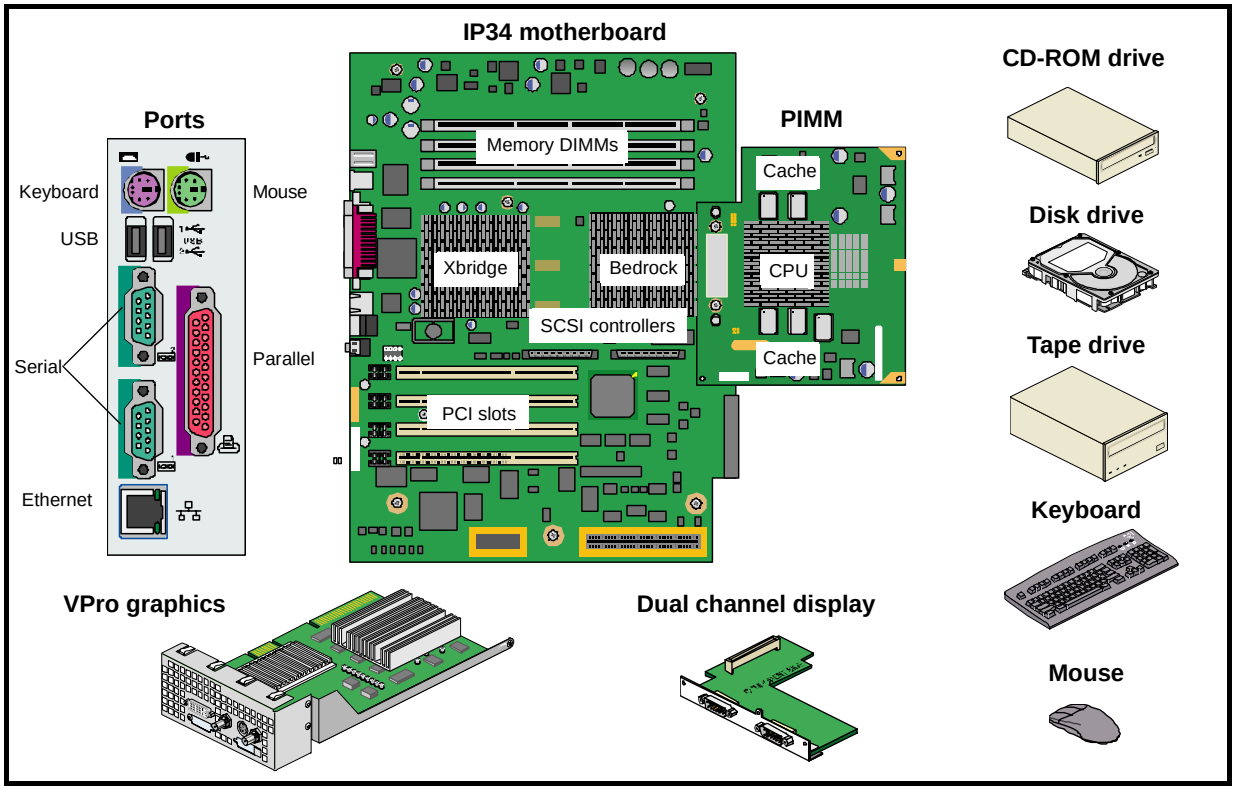

Figure 1-2 Hardware Components Tested by Diagnostics

The diagnostics are grouped into four categories:

- Scan diagnostics
- Power-on diagnostics
- Offline diagnostics
- Online diagnostics

Table 1-1 provides information about the hardware that the diagnostics test.

Table 1-1 Hardware Tested by Diagnostics

Functional Area	Component	Tested by:[a]				Hardware Detected	Part Number
		Scan	POD	Offline	Online		
Compute	CPU	x	x	x	x	PCA PIMM R14KA 500MHz w/2MB + heatsink	013-3415-00x
	Cache	x	x	x	x	PCA PIMM R14KA 600MHz w/4MB + heatsink	013-3512-00x
	Bedrock ASIC	x	x	x	x	PCA motherboard IP34	013-3414-00x
	Memory DIMMs	x	x	x	x	PCA 256MB DIMM 128MB DDR	030-1018-00x
						PCA 512MB DIMM 128MB DDR	030-1042-00x
						PCA 1GB DIMM 128MB DDR	030-1060-00x
I/O	XBridge ASIC	x	x	x	x	PCA motherboard IP34	013-3414-00x
	PCI slots	x	x	x	x		
	USB ports		x	x	x		
	Serial ports		x	x	x		
	Parallel port			x			
	Ethernet port		x	x	x		
	SCSI controller	x	x	x	x	PCA motherboard IP34	013-3414-00x
						Cable assy SCSI 68P w/term	018-1005-00x
	Disk		x		x	HDD 18GB	064-0232-00x
						HDD 18GB 10K RPM 68 pin	064-0169-00x
						HDD 36GB 10K RPM 68 pin	064-0233-00x
						HDD 73GB 10K RPM 68 pin	064-0218-00x
						Cable assy SCSI 68P w/term	018-1005-00x
	Tape drive				x	Drive TP 20GB DAT DDS4 5.25 inch	064-0150-00x
						Assy DAT DDS4 external tape	013-3168-00x
	CD-ROM drive		x		x	Drive DVDROM 10X internal	064-0194-00x
						Cable assy SCSI 50P flat, M/B to DVDROM	018-1006-00x

Table 1-1 (continued) Hardware Tested by Diagnostics

Functional Area	Component	Tested by:[a]				Hardware Detected	Part Number
		Scan	POD	Offline	Online		
I/O (cont.)	Keyboard		x	x		Keyboard PS/2 101-key, US ENGLISH	062-0046-00x
						Keyboard PS/2 102-key, GERMAN	062-0047-00x
						Keyboard PS/2 102-key, FRENCH	062-0048-00x
						Keyboard PS/2 102-key, ITALIAN	062-0049-00x
						Keyboard PS/2 102-key, SPANISH	062-0050-00x
						Keyboard PS/2 102-key, UK ENGLISH	062-0051-00x
						Keyboard PS/2 102-key, NORWEGIAN	062-0052-00x
						Keyboard PS/2 102-key, SWED/FINNISH	062-0053-00x
						Keyboard PS/2 106-key, JAPANESE	062-0054-00x
						Keyboard PS/2 101-key, KOREAN	062-0055-00x
						Keyboard PS/2 102-key, SWISS	062-0067-00x
	Mouse		x	x		Mouse 3-button opto-mech	063-0009-00x
Graphics	VPro graphics	x	x		x	PCA Odyssey VPro GFx, 32 MB, V10	013-3513-00x
						PCA Odyssey VPro GFx, 128 MB, V12	013-3416-00x
						PCA Odyssey X2-brick dual channel	030-1713-00x

a. An “x” in the “Scan” column indicates that scan diagnostics are available to test the component; an “x” in the “POD” column indicates that power-on diagnostics are available to test the component; an “x” in the “Offline” column indicates that offline diagnostics are available to test the component; and an “x” in the “Online” column indicates that online diagnostics are available to test the component.

1.1 Scan Diagnostics

The scan diagnostics use boundary scan features in the hardware to test connections on boards and between boards. The scan diagnostics include the following software:

- A boundary scan interconnect test (`scantest`) that manipulates the boundary scan registers in various chips throughout the system to verify system interconnections. It applies a set of test vectors to the system to detect physical connection defects (for example, stuck-at faults and shorts).
- A scan-based link-level protocol interconnect test (`slit`) that uses additional scan-accessible registers to exercise connections at-speed.

  Note: The scan-based link-level protocol test runs only in components that use LLP (the Bedrock and Xbridge ASICs).
- A script (`scan_asterix`) that enables you to run the boundary scan interconnect test and scan-based link-level protocol test.
- A scripting tool (`scantool`) that enables you to control scan operations from Tcl/Tk script.

The scan diagnostics are part of the L3 system controller software that is included on the *Internal Support Tools 2.7* CD, SGI part number 812-0640-011. To run the scan diagnostics, you must have an external system running the L3 system controller software connected to the workstation or a remote support connection to a workstation.

Note: Silicon Graphics Fuel visual workstations can use the same L3 system controller software as SGI 3000 series systems. Refer to *SGI 3000 Family System Controllers*, publication number 108-0241-00x, for more information about the L3 system controller software and the systems that can run it.

1.2 Power-on Diagnostics

Power-on diagnostics are PROM-resident tests that run automatically when you power on the system. As the boot process discovers hardware components, it runs power-on diagnostics to verify that each component is functional enough to load the operating system. You can also run several of the diagnostics manually to test specific components.

Table 1-2 lists the power-on diagnostics that are available.

Table 1-2 Power-on Diagnostics

Name	Description
n/a	Register test
cache_test_i	Primary instruction cache test
cache_test_d	Primary data cache test
cache_test_s	Secondary cache test
hub_intrpt_diag	Hub interrupt test
n/a	Hub local test
n/a	Hub config test
n/a	Hub UART test
hubsde[a]	Hub send-data test
test_hub_error_detection[a]	Hub error-detection test
rtrsde[a]	Router send-data test
test_rtr_error_detection[a]	Router error-detection test
get_chipid[a]	Get Chipid test
rt_clock_test	GCLk logic test
hub_bte_diag	Block transfer engine test
prom_checksum	PROM checksum test
n/a	Memory configuration test
dirtest	Backdoor directory space test
memtest	Memory test
chklink[a]	NUMAlink test
xbow_sanity	XBOW test
bridge_sanity	Bridge test
io7config_space	PCICONFIG test
pcibus_sanity	PCI bus test
serial_pio	Serial programmed I/O test

Table 1-2 (continued) Power-on Diagnostics

Name	Description
serial_dma	Serial DMA test
scsi_ram	SCSI controller SSRAM test
scsi_dma	SCSI controller DMA test
scsi_controller	SCSI controller self-test
pckm	Keyboard and mouse test
gfx	Graphics test
enet_all	Comprehensive Ethernet test
enet_ssram	Ethernet SSRAM test
enet_phy_reg	PHY register test
enet_tx_clk	TX clock test
enet_ioc3_loop	IOC3 internal loopback test
enet_phy_loop	PHY chip internal loopback test
enet_tw_loop	Twister chip internal loopback test
enet_ext_loop	External loopback DMA test
enet_10_ext_loop	External loopback DMA (10 Mb/s) test
enet_100_ext_loop	External loopback DMA (100 Mb/s) test
enet_xtalk_stress	Xtalk stress test

a. These IP35 router and hub tests are disabled because they do not apply to the IP34 hardware; these tests are not documented in this manual.

Refer to Chapter 3, “Power-on Diagnostics,”for more information about the power-on diagnostics and how to run them.

1.3 Offline Diagnostics

Offline diagnostics are tests that run under the scalable micro-diagnostic kernel (sMDK). sMDK is a stand-alone diagnostic environment that runs in kernel mode. sMDK provides test access to hardware components that cannot be accessed from a user program that is running under the operating system.

Offline diagnostics require full use of the system: you must bring down (reboot) the system to run them. No operating system can be running in the system when you use the offline diagnostics.

The offline diagnostics include a "launcher" that automatically runs a sequence of tests. In most cases, you should run the offline diagnostics automatically with the launcher. You can also run the offline diagnostics manually to test a specific component. (When you manually run the offline diagnostics, you must reboot the system when you finish testing it.)

Table 1-3 lists the offline diagnostics that are available.

Table 1-3 Offline Diagnostics

Name	Description
`CPU_Test`	CPU test
`sct`	Secondary cache test
`scct`	Cache coherency test
`nmem`	Memory test
`ndir`	Directory memory test
`io8bt`	I/O test (checks the following hardware on the IP34 motherboard: IOC3 ASIC, USB ports, SCSI controller components, PCI slots, parallel port, keyboard port, and mouse port)
`xbg`	Xbridge ASIC and PCI slot test

Refer to Chapter 4, "Offline Diagnostics," for more information about the offline diagnostics and how to run them.

1.4 Online Diagnostics

Online diagnostics are tests that verify system hardware while the operating system is running. When you run an online diagnostic from the IRIX operating system prompt, the diagnostic runs a set of tests for a certain number of loops. Each online diagnostic has one or more *standard* tests that run by default if you do not specify a test in the command line. You may need to specifically request additional tests that you need to run.

The online diagnostics include a `runalldiags` script that automatically runs a sequence of tests. In most cases, you should run the online diagnostics automatically with the `runalldiags` script. You can also run the online diagnostics manually to test a specific component.

Table 1-4 lists the online diagnostics that are available.

Table 1-4 Online Diagnostics

Name	Description
`bridgeloc`	PCI bridge locator and diagnostic listing tool
`odydiags`	Graphics diagnostics
`olcdrom`	CD-ROM diagnostic
`oldisk`	Disk diagnostic
`olenet`	Ethernet diagnostic
`olmem`	Memory diagnostic
`olpci`	PCI configuration viewer and diagnostic listing tool
`olperi`	CPU diagnostic
`olsio`	SIO serial diagnostic
`oltape`	Tape diagnostic
`olusb`	USB diagnostic
`olvst`	Socket-based network diagnostic
`pandora`	System stress test
`runalldiags`	Script to automatically run a sequence of tests

Refer to Chapter 5, “Online Diagnostics,” for more information about the online diagnostics and how to run them.

1.5 Location of Diagnostic Output

The diagnostics display output via the physical LEDs, the virtual LEDs, messages on an external system that is running the L3 system controller software, and messages on the workstation monitor. (Refer to Figure 1-3.)

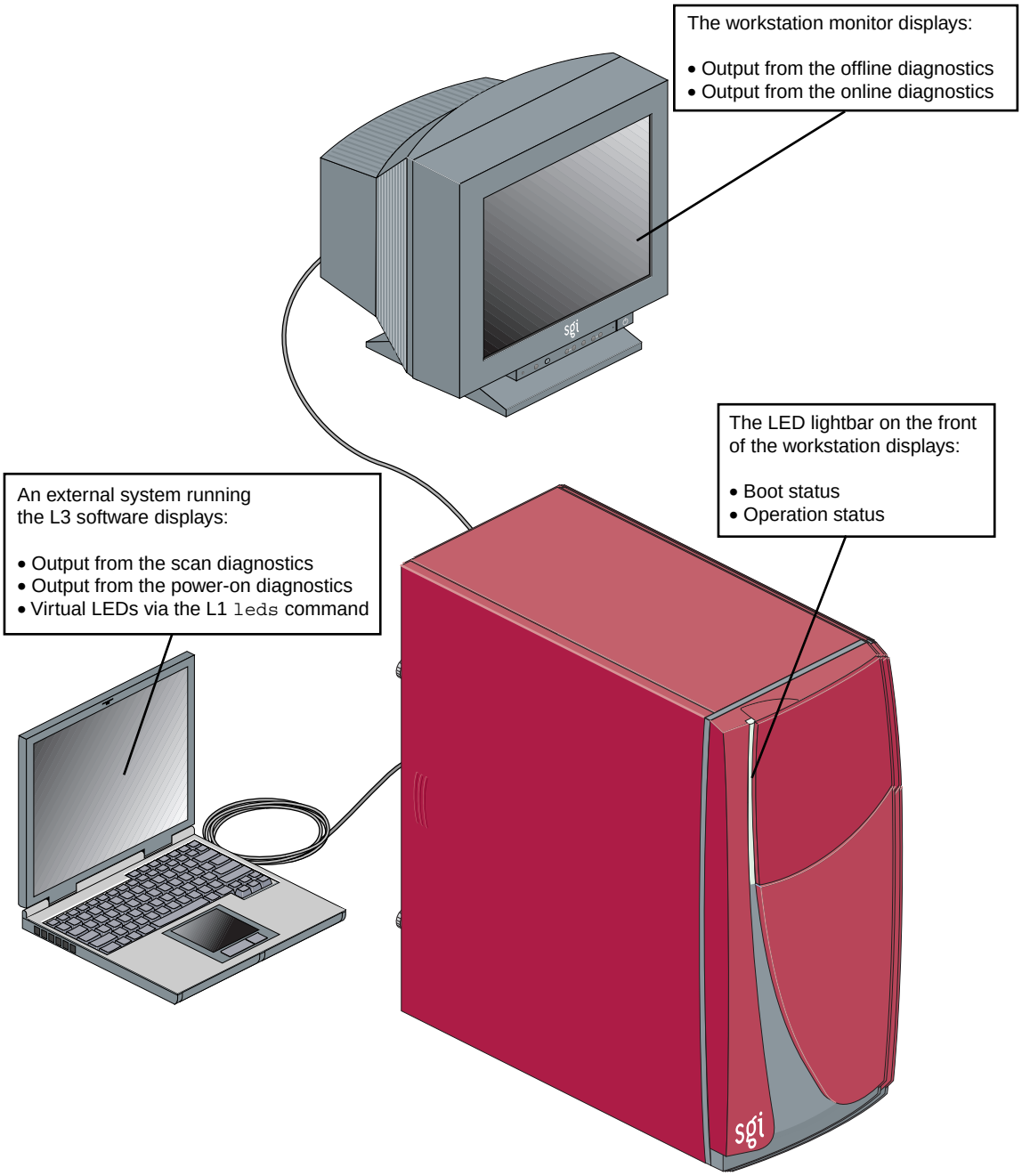

Figure 1-3 Location of Diagnostic Output

1.6 Before You Replace Components

Always try the following actions before you replace any hardware components in a system:

- If you suspect a keyboard or mouse failure, switch the keyboard and mouse PS/2 port connections to determine whether a port is failing or a device is failing.
- Disconnect and reconnect any cables or devices (for example, the keyboard) that might affect the failing hardware.
- Remove and reinstall any memory DIMMs that you suspect are failing.
- Remove and reinstall any PCI boards that might affect the failing hardware.
- Reset the system, and run the diagnostics again.

Chapter 2

Scan Diagnostics

The Institute of Electrical and Electronics Engineers (IEEE) standard 1149.1, *Standard Test Access Port and Boundary-Scan Architecture*, defines a set of test logic and related features that enable connections within and between ASICs to be tested. These features are collectively called boundary scan, scan, or JTAG.

The test logic includes a test access port (TAP), a TAP controller, an instruction register, and a group of test data registers. (Refer to Figure 2-1.)

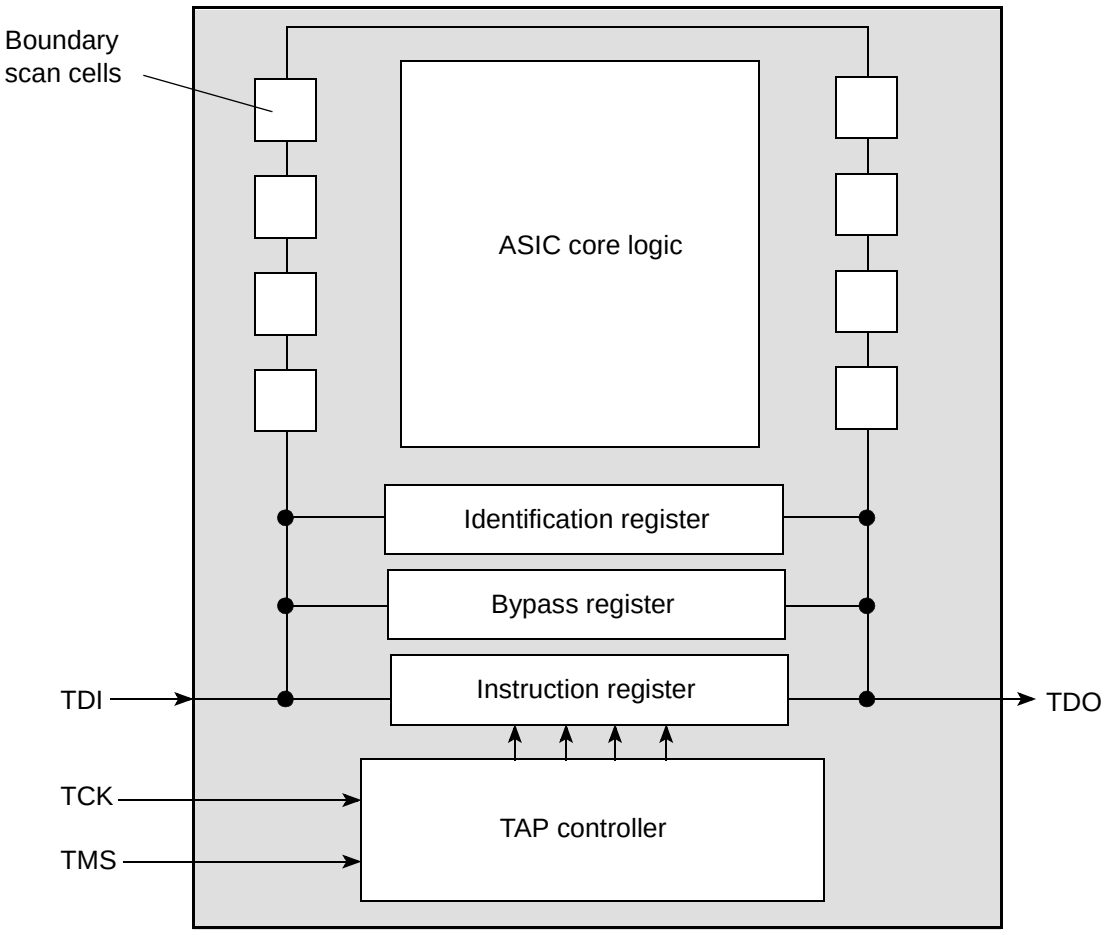

Figure 2-1 Boundary Scan Test Logic

Table 2-1 describes the boundary scan components.

Table 2-1 Boundary Scan Components

Component	Description
Test access port (TAP)	Provides access to the test logic and consists of four signal pins: test data input (TDI), test data output (TDO), test clock (TCK), and test mode select (TMS)
TAP controller	Controls the instruction register, bypass register, identification register, and boundary scan register
TMS signal	Controls test operations
TCK signal	Inputs a signal that clocks the TAP controller
TDI signal	Inputs data serially
TDO signal	Outputs data serially
Instruction register	Receives, holds, and decodes boundary scan instructions
Bypass register	Connects the TDI and TDO signals during a BYPASS instruction, which causes the data to go through the bypass register instead of the boundary scan cells (This register is used to verify integrity of the scan chain and to perform board-level testing.)
Identification (ID) register	Holds 32 bits of information that specify the manufacturer of the chip (or ASIC), the part number of the chip, and the revision of the chip ID registers are optional components; not all chips contain ID registers
Boundary scan cells	Components of the boundary scan register that are connected in a shift-register path around the boundary of the chip; these components are located between the I/O pins and the core logic, which enables the boundary scan operation to control and monitor the I/O pins
ASIC core logic	Contains the functional components of the ASIC

The following components in Silicon Graphics Fuel visual workstations include the scan/JTAG test logic as defined by the IEEE 1149.1 standard:

- Bedrock ASIC
- Xbridge ASIC
- Memory DIMM slots (with special cards)
- PCI slots (with special cards)
- VPro Odyssey graphics card
- IOC3 ASIC
- SCSI connectors
- PIMM

Several scan diagnostics are available to test connections within boards and between boards.

2.1 Scan Hardware

The following hardware supports scan testing on Silicon Graphics Fuel visual workstations (refer to Figure 2-2):

- An external system running the L3 system controller software uses a USB connection to communicate with the L1 system controller.
- The L1 system controller acts as a JTAG bus master and passes JTAG commands / data to the scan interface chip (SIC).
- The SIC passes the JTAG commands / data to the individual scan chains.
- The scan chains pass the commands / data to the individual ASICs. A Silicon Graphics Fuel visual workstation has the following four scan chains:
  - Scan chain 0 includes the PIMM.
  - Scan chain 1 includes the Bedrock and Xbridge ASICs.
  - Scan chain 2 includes the VPro graphics card, the SCSI host adapters, PCI bus E slot 2, PCI bus E slot 1, PCI bus F slot 2, PCI bus F slot 1, and the IOC3 ASIC.

    Note: Scan testing bypasses the graphics slot if a VPro graphics card is not installed. Scan testing bypasses the PCI slots if scannable PCI test boards (SPTBs) are not installed. SPTBs are not available for field use.
  - Scan chain 3 includes the DIMMs.

    Note: Scan testing bypasses the DIMM slots if scannable DIMM test boards (SDTBs) are not installed. SDTBs are not available for field use.

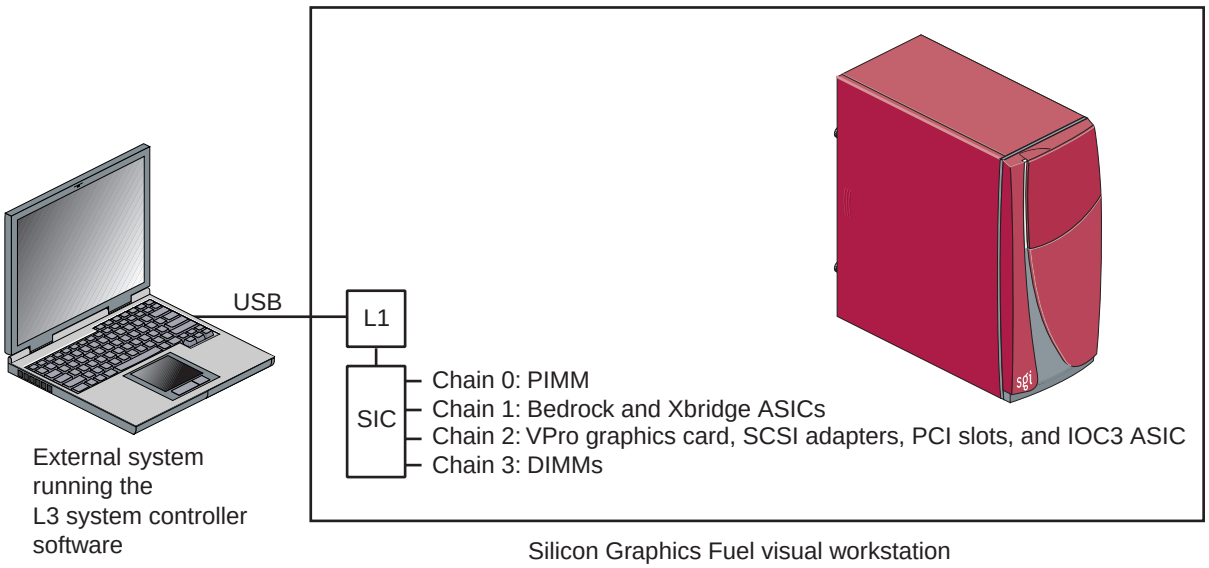

Figure 2-2 Scan Hardware Components

2.2 Scan Software

The scan diagnostics include the following software:

- A boundary scan interconnect test (`scantest`) that manipulates the boundary scan registers in various chips throughout the system to verify system interconnections. It applies a set of test vectors to the system to detect physical connection defects (for example, stuck-at faults and shorts).
- A scan-based link-level protocol interconnect test (`slit`) that uses additional scan-accessible registers to exercise connections at-speed.

  Note: The scan-based link-level protocol test runs only in components that use LLP (the Bedrock and Xbridge ASICs).
- A script (`scan_asterix`) that runs the scan the boundary scan interconnect test and scan-based link-level protocol test.
- A scripting tool (`scantool`) that enables you to control scan operations from Tcl/Tk script.

The scan diagnostics reside on an external system running the L3 system controller software. To use the scan diagnostics, you must have an L3 system controller, a laptop loaded with the L3 system controller software, or a remote support connection to a system.

Note: The scan diagnostics are the same as the SGI 3000 family scan diagnostics. For more detailed information about the scan diagnostics, refer to *Scan Tools*, publication number 108-0263-00x.

2.3 Distribution

The *Internal Support Tools 2.7* CD, SGI part number 812-0640-011, contains the L3 system controller software, which includes the scan diagnostics. The scan diagnostics are SGI proprietary and confidential and are available only for SGI personnel. You need to install the L3 system controller software on an external system to run the scan diagnostics. (The CD booklet includes installation procedures.)

2.4 Boundary Scan Interconnect Test

The boundary scan interconnect test (`scantest`) manipulates the boundary scan registers in various chips to verify connections within the system. `scantest` applies a set of test vectors to detect physical connection defects (stuck-at faults and shorts). Use it to verify proper operation of components that you add to a system.

2.4.1 Test Algorithm

The boundary scan interconnect test uses the following test algorithm:

1. Load the chain configuration and reference data.
2. Load the reference data for each board definition.
3. Initialize JTAG communications to the selected scan controller.
4. Initialize the chain TAPs and configure the SIC.
5. Perform comprehensive scan hardware integrity tests to verify that the scan hardware is functional enough to perform boundary scan interconnect testing:
   - Perform the instruction register length test.
   - Perform the instruction register path test.
   - Perform the bypass register length test.
   - Perform the bypass register path test.
   - Perform the chain-level boundary register length test.
   - Perform the chain-level boundary register path test.
6. Perform the board-level boundary scan register interconnect test.
7. Print a program termination status message.

Caution: Each step must pass before the next step can be run. If the integrity tests in Step 5 do not pass, the interconnect test does not run because the results would be invalid. Do not manually run the interconnect test until the integrity tests pass.

2.4.2 Running the Boundary Scan Interconnect Test

Three ways to run the boundary scan interconnect test exist:

- Use the `scan_asterix` script.
- Use the `brick_scan` graphical user interface.
- Use the `scantest` command.

Warning: **Ensure that the system is offline before you run the boundary scan interconnect test. If the operating system is running when you run this test, the operating system will crash and customer data will be lost.**

2.4.2.1 Using the scan_asterix Script

The easiest way to run the boundary scan interconnect test is to use the `scan_asterix` script to automatically run the test. (Refer to “scan_asterix” on page 2-23.)

2.4.2.2 Using the brick_scan Graphical User Interface

The `brick_scan` graphical user interface enables you to set test parameters and view test output. (Refer to Figure 2-3.)

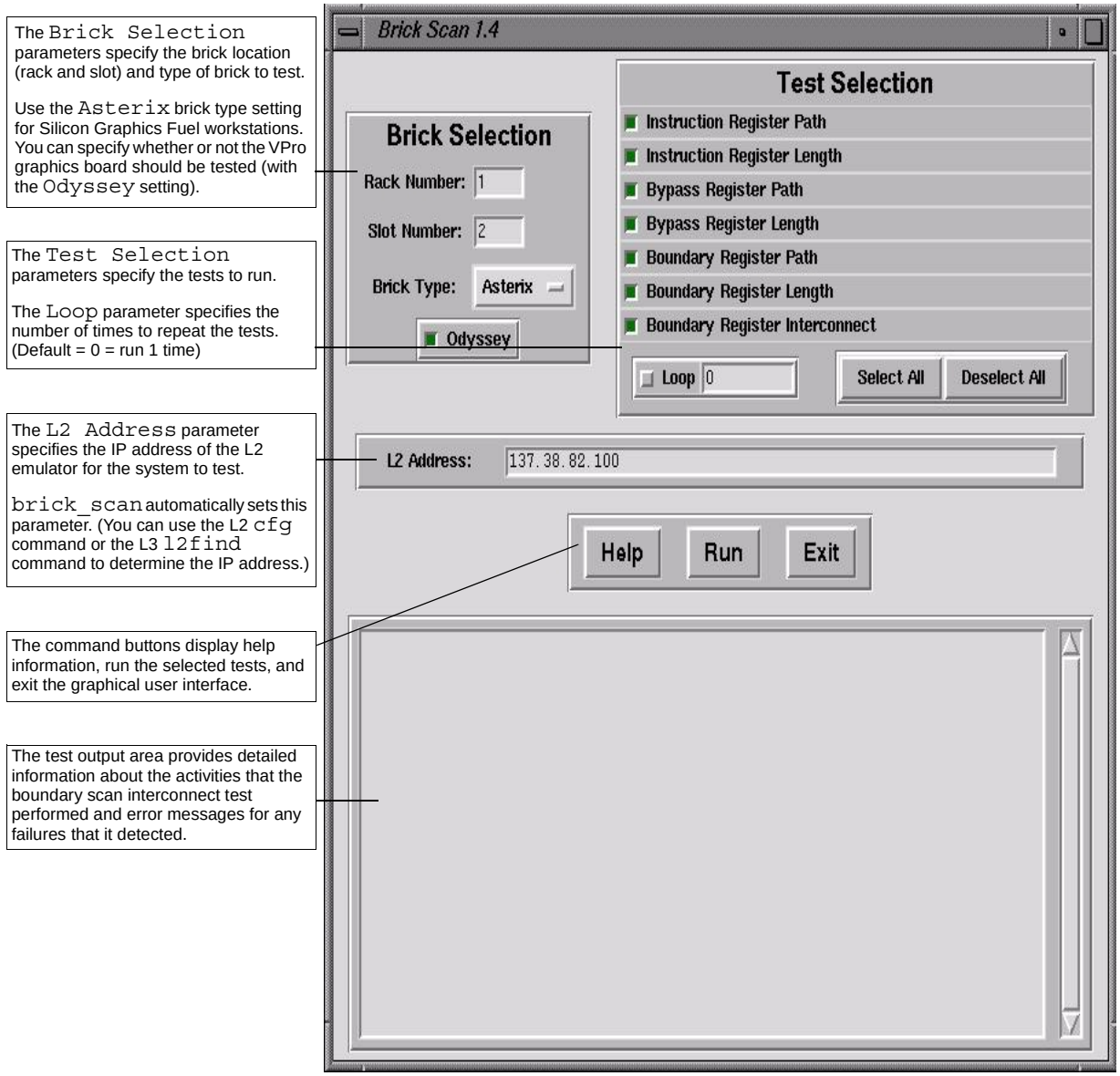

Figure 2-3 Boundary Scan Test Graphical User Interface

Perform the following procedure to run the boundary scan interconnect test from the `brick_scan` graphical user interface:

1. Start the L2 system controller emulator, and power up the system.
2. Log into the L3 system controller as `sgidiag` (default password: `sgi!diag`).
3. Enter the following command to change to `/stand/sysco/usr` directory:

   `cd /stand/sysco/usr`

4. Enter the following command to start the L2 emulator:

   `/stand/sysco/bin/l2`

5. Enter the following command to start the `brick_scan` application:

   `/stand/sysco/bin/brick_scan`

   The `brick_scan` application displays a warning message. (Refer to Figure 2-4.) Review this information and ensure that the system is ready for testing; then, click `Continue`.

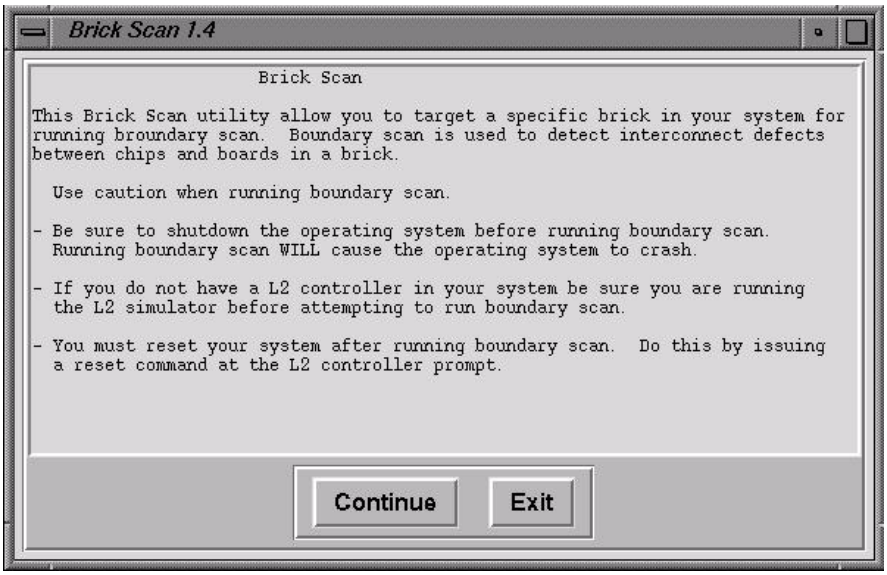

Figure 2-4 Boundary Scan Interconnect Test Warning Message

6. Select the brick that you want to test (refer to Figure 2-5):

   - Enter the rack number in the `Rack Number` field. (Normally, you should set this parameter to 1.)
   - Enter the slot number in the `Slot Number` field. (Normally, you should set this parameter to 1.)

     If you do not know the rack and slot locations of the brick that you want to test, use the L2 `cfg` command to determine the bricks that are available.

   Note: If you connect more than one Silicon Graphics Fuel visual workstation to an external system that is running the L3 system controller software, you must assign unique rack and slot combinations to each workstation.

   - Select `Asterix` type from the `Brick Type` menu.

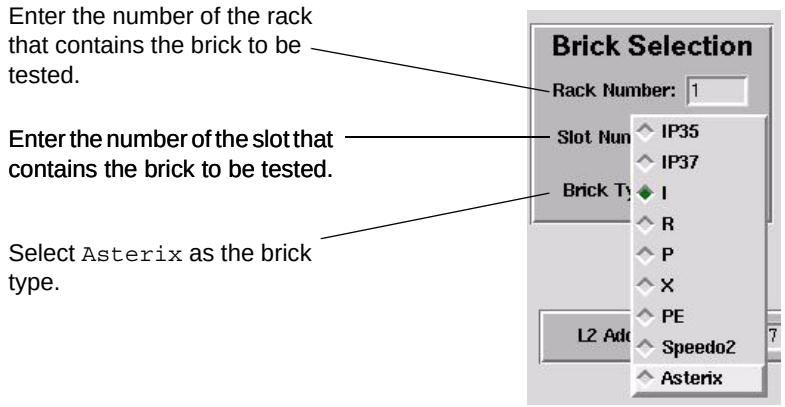

Figure 2-5 Setting the Brick Selection Parameters

7. Select the test sections that you want to run. (Refer to Figure 2-6.) SGI recommends that you run all test sections.

Warning: **Each test section must pass before the next section can be run. If the integrity tests do not pass, the interconnect test does not run because the results would be invalid. Do not manually run the interconnect test until the integrity tests pass.**

If you wish to run the test multiple times, change the `Loop` parameter, and click on the box next to the `Loop` parameter. The test defaults to one pass.

Note: If you set the `Loop` parameter but do not click the box next to it, the test performs only one pass. If you want to run multiple passes, you must set the `Loop` parameter and click on the box next to it.

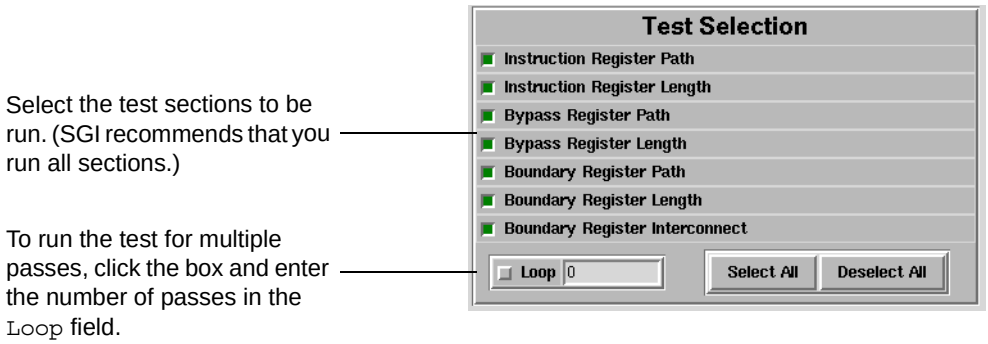

Figure 2-6 Setting the Test Selection Parameters

8. Modify the `L2 Address` parameter, if necessary. (Refer to Figure 2-7.)

   The `L2 Address` parameter specifies the IP address of the L2 emulator for the system to be tested.

   Note: The `brick_scan` application defaults to the first L2 system controller address that it finds. If the `L2 address` field does not contain the IP address of the L2 emulator that you want to use, use the L3 `l2find` command or L2 `cfg` command to determine the correct IP address.

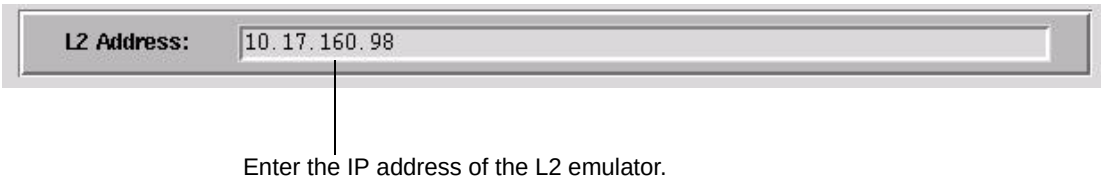

Figure 2-7 Setting the Scan Communications Method Parameters

9. Click on the `Run` button. (Refer to Figure 2-8.)

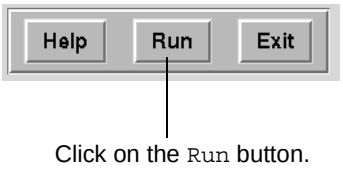

Figure 2-8 Run Button

10. Interpret the output. (Refer to Section 2.4.3, “Error Output.”)

    Note: The boundary scan interconnect test saves the test output in the `scantest.dmp` file in the current directory; however, it overwrites this file each time you run the test. If you want to save the output so you can analyze it later, copy the contents of the file to a new file before you run the boundary scan interconnect test again.

11. When you are done testing, enter the L2 `reset` command to reset the system.

2.4.2.3 Using the scantest Command

You can manually run the test with the `scantest` command. This command uses the following syntax:

```
scantest [-h] <-f file> [-ID num] [-d file] [-e num] [-p num] [-t test] [-m mode]
[-l level] [-s mode] [-v] [--l2 <host> | --interface <iface> |
--ssn <serial>[:<rack>] | --sysname <system_name>[:<rack>] | --scdev <device> |
--serial <host>:<port> | --dev <device_name>]
```

Table 2-2 describes the command-line options that you can use to control `scantest`.

Table 2-2 scantest Command-line Options

Option	Description
`-h`	Displays the available command-line options and then exits.
`-f <file>`	Specifies a configuration file to use. The configuration file defines the hardware to test. (Default: `slit.cfg`) The following configuration files are available in `/stand/sysco/cfg/scan` to test Silicon Graphics Fuel visual workstations: `ip34_002_ast.cfg` (nets between Xbridge ASIC and graphics board `ip34_002_po.cfg` (nets between processor and cache) `ip34_002_np.cfg` (nets between PIMM, Xbridge ASIC, and Bedrock ASIC) The `ID` field in each configuration file describes the brick to test. By default, the `ID` field is set to zero (0), which selects the local brick. Use the following formula to select a different brick: ID = (rack * 64) + slot If you want to modify a configuration file, copy it to a directory for which you have write permission. If you modify the ID field, you must set the first number of the chain definition to the same value (for example, `START CHAIN 138`).
`-ID <num>`	Overrides the `ID` field listed in a configuration file. This enables you select a different brick for testing without modifying the configuration file. Use the following formula to select a different brick: ID = (rack * 64) + slot
`-d <file>`	Specifies the output file that receives output from the test. (Default: `scantest.dmp`)
`-e <num>`	Defines the maximum number of errors that the test displays. (Valid values are between 0 [display only pass/fail information] and 10,000; default: 200)
`-p <num>`	Specifies the number of test passes that `scantest` should perform. (Valid values are between 1 and 10,000; default: 1)

Table 2-2 (continued) scantest Command-line Options

Option	Description
`-t <test>`	Specifies the tests to run. Valid tests: irp = instruction register path test irl = instruction register length test ir = instruction register path test and instruction register length test prp = bypass register path test prl = bypass register length test pr = bypass register path test and bypass length test brp = boundary register path test brl = boundary register length test br = boundary register path test and boundary register length test int = boundary register interconnect test id = read and display device ID registers all = all tests Use + to select multiple tests (for example: ir+br+int). The order in which the tests are specified is irrelevant. Certain tests depend on other tests completing successfully, so selecting one test may cause other tests to be automatically selected. (Default: ir+pr+br+int)
`-m <mode>`	Specifies the mode for register tests. Valid modes: normal = normal lengths and minimal pattern(s) elen = extended lengths epat = extended pattern set If normal is specified, then no other option is valid. (Default: normal)
`-l <level>`	Specifies the level of the register tests. Valid modes: chn = chain level register test brd = board level register test chip = chip level register test (Default: chn)
`-s <mode>`	Selects a stepping mode. Valid modes: no = no test stepping; run continuous all = step through each pattern; prompt user err = step through failing pattern; prompt user (Default: no)
`-v`	Selects verbose mode, which writes additional messages to the output.
`--l2 <host>`	Connects to the L2 system controller with the specified hostname or IP address

Table 2-2 (continued) scantest Command-line Options

Option	Description
`--interface` <*iface*>	Connects to the L2 system controller with the specified network interface (for example, eth0)
`--ssn` <*serial*>`[:`<*rack*>`]`	Connects to the system with the specified serial number (for example, L01234567)
`--sysname` <*system_name*> `[:`<*rack*>`]`	Connects to the system specified by <*system_name*>
`--scdev` <*device*>	Connects to the specified system controller
`--serial` <*host*>`:`<*port*>	Connects to the specified Ethernet hostname and port
`--dev` <*device_name*>	Connects to the specified local serial port device

2.4.3 Error Output

The boundary scan interconnect test returns the following error messages:

```
ERROR - Unable to open output file for writing.
      File: scantest.dmp
      Writing output data to screen instead of file
```

This message appears when you do not have permission to create the `scantest.dmp` output file. The boundary scan interconnect test continues to run, but it does not send output to a file. To save test output to a file, run the test from a directory for which you have write permission.

```
Unable to run interconnect test because of register test error(s).
```

This message appears when one of the boundary scan validation tests fails. When this happens, the interconnect test does not run. Review the output that precedes this message to determine the failing register:

- If the failing register is an instruction register, verify that the brick is powered up. If the brick is powered up, there is a failure in the boundary scan chain. If the brick is not powered up, power it up and rerun the test.
- If the failing register is not an instruction register, there is a failure in the boundary scan chain.

```
ERROR - scantest V3.11 (snl3sc.c @ 280, code=0)
  Unable to read scan response packet from controller.
  Unknown Module 0, message = -1
  Confirm configuration of brick specified with ID=483 (rack=7, slot=35)
```

This message typically appears when you specify incorrect rack and slot numbers for the brick to test. Verify the rack and slot number of the brick that you want to test and rerun the test.

This message also appears if the brick is completely powered off (the L1 system controller is not running). Power on the brick and rerun the test.

```
Errors were detected during interconnect test.
```

This message appears when the boundary scan interconnect test detects a failing component in the brick. Review the output following this message to determine the failing component.

After displaying this message, the test continues. When testing completes, the `scantest` generates a list of all chain positions that had failing data and identifies the failing chips and pins by physical and logical locations. The detailed failure information uses the following format:

```
BOUNDARY REGISTER INTERCONNECT FAILURE
--------------------------------------

Chain and raw position  : 000-000601
Expected data           : 0110011010010101
Actual data             : 0000000000000100
Difference data         :   ^^  ^^ ^  ^   ^
Driving node index      : 1111111111110011
Logical description     : [001a01:NODE:XBRG:PF_PCI_SERR_N]
001a01:NODE:IOC3:PCI_SERR_L
Physical description    : [001a01:NODE:G0J7:AD21] 001a01:NODE:F8B5:N2
```

The failure information contains the following data:

- `Chain and raw position`: indicate the location where the test detected a miscompare. The format of this value is <chain number>-<bit position>. (For example, `000-000601` indicates that the test detected the miscompare on chain number 0 at bit position 601.)
- `Expected data`: indicates the signature that the test expects to receive for the data patterns that it is using.
- `Actual data`: indicates the actual signature that was sensed for the data patterns.
- `Difference data`: contains a ^ symbol for each pattern in the signature where the actual data did not match the expected data.

Note: The number of data patterns that the test uses depends on the number of nets being tested. Two types of patterns exist: patterns to detect *stuck-at* failures and patterns to detect *short* failures. For the `Expected data`, `Actual data`, and `Difference data` values, each 0/1 value represents a pattern, with the first pattern (pattern 0) starting at the right and the pattern numbers incrementing by 1 as they go to the left (refer to Figure 2-9).

```
Expected data          : 0110011010010101
Actual data            : 0000000000000100
Difference data        :   ^^  ^^ ^  ^   ^
                                     | | |
                                     | | pattern 0
                                     | pattern 2
                                    pattern 4
```

Figure 2-9 Data Pattern Indicator

- `Driving node index`: indicates which node on the net is *driving* the transfer for each test pattern. (The driving node sends the data pattern to the receiving node.)

  The node index value is a single digit that specifies which node is the driving node for the data pattern. The digit starts at 0 for the first node on the left side of the `Logical description` and `Physical description` lists and then increments by 1 for each successive node. The node index value displays an `x` if a net is not valid for a pattern (for example, no driver is available).

  A net may have more than one driver if bidirectional signals are available on the net. If possible, *stuck-at* patterns are transferred from all potential drivers; *short* patterns are applied using a single driver.

- `Logical description`: provides a logical description of the nodes on the failing net. The format of each node is <chain and brick>:<board>:<chip>:<pin>. (Refer to `Figure 2-10`.) The boundary scan interconnect test displays the node that detected the miscompare in square brackets `[ ]`.

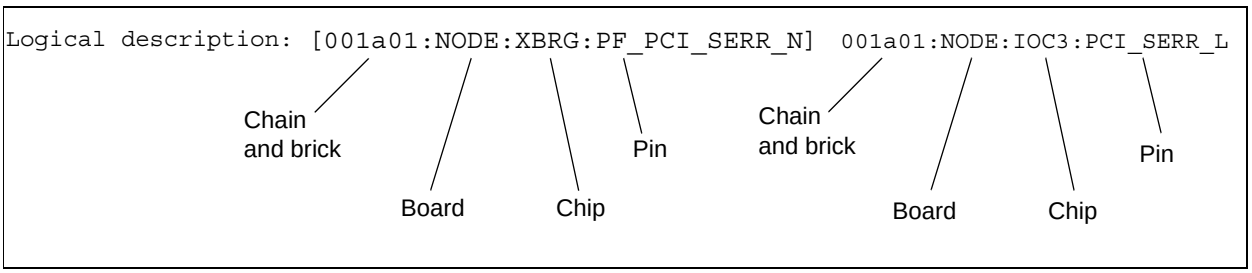

Figure 2-10 Logical Description Format

- `Physical description`: provides a physical description of the nodes on the failing net. The format of each node is <chain and brick>:<board>:<chip>:<pin>. `scantest` displays the node that detected the miscompare in square brackets `[ ]`.

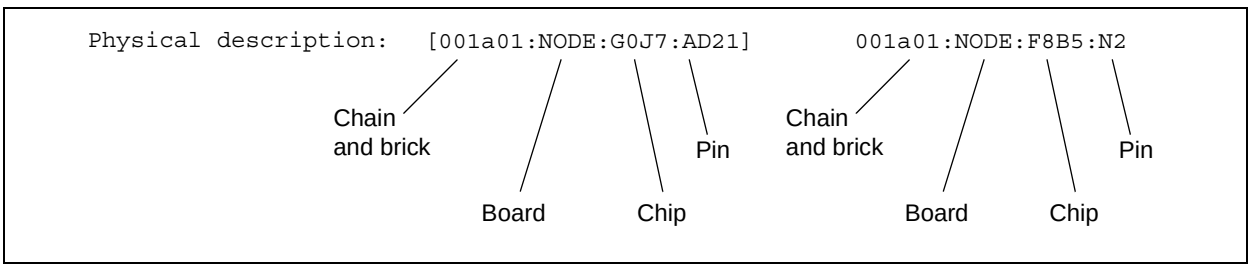

Figure 2-11 Physical Description Format

2.4.4 Determining the Hardware to Replace

The board information in the physical and logical descriptions indicates the FRUs used in the failing connection. (Refer to Figure 2-12.)

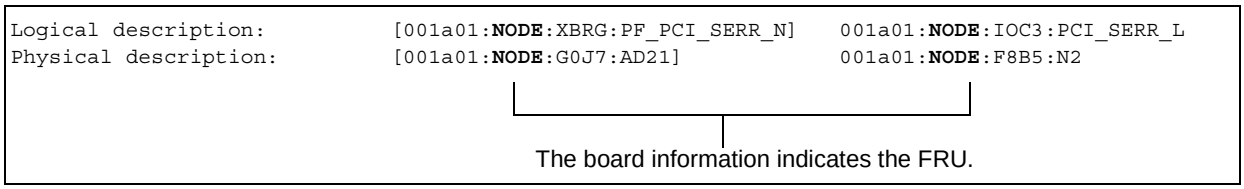

Figure 2-12 Determining the Hardware to Replace (scantest)

Table 2-3 lists the FRUs that correspond to the board type listed in the physical and logical descriptions.

Table 2-3 FRUs Corresponding to Board Type

Board	FRU
NODE	PCA motherboard IP34
PIMM0	PCA PIMM R14KA 500MHz w/2MB + heatsink PCA PIMM R14KA 600MHz w/4MB + heatsink
DIMM0	PCA 256MB DIMM 128MB DDR PCA 512MB DIMM 128MB DDR PCA 1GB DIMM 128MB DDR
DIMM1	PCA 256MB DIMM 128MB DDR PCA 512MB DIMM 128MB DDR PCA 1GB DIMM 128MB DDR
DIMM2	PCA 256MB DIMM 128MB DDR PCA 512MB DIMM 128MB DDR PCA 1GB DIMM 128MB DDR
DIMM3	PCA 256MB DIMM 128MB DDR PCA 512MB DIMM 128MB DDR PCA 1GB DIMM 128MB DDR
PCIE1	PCA motherboard IP34
PCIE2	PCA motherboard IP34
PCIF1	PCA motherboard IP34
PCIF2	PCA motherboard IP34
ODYSSEY	PCA Odyssey VPro GFx, 32 MB, V10 PCA Odyssey VPro GFx, 128 MB, V12 PCA Odyssey X2-brick dual channel

2.5 Scan-based Link-level Protocol Interconnect Test

The Bedrock and Xbridge ASICs contain an additional boundary scan register (LLP register) that can be used to send a single micropacket (160 bits in eight 20-bit, time-MUXed transfers) from one LLP port to another port. The scan-based link-level protocol (LLP) interconnect test (`slit`) uses the LLP registers to exercise links at-speed. It tests one or two ports at a time.

This test uses a minimal number of patterns to determine whether a port is functional. If it detects an error, it attempts to isolate the failure to a single net. If it detects a single failing net, it identifies all chips and pins on the net and displays the actual/expected data.

2.5.1 Test Algorithm

This test uses the following algorithm:

1. Load the chain configuration and reference data.
2. Load the LLP link configuration and selection data.
3. Initialize JTAG communication to the scan controller.
4. Initialize the chain TAPs and configure the SIC.
5. Perform the scan hardware integrity tests:
   - Perform the IR length test.
   - Perform the chain-level LLP REG0 length test.
   - Perform the chain-level LLP REG1 length test.
6. Perform the scan-based LLP interconnect test.
7. Print a program termination status message.

Caution: Each test step must pass before the next step can be run. If the integrity tests in Step 5 do not pass, the interconnect test does not run because the results would be invalid. Do not manually run the interconnect test until the integrity tests pass.

2.5.2 Running the Scan-based Link-level Protocol Interconnect Test

The easiest way to run the boundary scan interconnect test is to use the `scan_asterix` script to automatically run the test. (Refer to "scan_asterix" on page 2-23.)

You can also manually run the test with the `slit` command. This command uses the following syntax:

`slit [-h] <-f` *file*`> <-i` *file*`> [-ID` *num*`] [-d` *file*`] [-p` *num*`] [-e` *num*`] [-t` *test*`] [-m` *mode*`] [-l` *level*`] [-s` *mode*`] [-L` *pats*`] [-F` *bit_mask*`] [-R] [-v] [--l2 <`*host*`> | --interface <`*iface*`> | --ssn <`*serial*`>[:<`*rack*`>] | --sysname <`*system_name*`>[:<`*rack*`>] | --scdev <`*device*`> | --serial <`*host*`>:<`*port*`> | --dev <`*device_name*`>]`

Table 2-4 describes the command-line options that you can use to control `slit`.

Table 2-4 slit Command-line Options

Option	Description
`-h`	Displays the available command-line options and then exits.
`-f <`*file*`>`	Specifies a configuration file to use. The configuration file defines the hardware to test. (Default: `slit.cfg`) The following configuration files are available in `/stand/sysco/cfg/scan` to test Silicon Graphics Fuel visual workstations: `ip34_002_ast.cfg` (nets between Xbridge ASIC and graphics board `ip34_002_po.cfg` (nets between processor and cache) `ip34_002_np.cfg` (nets between PIMM, Xbridge ASIC, and Bedrock ASIC)
`-i <`*file*`>`	Specifies a file that contains input for the test. The input file describes the port-to-port connections that will be tested. (Default: `slit.in`) The following file is available in `/stand/sysco/cfg/scan` to test Silicon Graphics Fuel visual workstations: `ip34_002.links`
`-ID <`*num*`>`	Overrides the `ID` field listed in a configuration file. This enables you to select a different brick for testing without modifying the configuration file. Use the following formula to select a different brick: ID = (rack * 64) + slot
`-d <`*file*`>`	Specifies a file to receive output from the test. (Default: `slit.dmp`)
`-p <`*num*`>`	Specifies the number of test passes that `slit` should perform. (Valid values are between 1 and 10,000; default: 1)
`-e <`*num*`>`	Specifies the maximum number of errors to display from the interconnect test. (Range: 0 to 10,000; default: 200) If you specify 0 errors, no detailed failure information is reported, which may be useful if the test is run from an automated application.

Table 2-4 (continued) slit Command-line Options

Option	Description
`-t <test>`	Specifies the tests to run. Valid tests: irp = instruction register path test irl = instruction register length test ir = instruction register path test and instruction register length test prp = bypass register path test prl = bypass register length test pr = bypass register path test and bypass register length test lrp = LLP register path test lrl = LLP register length test lr = LLP path test and boundary length test llp = LLP at-speed interconnect test id = read and display device ID registers all = all tests Default test selection: irl+lrl+llp
`-m <mode>`	Specifies the mode for the register tests. Valid modes: normal = normal lengths and minimal pattern(s) elen = extended lengths epat = extended pattern set If normal is specified, then no other option is valid. (Default: normal)
`-l <level>`	Specifies the level of the register tests. Valid modes: chn = chain level register test brd = board level register test chip = chip level register test (Default: chn)
`-s <mode>`	Selects a stepping mode. Valid modes: no = no test stepping; run continuous all = step through each pattern; prompt user err = step through failing pattern; prompt user (Default: no)

Table 2-4 (continued) slit Command-line Options

Option	Description
`-L <pats>`	Selects the LLP test pattern that the test uses. Valid options: detect = patterns to detect interconnect faults isolate = patterns to isolate short/bridging faults stress = patterns to stress the interconnects custom = user-defined patterns (Default: detect)
`-F <mask>`	Selects the LLP flits that the test should capture and compare. (Each test packet contains eight separate flits.) The mask parameter is a bitmap of the chosen flits: If bit 0 is set, then flit 0 is captured/compared. If bit 1 is set, then flit 1 is captured/compared. (This parameter can be a hexadecimal number between 0x01 and 0xFF; default: 0xFF)
`-R`	Disables use of resets.
`-v`	Selects verbose mode, which writes additional messages to the output.
`--l2 <host>`	Connects to the L2 system controller with the specified hostname or IP address
`--interface <iface>`	Connects to the L2 system controller with the specified network interface (for example, eth0)
`--ssn <serial>[:<rack>]`	Connects to the system with the specified serial number (for example, L01234567)
`--sysname <system_name>[:<rack>]`	Connects to the system specified by <*system_name*>
`--scdev <device>`	Connects to the specified system controller
`--serial <host>:<port>`	Connects to the specified Ethernet hostname and port
`--dev <device_name>`	Connects to the specified local serial port device

2.5.3 Error Output

When this test detects an error, it continues testing the entire suite of test vectors before it generates error output. Once it has completed the entire suite of test vectors, it generates output that shows the actual and expected values for a miscompare. It also displays the specific chip and pin information for each net that had a miscompare. (The failing chip and pin are identified by logical location.)

The error output uses the following format:

```
Errors were detected during interconnect test.

Failure information for scan-based LLP interconnect test:

Pattern (8-bits each) : ---01--- ---02--- ---03--- ---04---
Actual data           : 01101010 0000.... 0000.... 10000000
Difference data       :   ^^     ^^ ^     ^^ ^       ^ ^  ^
Logical description   : 101a01:U0:PORT_A_OUT_DATA_6
[101a01:U0:PORT_A_IN_DATA_6]
```

The error information contains the following information:

- `Pattern`: indicates the data pattern that `slit` was using when the miscompare occurred. (Each data signal transfers 8 bits of data per test pattern. The number of patterns applied is determined by the pattern type and LLP version. `slit` supports three predefined pattern sets [detect, isolate, and stress] and user-defined custom patterns.)
- `Actual data`: shows the actual data that was read from the system. (If valid data was not captured for a specific pattern [due to test configuration or a problem] a period (.) is shown.)
- `Difference data`: contains a ^ symbol for each pattern in the signature where the actual data did not match the expected data.
- `Logical description`: provides a logical description of the ports on the failing connection.

  The format of each node is `<board>:<chip>:<pin>`. The driver is listed first, followed by the receiver; the receiver is the node that detected the miscompare.

  The format of the <pin> is `PORT_<port>_<dir>_DATA_<num>`, in which:

  `<port>` = LLP port on ASIC

  `<dir>` = `OUT` for SSD output; `IN` for SSR input

  `<num>` = Data line number (0 - 19)

  Note: The test can only identify a differential pair; it cannot isolate which signal of the pair is failing.

2.5.4 Determining the Hardware to Replace

If this test fails, the Bedrock ASIC, Xbridge ASIC, or a connection between them is failing. The FRU that contains both ASICs and the connections between them is the PCA motherboard IP34.

2.6 scan_asterix

The `scan_asterix` script automatically runs the `scantest` and `slit` tests on a Silicon Graphics Fuel visual workstation. This script uses the following syntax:

```
scan_asterix [-p] [-h]
```

Table 2-5 describes the command-line options that you can use to control `scan_asterix`.

Table 2-5 scan_asterix Command-line Options

Option	Description
`-p`	Disables the script from prompting you to press <Enter> before it starts each test
`-h`	Displays help information

Perform the following procedure to run this script:

1. Connect a USB cable from an external system that is running the L3 controller software to the diagnostic port on the back of the workstation. (Refer to Figure 2-13.)

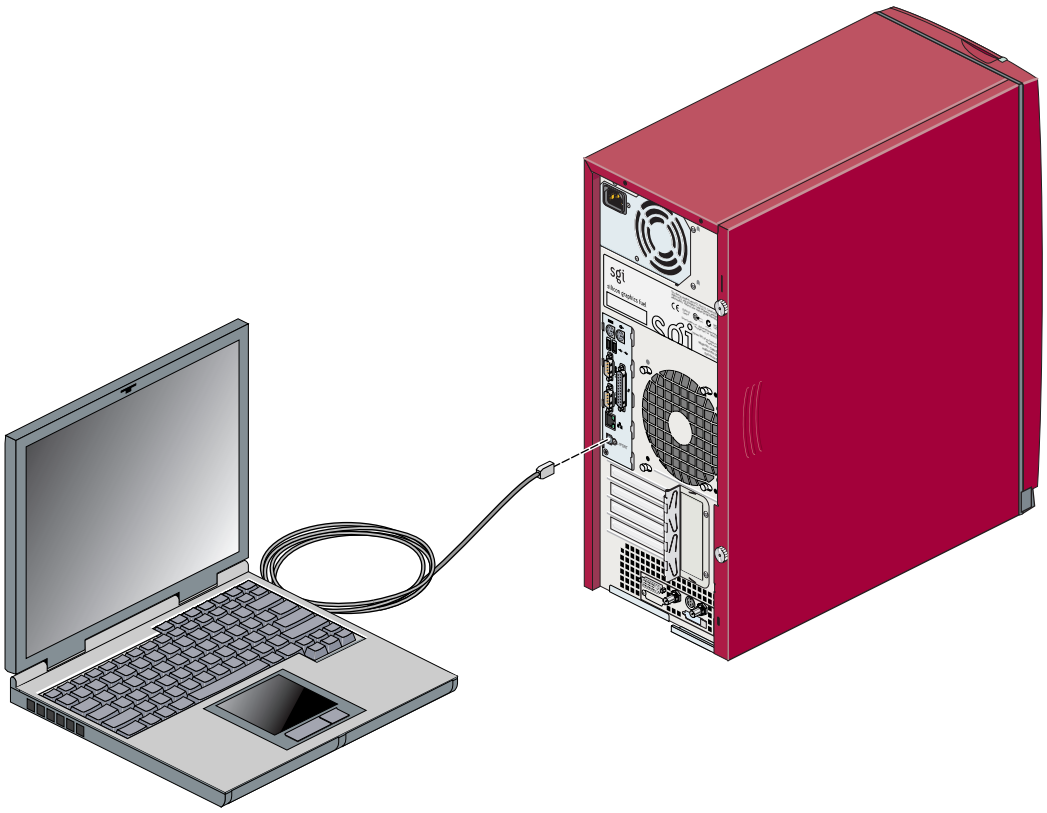

Figure 2-13 Connecting a System to the Diagnostic Port to Run the scan_asterix Script

2. Log into the system that is running L3 system controller software as `sgidiag` (default password: `sgi!diag`).
3. Enter **`cd /stand/sysco/usr`** to change to `/stand/sysco/usr` directory.
4. Enter **`/stand/sysco/bin/l2`** to start the L2 emulator.
5. Power up the workstation.
6. Ensure that the workstation is configured as 001a01 (rack 1, slot 1)
7. Enter **`/stand/sysco/bin/scan_asterix`** to start the script.
8. Select the tests that you want to run.

   Table 2-6 describes the tests that are valid for field use. (Test options 4, 6, 7, 8, 11, and 12 require special hardware that is not available in the field.)

Table 2-6 scan_asterix Test Options

Option	Description
1	Runs `scantest` to check the interconnects between the Xbridge ASIC, Bedrock ASIC, IOC3 ASIC, and SCSI connectors
2	Runs `scantest` to check the processor cache by checking the interconnects between the CPUs and cache SRAMs
3	Runs `scantest` to check the interconnects between the PIMM, Xbridge ASIC, and Bedrock ASIC
5	Runs `scantest` to check the interconnects on the IP34 motherboard, VPro Odyssey graphics board, and the interconnects between them
9	Runs `slit` to check the interconnects between the Xbridge ASIC and Bedrock ASIC at speed
10	Runs all of the tests that do not require test boards or additional cables (options 1 and 2)

9. Interpret the output. The script returns output to the screen and to the following files in the directory from which you started the script:
   - `ip34_002.dmp` (`scantest` output from test option 1)
   - `ip34_002_po.dmp` (`scantest` output from test option 2)
   - `ip34_002_np.dmp` (`scantest` output from test option 3)
   - `ip34_002_ast.dmp` (`scantest` output from test option 5)
   - `slit.dmp` (`slit` output from test option 9)
   - `all_tests.dmp` (output from all tests)

Example:

```
/stand/sysco/usr> /stand/sysco/bin/scan_asterix

--- SCAN INTERCONNECT TEST OF IP34/Asterix (scan_ip34 version 1.2) ---

Run scan interconnect tests on Asterix brick with IP34 node board.
Brick location MUST be set to rack=1, slot=1 during scan test.

********** RUNNING IN SIMULATION MODE **********

There are several interconnect tests with different areas of coverage.
For complete test of an Asterix brick, install scannable four DIMMs,
  four scannable PCIs and loopback cables; run all available tests.

Select from the following interconnect tests available:
1) IP34 Node only (nets between XBRIDGE, BEDROCK, IOC3, and SCSI)
2) IP34 processor cache (nets between CPUs and SRAMs)
3) IP34 system interface (nets between PIMM, XBRIDGE and BEDROCK)
4) IP34 main memory (nets between BEDROCK and scannable DIMMs)
5) Asterix brick (IP34, Odyssey board and nets between)
6) Asterix brick PCI E slots (nets between IP34 and PCI E bus slots)
7) Asterix brick PCI F slots (nets between IP34 and PCI F bus slots)
8) Asterix brick PCI E & F slots (nets between IP34 and PCI E & F bus slots)
9) IP34 at-speed LLP test (BEDROCK and XBRIDGE)

10) All tests for Asterix brick stand-alone (1, 2)
11) All tests for Asterix with SDTBs (1-5,9)
12) All tests for Asterix with SDTBs, SPTBs (1-5,8,9)
> 10

The L2 software emulator must be running if real L2 not used.
Turn on power to boards under test (IP34/PCI).
Brick location MUST be set to rack=1, slot=1 during scan test.

--- TEST 1: IP34 Node Only ---
Press ENTER when ready to start running selected test...
<Enter>

Boundary-Scan Test Software (scantest V3.13) Fri Nov 30 16:20:10 2001

Loading chain configuration and reference data
   File: /stand/sysco/cfg/scan/ip34_002.cfg

Loading reference data for each board definition
   Path: /stand/sysco/data/scan/cmp

Initializing JTAG communications to scan controller(s)
   No hardware; running in software simulation/emulation mode

Initializing chain TAPs and configuring SICs:
        001a01 with SIC at address 0x00 (CER=0x06)

Testing Integrity of Boundary Scan Instruction Registers
Performing IR length test on active UUTs:
        001a01 SIC 0x0 ... passed (chain length=19)
```

```
Performing IR path test on active UUTs:
        001a01 SIC 0x0 ... passed

Testing Integrity of Boundary Scan Bypass Registers
Performing BYPASS length test on selected UUTs:
        001a01 SIC 0x0 ... passed (chain length=6)

Performing BYPASS path test on selected UUTs:
        001a01 SIC 0x0 ... passed

Testing Integrity of Boundary Scan Data Registers
Performing chain-level BOUNDARY length test on selected UUTs:
        001a01 SIC 0x0 ... passed (chain length=3196)

Performing chain-level BOUNDARY path test on selected UUTs:
        001a01 SIC 0x0 ... passed

Performing system-level interconnect test on given configuration:
        (22 of 22 patterns will be applied)

Results stored in log file: ip34_002.dmp

--- TEST 2: IP34  processor cache ---
Press ENTER when ready to start running selected test...
<Enter>

Boundary-Scan Test Software (scantest V3.13) Fri Nov 30 16:20:14 2001

Loading chain configuration and reference data
   File: /stand/sysco/cfg/scan/ip34_002_po.cfg

Loading reference data for each board definition
   Path: /stand/sysco/data/scan/cmp

Initializing JTAG communications to scan controller(s)
   No hardware; running in software simulation/emulation mode

Initializing chain TAPs and configuring SICs:
        001a01 with SIC at address 0x00 (CER=0x01)

Testing Integrity of Boundary Scan Instruction Registers
Performing IR length test on active UUTs:
        001a01 SIC 0x0 ... passed (chain length=19)

Performing IR path test on active UUTs:
        001a01 SIC 0x0 ... passed

Testing Integrity of Boundary Scan Bypass Registers
Performing BYPASS length test on selected UUTs:
        001a01 SIC 0x0 ... passed (chain length=6)

Performing BYPASS path test on selected UUTs:
        001a01 SIC 0x0 ... passed

Testing Integrity of Boundary Scan Data Registers
Performing chain-level BOUNDARY length test on selected UUTs:
        001a01 SIC 0x0 ... passed (chain length=708)
```

```
Performing chain-level BOUNDARY path test on selected UUTs:
        001a01 SIC 0x0 ... passed

Performing board-level interconnect test on selected boards:

        001a01 PIMM0 (#Patterns=18)
        ++++++++++++++++++
        No errors detected during interconnect test.

Normal Program Termination
Results stored in log file: ip34_002_po.dmp

NOTE: Check results for each of the individual tests run.
      Results for all tests stored in log file: all_tests.dmp
```

2.7 scantool

`scantool` enables you to control scan operations from Tcl/Tk scripts. These scripts can control the following test functions that are accessible via the JTAG/scan hardware:

- Memory built-in self-test (BIST)
- Clock control
- Auxiliary access to memory-mapped registers (MMRs)
- Internal latch dumping
- Debug port control

`scantool` includes graphical user interface and command-line modes.

Table 2-7 describes useful scripts that are available in the current diagnostic release.

Table 2-7 scantool Scripts (Located in /stand/sysco/data/scan/diags/ip34)

Script	Description
`ip34_dump_br_latches.tk`	Dumps the Bedrock ASIC latches (This script could be used with the FRU analyzer to dump the state of the Bedrock ASIC when a brick does not respond to an NMI.)
`ip34_membist.tcl`	Runs the on-board memory built-in self test (BIST) feature on the Bedrock ASIC (This script is not used in the field.)

2.7.1 scantool Command-line Options

`scantool` uses the following command syntax:

`scantool [-h] [-l] [-c <config_file>] [-f <`*script_name*`>] [-g] [-n | -s |-0 | -1] [--l2 <`*host*`> | -- interface <`*iface*`> | --ssn <`*serial*`>[:<`*rack*`>] | --sysname <`*system_name*`>[:<`*rack*`>] | --serial <`*host*`>:<`*port*`> | --dev <`*device_name*`>]`

Table 2-8 describes the command-line options that you can use to control `scantool`.

Table 2-8 scantool Command-line Options

Option	Description
`-h`	Displays help information
`-l`	Starts `scantool` in command-line mode (a prompt appears at which you can enter `scantool` commands) This option is required for command-line mode.
`-c <`*config_file*`>`	Selects the configuration file to use This option is required for command-line mode.

Table 2-8 (continued) scantool Command-line Options

Option	Description
`-f` <*script_name*>	Selects the script to run You can also use the graphical user interface to select which script to run.
`-g`	Enables scantest to run graphical Tk scripts in command-line mode
`-n`	Sets the communication method to “null” mode: `scantool` runs the script but does not send the commands to the scan hardware. No scan data is received from the hardware.
`-s`	Sets the communication method to “simulation” mode: `scantool` connects to a Verilog simulator through a socket connection. (This configuration is not used in the field.)
`-0`	Sets the communication method to “SN0” mode: `scantool` connects to an external system through the parallel port of a Silicon Graphics Indy visual workstation. (This configuration is not used in the field.)
`-1`	Sets the communication method to “SN1” mode: `scantool` connects to a system through an L3 system controller. The L3 system controller must be connected to an L2 system controller or it must be running the L2 emulator. This is the default communication method.
`--l2` <*host*>	Connects to the L2 system controller with the specified hostname or IP address
`--interface` <*iface*>	Connects to the L2 system controller with the specified network interface (for example, eth0)
`--ssn` <*serial*>`[:`<*rack*>`]`	Connects to the system with the specified serial number (for example, L01234567)
`--sysname` <*system_name*>`[:`<*rack*>`]`	Connects to the system specified by <*system_name*>
`--serial` <*host*>`:`<*port*>	Connects to the specified Ethernet hostname and port
`--dev` <*device_name*>	Connects to the specified local serial port device

2.7.2 Using scantool to Dump the Internal Bedrock Latches

If a system hangs and does not respond to an NMI, it is sometimes useful to dump the internal Bedrock latches. You may be asked to collect this data and return it to SGI for further analysis.

To dump the internal Bedrock latches from the Bedrock ASIC, perform the following procedure:

1. Connect a USB cable from an external system that is running the L3 controller software to the diagnostic port on the back of the workstation. (Refer to Figure 2-14.)

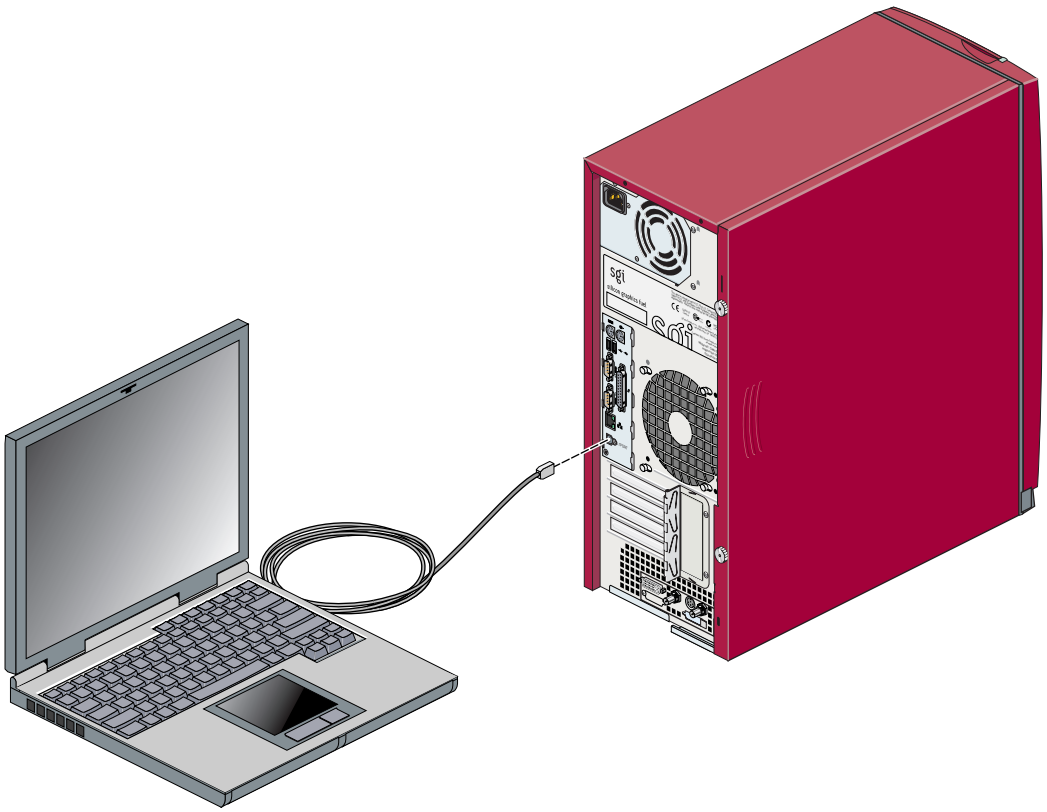

Figure 2-14 Connecting a System to the Diagnostic Port to Run scantool

2. Log into the system that is running the L3 system controller software as `sgidiag` (default password: `sgi!diag`).
3. Enter the following command to start the L2 emulator:

   `/stand/sysco/bin/l2`

4. Ensure that the workstation is configured as 001a01 (rack 1, slot 1)
5. Enter **`scantool`** at the L3 system controller prompt.

   Figure 2-15 shows the `scantool` graphical user interface (also called the graphical console).

Figure 2-15 scantool Graphical User Interface (Initial Window)

6. Load the configuration file:
   - Choose **`File -> Load Config...`**
   - Double-click on `scan`.
   - Double-click on the `ip34_001_st.cfg` configuration file. (Refer to Figure 2-16.)

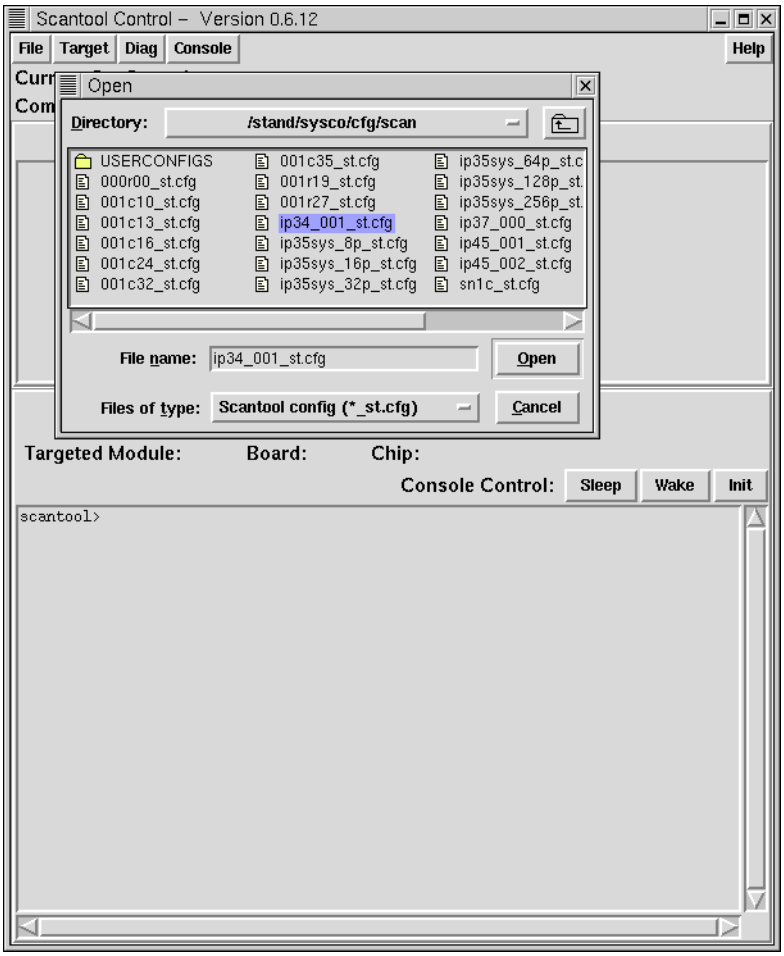

Figure 2-16 Loading the Configuration File

7. Select the module to use:
   - Choose **`Diag -> Start Console Diag...`**
   - Double-click on `001a01`. (Refer to Figure 2-17.)

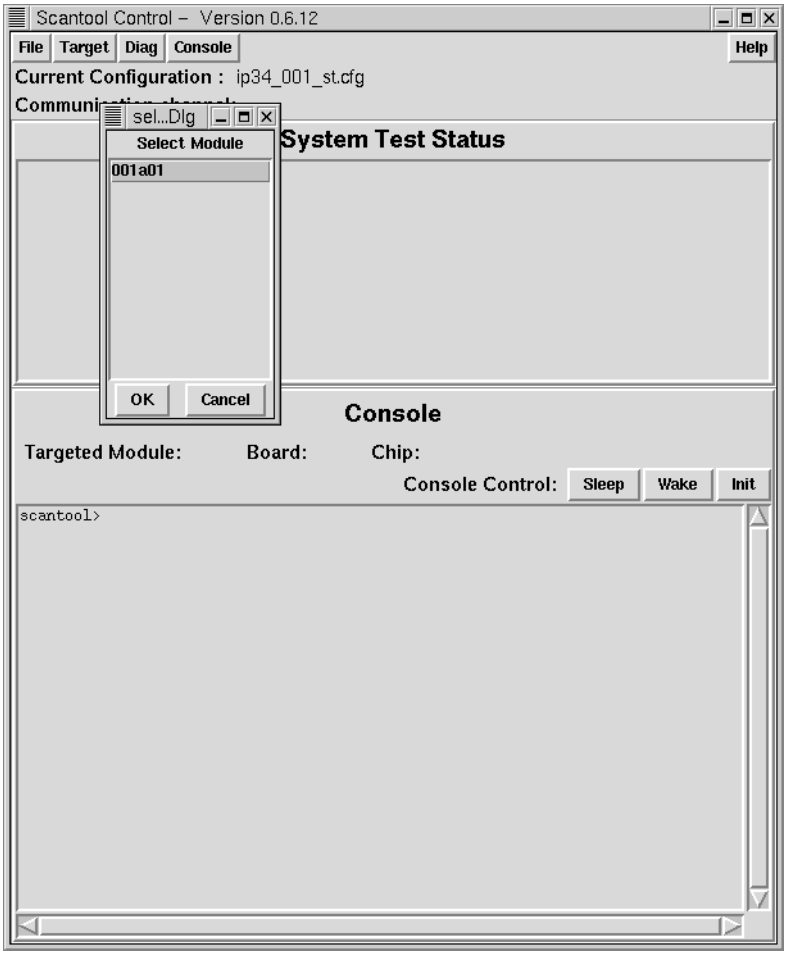

Figure 2-17 Selecting the Module to Use

The interface displays the `Open` window.

8. Load the `ip34_dump_br_latches.tk` script:
   - Double-click on `ip34`.
   - Double-click on `ip34_dump_br_latches.tk`. (Refer to Figure 2-18.)

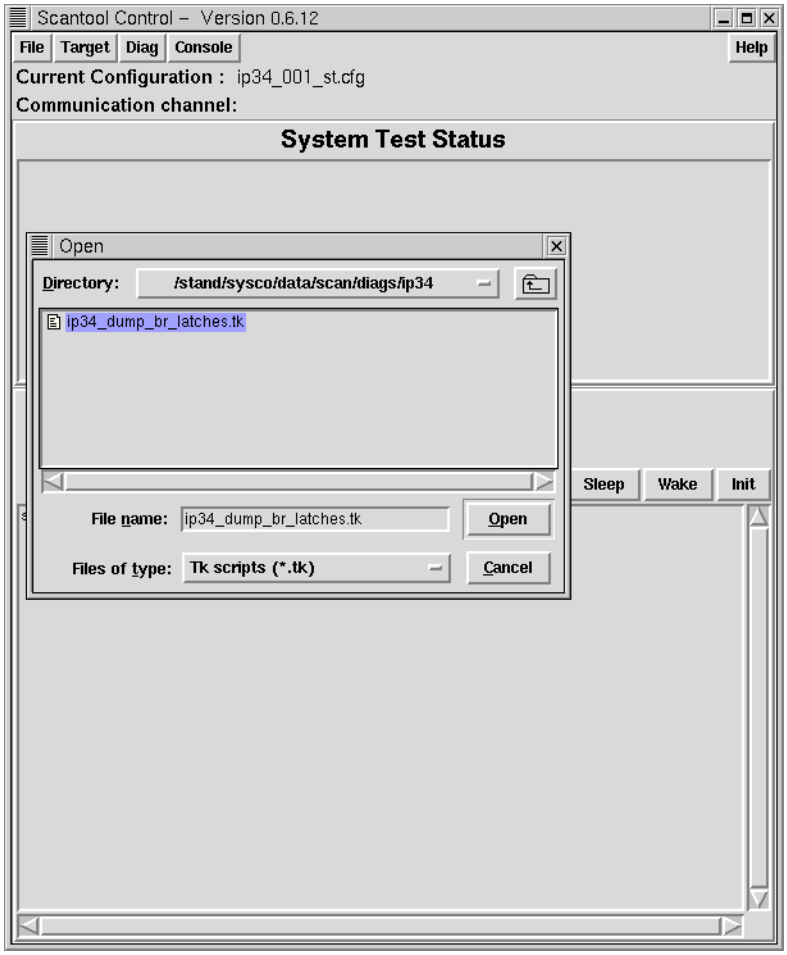

Figure 2-18 Loading the ip34_dump_br_latches.tk Script

The interface displays the `IP34 Bedrock Internal Latch Dump` window. (Refer to Figure 2-19.)

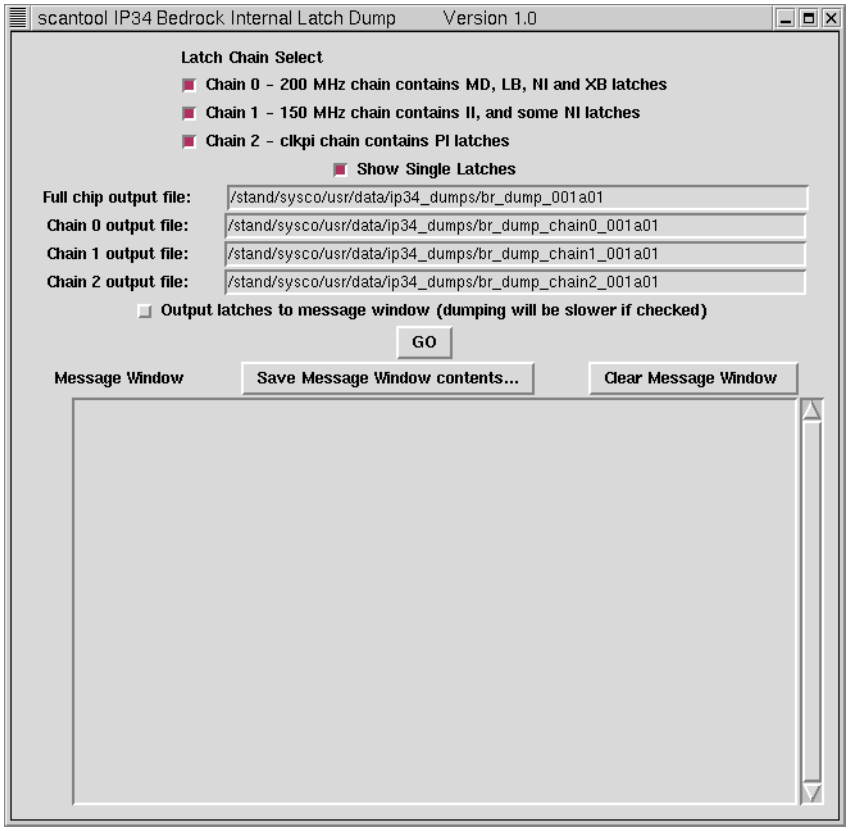

Figure 2-19 IP34 Bedrock Internal Latch Dump Window

9. Modify the parameters, if necessary.

   The parameters select the scan chains to use, the output files to use, and whether or not `scantool` should show single latches and output the latches to the `Message Window`.

   Note: Normally, you can use the default parameter settings.

10. Click on the `GO` button.

The interface displays a warning message. (Refer to Figure 2-20.)

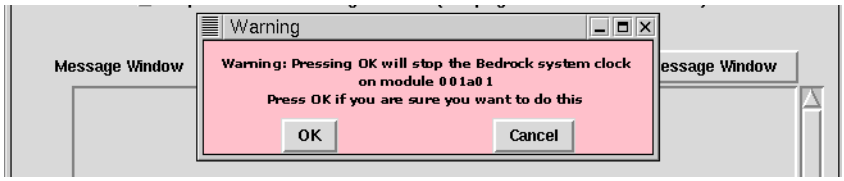

Figure 2-20 Bedrock System Clock Warning Message

11. Click on the `OK` button.

    Output appears in the lower portion of the window. (Refer to Figure 2-21.) Output is also written to the specified files.

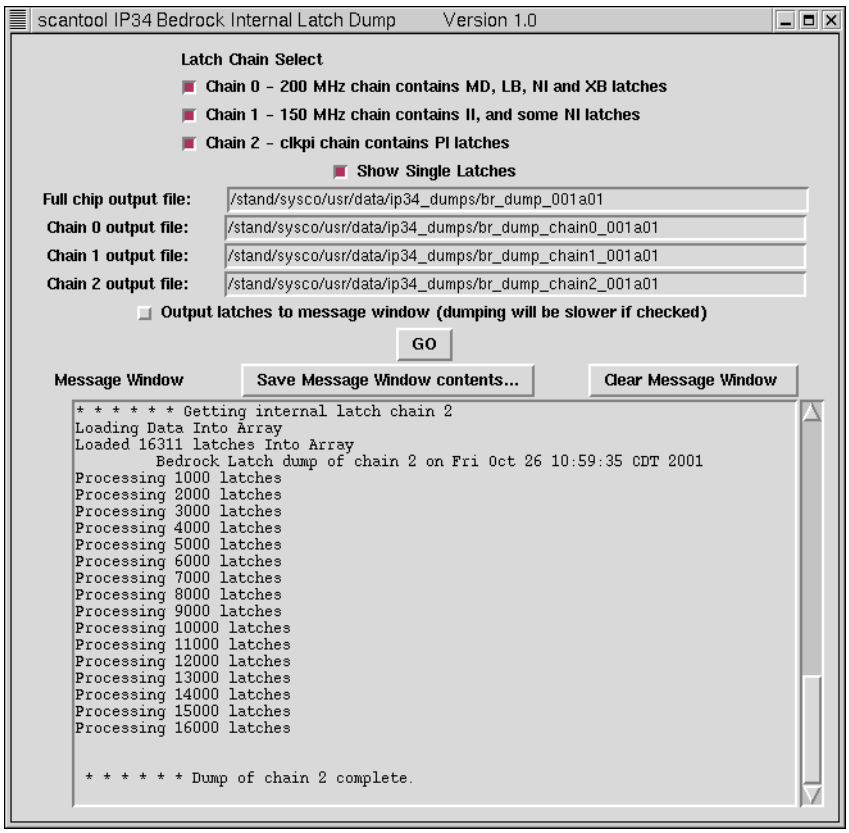

Figure 2-21 Output from the ip34_dump_latches.tk Script

12. When you are finished viewing the output, close any windows that `scantool` opened and then choose **`File -> Exit`**.
13. Run the `reboot_l1` command in the L2 emulator window.
14. Reset the system.

Chapter 3

Power-on Diagnostics

Power-on diagnostics are PROM-resident tests that run automatically when you power on a Silicon Graphics Fuel visual workstation. As the boot process discovers hardware components, it runs the power-on diagnostics needed to verify that each component is functional enough to load the operating system. You can also run several of the power-on diagnostics manually.

PROM chips on the IP34 motherboard store the power-on diagnostics. The PROM chips contain two PROM images:

- The IP35 PROM image runs the processor, memory, Bedrock ASIC (also called *hub* by the power-on diagnostics), Xbridge ASIC, IOC3 ASIC, PCI slot, and I/O diagnostics.
- The BaseIO PROM image runs a SCSI channel test to check the I/O path to the system disk controller, keyboard and mouse tests to check the keyboard and mouse connections, graphics tests to check the graphics hardware on the VPro graphics board, and Ethernet tests to check the Ethernet hardware on the motherboard.

Note: The term "IP35 PROM" is used throughout the power-on diagnostic output and this manual because these diagnostics were originally developed for IP35 systems.

Table 3-1 lists the power-on diagnostics that are available.

Table 3-1 Power-on Diagnostics

Name	Description
n/a	Register test
`cache_test_i`	Primary instruction cache test
`cache_test_d`	Primary data cache test
`cache_test_s`	Secondary cache test
`hub_intrpt_diag`	Hub interrupt test
n/a	Hub local test
n/a	Hub config test
n/a	Hub UART test
`hubsde`[a]	Hub send-data test
`test_hub_error_detection`[a]	Hub error-detection test

Table 3-1 (continued) Power-on Diagnostics

Name	Description
rtrsde[a]	Router send-data test
test_rtr_error_detection[a]	Router error-detection test
get_chipid[a]	Get Chipid test
rt_clock_test	GCLk logic test
hub_bte_diag	Block transfer engine test
prom_checksum	PROM checksum test
n/a	Memory configuration test
dirtest	Backdoor directory space test
memtest	Memory test
chklink[a]	NUMAlink test
xbow_sanity	XBOW test
bridge_sanity	Bridge test
io_config_space	PCICONFIG test
pcibus_sanity	PCI bus test
serial_pio	Serial programmed I/O test
serial_dma	Serial DMA test
scsi_ram	SCSI controller SSRAM test
scsi_dma	SCSI controller DMA test
scsi_controller	SCSI controller self-test
pckm	Keyboard and mouse test
gfx	Graphics test
enet_all	Comprehensive Ethernet test
enet_ssram	Ethernet SSRAM test
enet_phy_reg	PHY register test
enet_tx_clk	TX clock test
enet_ioc3_loop	IOC3 internal loopback test
enet_phy_loop	PHY chip internal loopback test
enet_tw_loop	Twister chip internal loopback test
enet_ext_loop	External loopback DMA test
enet_10_ext_loop	External loopback DMA (10 Mb/s) test
enet_100_ext_loop	External loopback DMA (100 Mb/s) test

Table 3-1 (continued) Power-on Diagnostics

Name	Description
`enet_xtalk_stress`	Xtalk stress test

a. These IP35 router and hub tests are disabled because they do not apply to the IP34 hardware; these tests are not documented in this manual.

Figure 3-1, Figure 3-2, and Figure 3-3 show the components that are tested when the power-on diagnostics run automatically during the boot process.

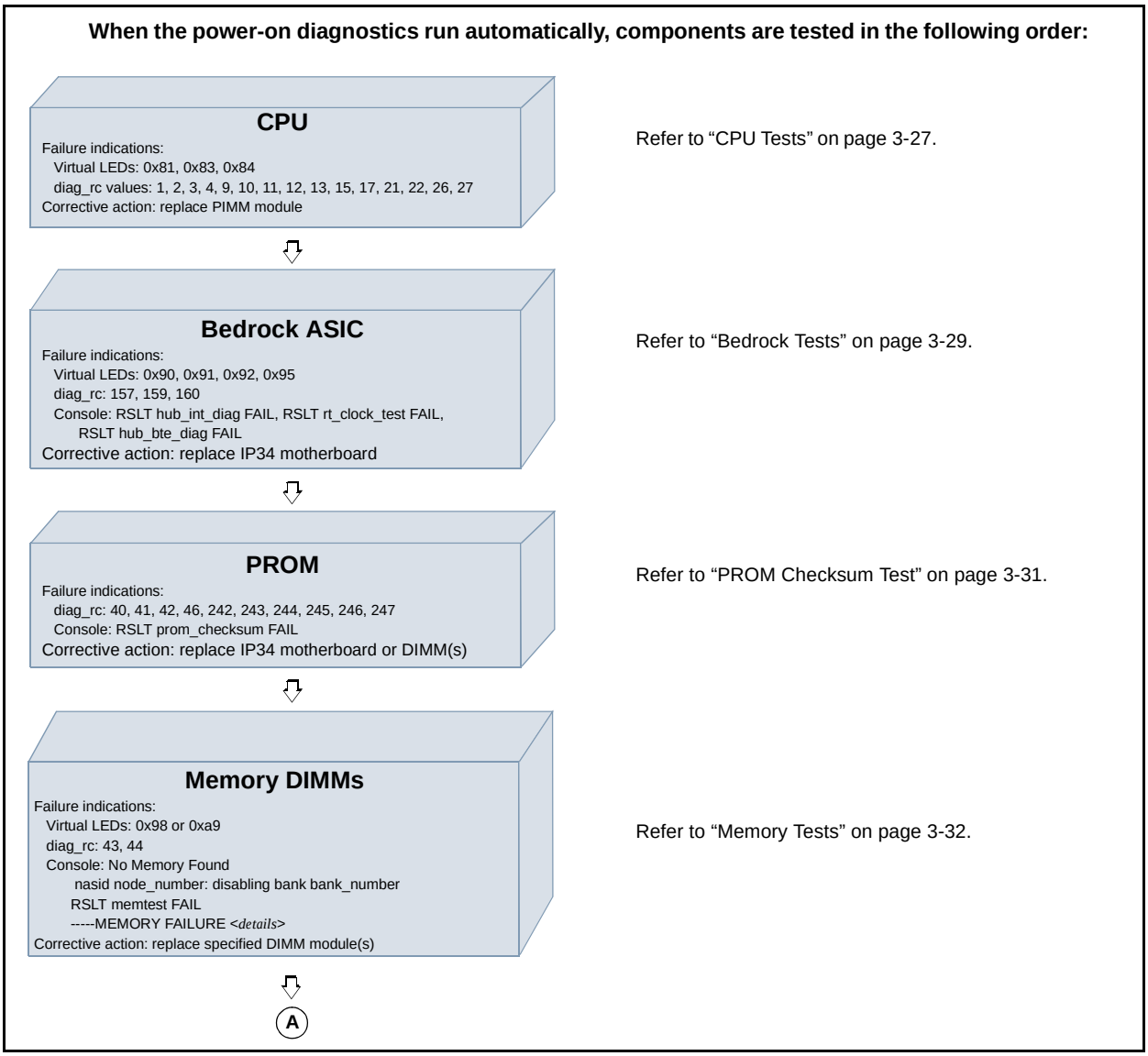

Figure 3-1 Components Tested by Power-on Diagnostics (Part 1 of 3)

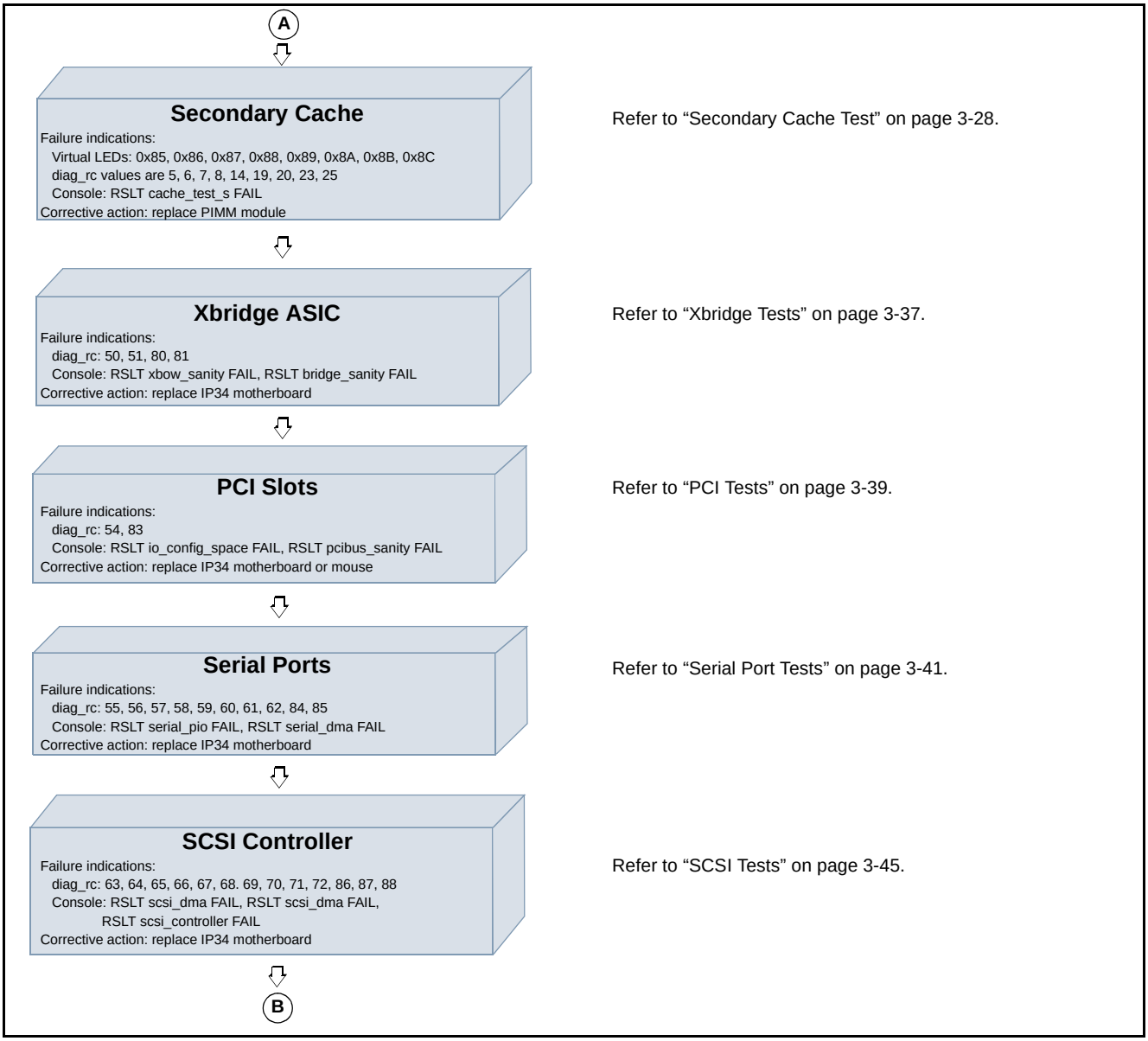

Figure 3-2 Components Tested by Power-on Diagnostics (Part 2 of 3)

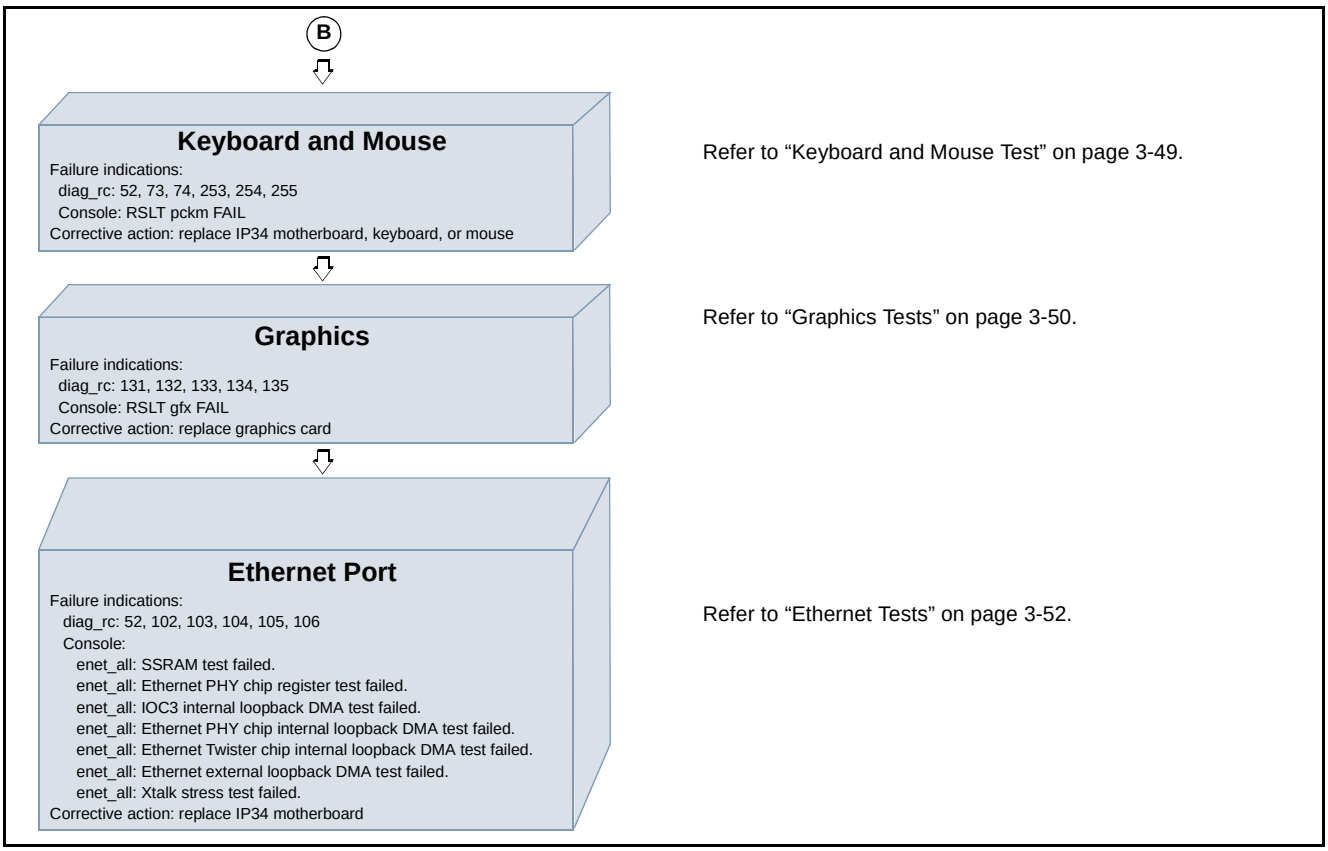

Figure 3-3 Components Tested by Power-on Diagnostics (Part 3 of 3)

3.1 Distribution

Each IRIX operating system release includes an updated image of the power-on diagnostics. When you install an operating system, the updated power-on diagnostics are automatically flashed to the PROM chips.

3.2 Running the Power-on Diagnostics Automatically

The power-on diagnostics run automatically when you power-on or reset the system. As the boot process discovers hardware, it verifies that each component is functional enough to load the operating system.

The power-on diagnostics run in four modes:

- No diags (or none) mode performs no diagnostic testing; the system boots as quickly as possible. You might use this mode to debug software such as kernel drivers, but use it only when you have complete confidence that the hardware is operating correctly.
- Normal mode tests each part of the system for basic functionality, but it uses relatively fast tests to expedite system boot as well as catch any severely impaired hardware.
- Heavy mode runs the most thorough diagnostic testing available on each part of the system. The tests may take a very long time to complete, especially the memory tests. Therefore, it may be desirable to run them after you install new hardware or if the system is having problems that are thought to be hardware-related.
- Manufacturing runs heavy diagnostic testing and outputs special FRU (field replaceable unit) information. The system controller port, which must be connected to SGI manufacturing equipment, handles console input and output.

You can use virtual debug switch settings to select the mode. Use the L1 controller `debug` command to view the virtual debug switch settings. Example 3-1 shows sample output from the `debug` command.

Example 3-1 debug Command Output

```
001a01-L1>debug
debug switches set to 0x0001.
```

Use the `debug` <*switches*> command to set the virtual debug switches. In this command, the variable <*switches*> is a hexadecimal value for the switches (refer to Table 3-2). Example 3-2 shows sample output from the `debug` <*switches*> command.

Example 3-2 debug <*switches*> Command Output

```
001a01-L1>debug 0x0003
debug switches set to 0x0003
```

Table 3-2 lists the switch-setting values and describes their functions.

Table 3-2 Virtual Debug Switch Settings

Category	Value	Description
Diagnostic Testing Level	0x0000	Normal testing
	0x0001	No testing
	0x0002	Heavy testing
	0x0003	Manufacturing-level testing
Diagnostic Output Level	0x0004	Verbose (information level set to verbose)

Table 3-2 (continued) Virtual Debug Switch Settings

Category	Value	Description
Boot Stop Point	0x0000	Normal. Normal setting, do not stop
	0x0008	Global POD
	0x0010	Local POD (boot stop requested at local POD)
	0x0018	Memoryless POD (boot stop requested at no memory POD)
Default Environment	0x0020	Ignore environment variables
Override CPU Disabling	0x0100	Overrides CPUs disabled with the environment variables in POD mode
Hardware Error State	0x1000	Dumps hardware error state at system boot time.
Ignore Autoboot	0x2000	BaseIO PROM ignores the autoboot environment variable.

3.3 Running the Power-on Diagnostics Manually

You can run several of the IP35 PROM image tests manually from the `POD>` prompt. Enter the `pod` command from the command monitor prompt(`>>`) to enter POD mode and display the `POD>` prompt.

Table 3-3 lists the power-on diagnostics that you can run from the `POD>` prompt and indicates the command that you can use to start the tests.

Table 3-3 Commands to Run the IP35 Power-on Diagnostics Manually

Diagnostic Test	Command to Run the Test
Backdoor directory space test	`dirtest [start] [len]`
Memory test	`memtest [start] [len]`
XBOW portion of the Xbridge ASIC test	`dgxbow [mn | mh | mm | mx] [nnasid]`
Bridge portion of the Xbridge ASIC test	`dgbrdg [mn | mh | mm | mx] [nnasid] [bbus]`
PCI configuration space test	`dgconf [mn | mh | mm | mx] [nnasid] [bbus]`
PCI bus test	`dgpci [mn | mh | mm | mx] [nnasid] [bbus] [ppcinum]`
Serial programmed I/O test	`dgspio [mn | mh | mm | mx] [nnasid] [bbus] [ppcinum]`
Serial DMA test	`dgsdma [mn | mh | mm | mx] [nnasid] [bbus] [ppcinum]`
Keyboard and mouse test	`dgpckm [mn | mm]`

You can run the BaseIO PROM image tests from the command monitor prompt (>>). You can access the `>>` prompt by selecting option `Enter Command Monitor` from the `System Maintenance Menu`. Table 3-4 lists the commands necessary to run the BaseIO power-on diagnostics manually.

Table 3-4 Commands to Run the BaseIO Power-on Diagnostics Manually

Diagnostic Test	Command to Run the Test
Comprehensive SCSI test	`diag_fibre [-N nasid] [-m moduleid -b bus] [-p pcidev] [-h] [-v] [-t testname] [-r count| -R count]`
SCSI SSRAM test	`diag_fibre -t ram [-N nasid] [-m moduleid -b bus] [-p pcidev] [-h] [-v] [-t testname] [-r count| -R count]`
SCSI SSRAM DMA test	`diag_fibre -t dma [-N nasid] [-m moduleid -b bus] [-p pcidev] [-h] [-v] [-t testname] [-r count| -R count]`
SCSI self-test	`diag_fibre -t controller [-N nasid] [-m moduleid -b bus] [-p pcidev] [-h] [-v] [-t testname] [-r count| -R count]`
Graphics test	`diag_gfx [-t testname] [-r count | -R count] [-h] [-v]`
Comprehensive Ethernet test	`diag_enet [-N <nasid>] [-m <moduleid> <-b bus>] [-p <pcidev>] [-h] [-e|-x] [-v] [-t <testname>] [-r <count>] [-R <count>]`

Table 3-4 (continued) Commands to Run the BaseIO Power-on Diagnostics Manually

Diagnostic Test	Command to Run the Test
Ethernet SSRAM test	`diag_enet -t ssram [-N <nasid>] [-m <moduleid> <-b bus>] [-p <pcidev>] [-h] [-e|-x] [-v] [-t <testname>] [-r <count>] [-R <count>]`
PHY register test	`diag_enet -t phyreg [-N <nasid>] [-m <moduleid> <-b bus>] [-p <pcidev>] [-h] [-e|-x] [-v] [-t <testname>] [-r <count>] [-R <count>]`
TX clock test	`diag_enet -t txclk [-N <nasid>] [-m <moduleid> <-b bus>] [-p <pcidev>] [-h] [-e|-x] [-v] [-t <testname>] [-r <count>] [-R <count>]`
PHY chip internal loopback test	`diag_enet -t phy_loop [-N <nasid>] [-m <moduleid> <-b bus>] [-p <pcidev>] [-h] [-e|-x] [-v] [-t <testname>] [-r <count>] [-R <count>]`
Twister chip internal loopback test	`diag_enet -t tw_loop [-N <nasid>] [-m <moduleid> <-b bus>] [-p <pcidev>] [-h] [-e|-x] [-v] [-t <testname>] [-r <count>] [-R <count>]`
External loopback DMA test	`diag_enet -t ext_loop [-N <nasid>] [-m <moduleid> <-b bus>] [-p <pcidev>] [-h] [-e|-x] [-v] [-t <testname>] [-r <count>] [-R <count>]`
External loopback DMA test (10 Mb/s)	`diag_enet -t ext_loop_10 [-N <nasid>] [-m <moduleid> <-b bus>] [-p <pcidev>] [-h] [-e|-x] [-v] [-t <testname>] [-r <count>] [-R <count>]`
External loopback DMA test (100 Mb/s)	`diag_enet -t ext_loop_100 [-N <nasid>] [-m <moduleid> <-b bus>] [-p <pcidev>] [-h] [-e|-x] [-v] [-t <testname>] [-r <count>] [-R <count>]`
Xtalk stress test	`diag_enet -t xtalk [-N <nasid>] [-m <moduleid> <-b bus>] [-p <pcidev>] [-h] [-e|-x] [-v] [-t <testname>] [-r <count>] [-R <count>]`

3.4 Power-on Diagnostic Output

The power-on diagnostics send output to the LED lightbar on the front of the workstation, the virtual LEDs, and the diagnostic console. You can view the virtual LED and diagnostic console output by connecting a system running the L3 system controller software to the diagnostic port or by connecting a terminal to the L1 system controller DB-9 connector on the motherboard.

3.4.1 Decoding the LED Lightbar

The LED lightbar on the front of the workstation shows the system status as the system powers up. (Refer to Table 3-5.)

Table 3-5 LED Lightbar Patterns

Pattern	Meaning
Blinking white	The system is powering up.
Solid white	The system powered up successfully and is operating normally.
Solid red	The system has a motherboard failure. (The PROM could not be read.)
Blinking red	The system has a memory failure.
Blinking red and white (alternating)	The system has a graphics failure.

3.4.2 Viewing Power-on Diagnostic Output through the Diagnostic Port

You can connect a system running the L3 system controller software to the diagnostic port (the L1 controller USB-B port) to view the power-on diagnostic output. (Refer to Figure 3-4). Using the connected system, you can see any messages that the power-on diagnostics generate; you can also issue the L1 `leds` command to read the status of the virtual LEDs.

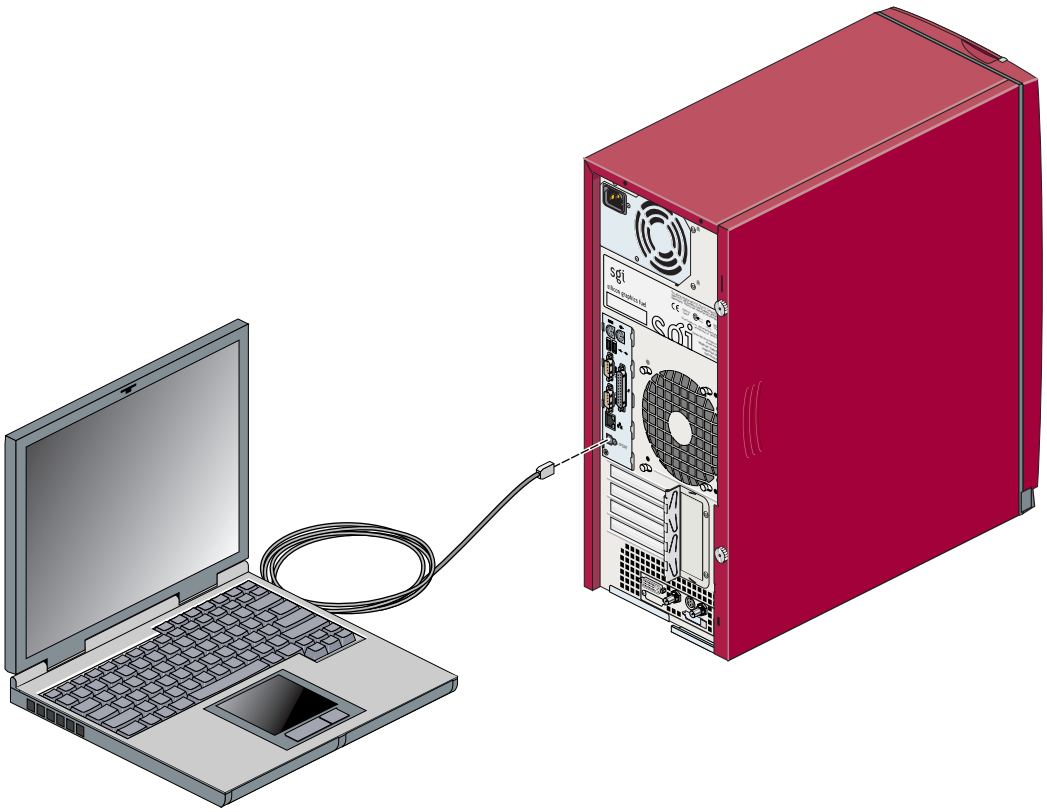

Figure 3-4 Connecting a System to the Diagnostic Port to View Power-on Diagnostic Output

Perform the following procedure to view the power-on diagnostic output through the diagnostic port:

1. Connect a USB cable from an external system that is running the L3 system controller software to the diagnostic (L1) port on the back of the workstation.
2. In a window on the external system, enter **`/stand/sysco/bin/l2`**.
3. Type Ctrl+D to access the diagnostic console.

   Note: Type Ctrl+T to return to the `L2>` prompt. You can enter any L2 command at the prompt. When the command completes execution, the diagnostic console returns. (To permanently engage L2 mode, enter the `l2` command at the `L2>` prompt.) To access L1 mode to view the virtual LEDs, enter the `l1` command at the `L2>` prompt. Type Ctrl+T at the `L1>` prompt to return to the `L2>` prompt.
4. Power up the workstation.

3.4.3 Viewing Power-on Diagnostic Output through the DB-9 Connector

If you do not have the L3 system controller software or USB cables that are required to use the diagnostic port, you can view the power-on diagnostic output by connecting a terminal to a DB-9 connector on the motherboard (refer to Figure 3-5).

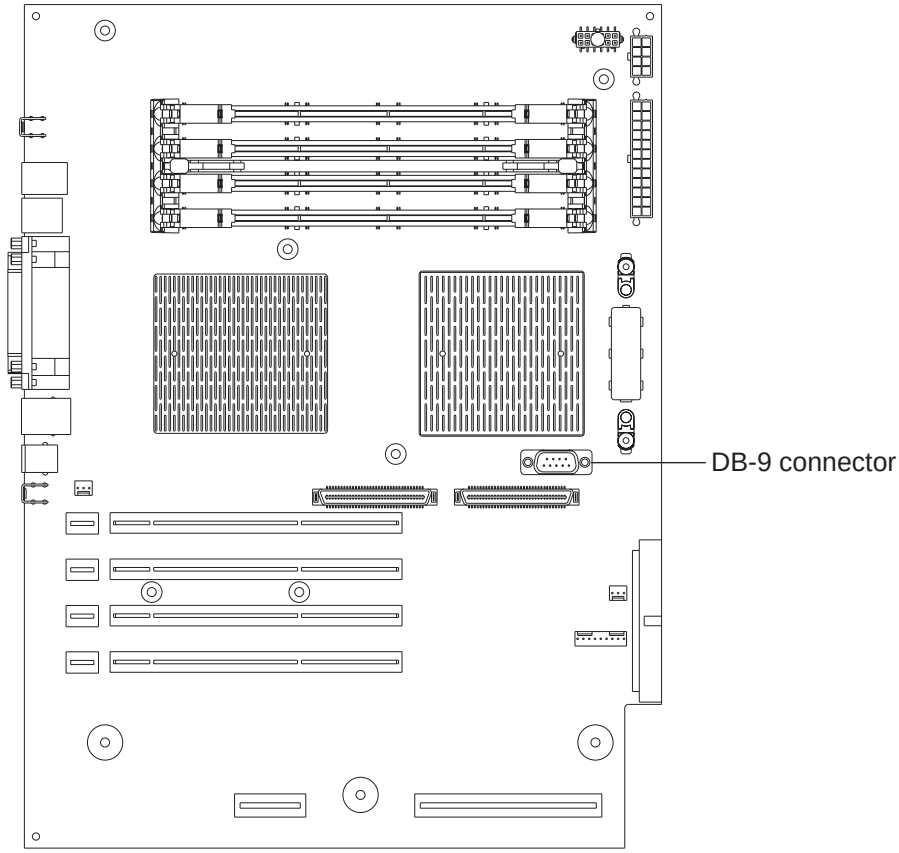

Figure 3-5 DB-9 Connector Location on the Motherboard

Perform the following procedure to view the power-on diagnostic output through the DB-9 connector on the motherboard:

1. Configure your external terminal with the following settings:
   - 38,400 baud
   - 8 data bits
   - No parity
   - 1 stop bit
   - Hardware flow control on (rts/cts)
2. Connect a null modem serial cable from the serial connector on the external system to the DB-9 connector on the motherboard.
3. Initiate a connection between the external terminal and the DB-9 connector (for example, start your terminal emulation software).

   The L1 controller displays `Connected`.
4. Press the Enter key to get to the L1 prompt:

   ```
   001a01-L1>
   ```
5. Type Ctrl+D to access the diagnostic console.

Note: You can type Ctrl+T to return to the `001a01-L1>` prompt to view the virtual LEDs.

3.4.4 Virtual LED Values and Error Messages

Table 3-6 contains the virtual LED values and error messages that the power-on diagnostics return.

Table 3-6 Virtual LED Values and Error Messages

LED Value	Test or Routine	Functional Unit	Failure Message	Failing Component
0x0	In slave loop; 0x00/0x45=okay; solid 0x00=possible hang	Power-on discovery	n/a	n/a
0x01	Initialize the processor, FPRs, and COP0 registers	Power-on discovery	n/a	n/a
0x02	Test processor COP1 registers	Power-on discovery	n/a	n/a
0x03	Switch to mapped mode	Power-on discovery	n/a	n/a
0x04	Test processor primary instruction cache	Power-on discovery	n/a	n/a
0x05	Test processor primary data cache	Power-on discovery	n/a	n/a
0x06	Test secondary cache	Power-on discovery	n/a	n/a
0x07	Flush all caches	Power-on discovery	n/a	n/a
0x0A	Invalidate processor primary instruction cache	Power-on discovery	n/a	n/a
0x0B	Invalidate processor primary data cache	Power-on discovery	n/a	n/a
0x0C	Invalidate secondary cache	Power-on discovery	n/a	n/a
0x0D	Successfully jumped to the main () function	Power-on discovery	n/a	n/a
0x0E	About to increase PROM access speed	Power-on discovery	n/a	n/a
0x0F	Increase PROM access speed	Power-on discovery	n/a	n/a
0x10	Not used	Power-on discovery	n/a	n/a
0x18	UART putc timed out	Power-on discovery	n/a	n/a
0x1D	About to initialize selected UART	Power-on discovery	n/a	n/a
0x1E	Done initializing selected UART	Power-on discovery	n/a	n/a

Table 3-6 (continued) Virtual LED Values and Error Messages

LED Value	Test or Routine	Functional Unit	Failure Message	Failing Component
0x21	About to enter POD mode, C portion	Power-on discovery	n/a	n/a
0x22	Just about to enter POD prompt loop	Power-on discovery	n/a	n/a
0x23	About to enter POD mode, assembler portion	Power-on discovery	n/a	n/a
0x24	Performing local arbitration (CPU A/B)	Power-on discovery	n/a	n/a
0x28	About to perform first local barrier	Power-on discovery	n/a	n/a
0x2A	About to configure Dex mode stack and data	Power-on discovery	n/a	n/a
0x2B	Reached main ()	Power-on discovery	n/a	n/a
0x31	In entry because of nonmaskable interrupt (NMI)	Power-on discovery	n/a	n/a
0x35	About to initialize hub real-time counter	Power-on discovery	n/a	n/a
0x36	Done initializing hub real-time counter	Power-on discovery	n/a	n/a
0x38	First local barrier succeeded	Power-on discovery	n/a	n/a
0x3C	About to jump to UALIAS space	Power-on discovery	n/a	n/a
0x3D	Jumped to UALIAS space	Power-on discovery	n/a	n/a
0x3E	About to jump to cached space	Power-on discovery	n/a	n/a
0x3F	Jumped to cached space	Power-on discovery	n/a	n/a
0x40	About to test stack area of memory	Power-on discovery	n/a	n/a
0x41	Done testing stack area of memory	Power-on discovery	n/a	n/a
0x45	Slave loop; 0x00/0x45=okay; solid 0x45=possible hang	Power-on discovery	n/a	n/a
0x46	Received launch interrupt	Power-on discovery	n/a	n/a

Table 3-6 (continued) Virtual LED Values and Error Messages

LED Value	Test or Routine	Functional Unit	Failure Message	Failing Component
0x47	Calling launched function	Power-on discovery	n/a	n/a
0x48	Launched function returned	Power-on discovery	n/a	n/a
0x49	Not used	Power-on discovery	n/a	n/a
0x4A	About to initialize hub MD and SIMM controls	Power-on discovery	n/a	n/a
0x4B	About to probe and configure memory size	Power-on discovery	n/a	n/a
0x4F	About to discover hub I/O	Power-on discovery	n/a	n/a
0x51	About to write router configuration information into KLCONFIG	Power-on discovery	n/a	n/a
0x52	About to initialize I/O section of hub	Power-on discovery	n/a	n/a
0x53	About to probe I/O section for console	Power-on discovery	n/a	n/a
0x54	Console probing completed	Power-on discovery	n/a	n/a
0x55	Global master in PROM	Power-on discovery	n/a	n/a
0x56	Done initializing I/O section of hub	Power-on discovery	n/a	n/a
0x57	Reset error state saved	Power-on discovery	n/a	n/a
0x58	Hub error registers cleared	Power-on discovery	n/a	n/a
0x59	Hub error checking enabled	Power-on discovery	n/a	n/a
0x5A	Done discovering hub I/O	Power-on discovery	n/a	n/a
0x5B	About to initialize NMI handler area	Power-on discovery	n/a	n/a
0x5C	About to test hub interrupts	Power-on discovery	n/a	n/a
0x5D	About to perform early reset of hub I/O section	Power-on discovery	n/a	n/a
0x5E	CPU came out of reset	Power-on discovery	n/a	n/a

Table 3-6 (continued) Virtual LED Values and Error Messages

LED Value	Test or Routine	Functional Unit	Failure Message	Failing Component
0x5F	Going to reset I^2C	Power-on discovery	n/a	n/a
0x60	Finished resetting I^2C	Power-on discovery	n/a	n/a
0x70	Running bist on bank 0	Power-on discovery	n/a	n/a
0x71	Running bist on bank 1	Power-on discovery	n/a	n/a
0x72	Running bist on bank 2	Power-on discovery	n/a	n/a
0x73	Running bist on bank 3	Power-on discovery	n/a	n/a
0x74	Running bist on bank 4	Power-on discovery	n/a	n/a
0x75	Running bist on bank 5	Power-on discovery	n/a	n/a
0x76	Running bist on bank 6	Power-on discovery	n/a	n/a
0x77	Running bist on bank 7	Power-on discovery	n/a	n/a
0x81	CP1 failed	Processor	n/a	PIMM
0x82	Restart master unable to load IO7 PROM	Processor	n/a	PIMM
0x83	Primary instruction cache test	Primary instruction cache	`RSLT cache_test_i FAIL`	PIMM
0x84	Primary data cache test	Primary data cache	`RSLT cache_test_d FAIL`	PIMM
0x85	Secondary cache test	Secondary instruction cache	`RSLT cache_test_s FAIL`	PIMM
0x86	CPU was disabled		n/a	PIMM
0x87	Real-time counter failure		n/a	PIMM
0x8F	OS requested LEDs		n/a	n/a
0x91	Hub local test failed	Hub	n/a	IP34 motherboard
0x93	Some node not premium memory	Hub	n/a	IP34 motherboard
0x97	Main() returned	Hub	n/a	IP34 motherboard
0x98	No local memory	Hub	n/a	IP34 motherboard
0x99	I^2C cannot happen error	Hub	n/a	IP34 motherboard
0x9A	CPU disabled by environment variable	Hub	n/a	IP34 motherboard
0x9B	Memory download failure	Hub	n/a	IP34 motherboard
0x9C	Cannot set core debug register	Hub	n/a	IP34 motherboard

Table 3-6 (continued) Virtual LED Values and Error Messages

LED Value	Test or Routine	Functional Unit	Failure Message	Failing Component
0x9E	Hub KLCONFIG failed	Hub	n/a	IP34 motherboard
0x9F	Router KLCONFIG failed	Hub	n/a	IP34 motherboard
0xA0	Hub I/O init failed	Hub	n/a	IP34 motherboard
0xA1	Node KLCONFIG failed	Hub	n/a	IP34 motherboard
0xA4	Hub chip failed l/abist	Hub	n/a	IP34 motherboard
0xA6	Waiting for reset to go	Hub	n/a	IP34 motherboard
0xA7	LLP failed after reset	Hub	n/a	IP34 motherboard
0xA8	LLP never up after reset	Hub	n/a	IP34 motherboard
0xA9	No good local memory	Hub	n/a	IP34 motherboard
0xAB	Network discovery failed	Hub	n/a	IP34 motherboard
0xAC	NASID calculation failed	Hub	n/a	IP34 motherboard
0xAD	Route calculation failed	Hub	n/a	IP34 motherboard
0xAE	Route distribution failed	Hub	n/a	IP34 motherboard
0xAF	NASID distribution failed	Hub	n/a	IP34 motherboard
0xB0	Master assigned no NASID	Hub	n/a	IP34 motherboard
0xB1	NASID arbitration failed	Hub	n/a	IP34 motherboard
0xB4	Error copying mode bits	Hub	n/a	IP34 motherboard
0xB5	Error calculating backplane frequency	Hub	n/a	IP34 motherboard
n/a	Hub send-data error test	Hub	`RSLT hub_send_data_error FAIL`	IP34 motherboard
n/a	Hub error-detection test	Hub	`RSLT test_hub_error_detection FAIL`	IP34 motherboard
n/a	GClk logic test	Hub	`RSLT rt_clock_test FAIL`	IP34 motherboard
n/a	BTE test	Hub	`RSLT hub_bte_diag FAIL`	IP34 motherboard
n/a	Get chipid test	Hub	`RSLT get_chipid FAIL`	IP34 motherboard
n/a	PROM checksum test	PROM	`RSLT prom_checksum FAIL`	IP34 motherboard

Table 3-6 (continued) Virtual LED Values and Error Messages

LED Value	Test or Routine	Functional Unit	Failure Message	Failing Component
0x98 or 0xa9	Memory configuration test	Memory	`No Memory Found, nasid` *node_number*`: disabling bank` *bank_number*	DIMM
n/a	Memory test	Memory	`RSLT memtest FAIL` or `MEMORY FAILURE`	DIMM
n/a	XBOW test	XBOW portion of Xbridge	`RSLT xbow_sanity FAIL`	IP34 motherboard or PIMM
n/a	Bridge test	Bridge portion of Xbridge	`RSLT bridge_sanity FAIL`	IP34 motherboard
n/a	PCICONFIG test	PCI configuration space	`RSLT io_config_space FAIL`	IP34 motherboard
n/a	PCI bus test	PCI configuration space	`RSLT pcibus_sanity FAIL`	IP34 motherboard
n/a	Serial programmed I/O test	Serial port	`RSLT serial_pio FAIL`	IP34 motherboard
n/a	Serial DMA test	Serial port	`RSLT serial_dma FAIL`	IP34 motherboard
n/a	SCSI tests	SCSI SSRAM test	`RSLT scsi_ram FAIL`	IP34 motherboard
n/a		SCSI DMA test	`RSLT scsi_dma FAIL`	IP34 motherboard
n/a		SCSI self-test	`RSLT scsi_controller`	IP34 motherboard
n/a	Keyboard and mouse test	Keyboard and mouse	`Failed pckm diagnostics`	Keyboard, mouse, or IP34 motherboard
n/a	Graphics tests	Graphics board	`Failed graphics diagnostics`	Graphics board
n/a	Ethernet tests	Comprehensive Ethernet test	`RSLT enet_all FAIL`	IP34 motherboard
n/a		Ethernet SSRAM test	`RSLT enet_ssram FAIL`	IP34 motherboard
n/a		TX clock test	`RSLT enet_tx_clk FAIL`	IP34 motherboard
n/a		PHY register test	`RSLT enet_phy_reg FAIL`	IP34 motherboard
n/a		IOC3 internal loopback DMA test	`RSLT enet_ioc3_loop FAIL`	IP34 motherboard
n/a		PHY chip internal loopback test	`RSLT enet_phy_loop FAIL`	IP34 motherboard
n/a		Twister chip internal loopback test	`RSLT enet_tw_loop FAIL`	IP34 motherboard
n/a	Ethernet tests (cont.)	External loopback DMA test	`RSLT enet_ext_loop FAIL`	IP34 motherboard

Table 3-6 (continued) Virtual LED Values and Error Messages

LED Value	Test or Routine	Functional Unit	Failure Message	Failing Component
n/a		External loopback DMA (10 Mb/s) test	`RSLT enet_10_ext_loop FAIL`	IP34 motherboard
n/a		External loopback DMA (100 Mb/s) test	`RSLT enet_100_ext_loop FAIL`	IP34 motherboard
n/a		Xtalk stress test	`RSLT enet_xtalk_stress FAIL`	IP34 motherboard

3.4.5 diag_rc Values

Several of the power-on diagnostics return diagnostic result code (diag_rc) values in the error messages that they generate. These messages use the following format:

`RSLT` *test_name* `FAIL  diag_rc = xxx`

Example:

```
RSLT scsi_dma       FAIL                diag_rc = 67
```

Table 3-7 lists all of the diag_rc values that the power-on diagnostics return.

Table 3-7 diag_rc Values

Value	Error Description	Failing Component
0	No error occurred. The test completed successfully.	None
	L1 and L2 Cache Error Codes	
1	Failed dcache1 data test way0	PIMM
2	Failed dcache1 data test way1	PIMM
3	Failed dcache1 address test way0	PIMM
4	Failed dcache1 address test way1	PIMM
5	Failed scache1 data test way0	PIMM
6	Failed scache1 data test way1	PIMM
7	Failed scache1 address test way0	PIMM
8	Failed scache1 address test way1	PIMM
9	Failed icache data test way0	PIMM
10	Failed icache data test way1	PIMM
11	Failed icache address test way0	PIMM

Table 3-7 (continued) diag_rc Values

Value	Error Description	Failing Component
12	Failed icache address test way1	PIMM
13	dcache test hung	PIMM
14	scache test hung	PIMM
15	icache test hung	PIMM
16	Cache initialization	PIMM
17	dcache tag ram test	PIMM
18	scache tag ram test way0	PIMM
19	scache tag ram test way1	PIMM
20	FPU scache tag ram test	PIMM
21	Starting dcache test	PIMM
22	Starting icache test	PIMM
23	Starting scache test	PIMM
24	Invalidate icache and dcache	PIMM
25	Invalidate icache and dcache	PIMM
26	Generic dcache test failure	PIMM
27	Generic icache test failure	PIMM
	CPU Error Codes	
31	The CPU is disabled.	PIMM
32	The CPU failed in **earlyinit**.	PIMM
33	The CPU took an unexpected exception.	PIMM
34	Coprocessor 1 is dead.	PIMM
35	CPU died before or during local arbitration.	PIMM
36	The CPU and PROM mode bits do not match.	PIMM
	Memory Error Codes	
40	The checksum in the PROM is bad.	IP34 motherboard
41	The copy of the PROM in memory is bad.	IP34 motherboard
42	The copy of PROM to memory failed.	IP34 motherboard
43	The memory test (memtest) failed for one or more banks.	PIMM or DIMM
44	The directory memory test (dirtest) failed for one or more banks.	PIMM or DIMM
45	Memory is disabled by the user.	PIMM
46	Bank 0 is unusable.	PIMM

Table 3-7 (continued) diag_rc Values

Value	Error Description	Failing Component
47	Unsupported 256-MB bank on IP27	PIMM
48	Failed memory bist for some bank	PIMM
	BaseIO PROM Error Codes	
50	XBOW sanity on the IO7 failed.	IP34 motherboard
51	Bridge sanity on the IO7 failed.	IP34 motherboard
52	The IOC3 base address could not be found.	IP34 motherboard
53	The configuration space on the IO7 failed.	IP34 motherboard
54	The PCI bus test failed.	IP34 motherboard
55	Characters already exist in port A.	IP34 motherboard
56	Characters already exist in port B.	IP34 motherboard
57	Unable to transmit characters on port A.	IP34 motherboard
58	Unable to transmit characters on port B.	IP34 motherboard
59	Miscompares occurred on port A.	IP34 motherboard
60	Miscompares occurred on port B.	IP34 motherboard
61	Miscompares occurred on port A in DMA mode.	IP34 motherboard
62	Miscompares occurred on port B in DMA mode.	IP34 motherboard
63	A SCSI mailbox failure occurred.	IP34 motherboard
64	Unable to load words into the SCSI SSRAM.	IP34 motherboard
65	Unable to read words out of the SCSI SSRAM.	IP34 motherboard
66	Miscompares occurred on the SCSI SSRAM.	IP34 motherboard
67	Unable to load words into the SCSI SSRAM.	IP34 motherboard
68	Unable to dump words from the SCSI SSRAM.	IP34 motherboard
69	DMA miscompares from the SCSI SSRAM occurred.	IP34 motherboard
70	Unable to load the self-test firmware.	IP34 motherboard
71	Unable to execute the SCSI self-test.	IP34 motherboard
72	The SCSI self-test failed with miscompares.	IP34 motherboard
73	A timeout occurred during the keyboard self-test.	IP34 motherboard or keyboard
74	A timeout occurred during the mouse self-test.	IP34 motherboard or mouse
80	An XBOW exception occurred.	IP34 motherboard
81	A Bridge exception occurred.	IP34 motherboard

Table 3-7 (continued) diag_rc Values

Value	Error Description	Failing Component
82	An IO8 configuration exception occurred.	IP34 motherboard
83	A PCI bus exception occurred.	IP34 motherboard
84	A serial programmed I/O (PIO) exception occurred.	IP34 motherboard
85	A serial DMA exception occurred.	IP34 motherboard
86	A SCSI RAM exception occurred.	IP34 motherboard
87	A SCSI DMA exception occurred.	IP34 motherboard
88	A SCSI controller exception occurred.	IP34 motherboard
101	ENET_EXCEPTION	IP34 motherboard
102	ENET_SSRAM_FAIL	IP34 motherboard
103	ENET_PHYREG_FAIL	IP34 motherboard
104	ENET_PHY_R_BUSY_TO	IP34 motherboard
105	ENET_PHY_W_BUSY_TO	IP34 motherboard
106	ENET_PHY_RESET_TO	IP34 motherboard
107	ENET_NO_TXCLK	IP34 motherboard
108	ENET_LOOP_ILL_PARM	IP34 motherboard
109	ENET_LOOP_MALLOC	IP34 motherboard
110	ENET_EMCR_RST_TO	IP34 motherboard
111	ENET_EMCR_RXEN_TO	IP34 motherboard
112	ENET_LOOP_AN_TO	IP34 motherboard
113	ENET_LOOP_AN_ABLE	IP34 motherboard
114	ENET_LOOP_AN_MODE	IP34 motherboard
115	ENET_LOOP_DMA_TO1	IP34 motherboard
116	ENET_LOOP_DMA_TO2	IP34 motherboard
117	ENET_LOOP_DMA_TO3	IP34 motherboard
118	ENET_LOOP_BAD_RUPT1	IP34 motherboard
119	ENET_LOOP_BAD_RUPT2	IP34 motherboard
120	ENET_LOOP_BAD_RUPT3	IP34 motherboard
121	ENET_LOOP_BAD_RUPT4	IP34 motherboard
122	ENET_LOOP_NO_RUPT	IP34 motherboard
123	ENET_LOOP_BAD_W1	IP34 motherboard
124	ENET_LOOP_BAD_STAT	IP34 motherboard

Table 3-7 (continued) diag_rc Values

Value	Error Description	Failing Component
125	ENET_LOOP_BAD_DATA1	IP34 motherboard
126	ENET_LOOP_BAD_DATA2	IP34 motherboard
127	ENET_NIC_FAIL	IP34 motherboard
131	The `info` graphics test failed.	Graphics board
132	The `xtalk` graphics test failed.	Graphics board
133	The `dcb` graphics test failed.	Graphics board
134	The `buzz` graphics test failed.	Graphics board
135	The `analog` graphics test failed.	Graphics board
150	There were too many link errors in the hub NI.	IP34 motherboard
151	There were too many miscellaneous errors from the router in the hub NI.	IP34 motherboard
152	The hub NI did not detect forced errors.	IP34 motherboard
153	The hub chip failed the lbist test.	IP34 motherboard
154	The hub chip failed the abist test.	IP34 motherboard
155	The LLP interface on the hub chip is down.	IP34 motherboard
156	The hub could not launch discovery.	IP34 motherboard
157	The interrupt test failed for the hub chip.	IP34 motherboard
158	An NI error occurred in the kernel.	IP34 motherboard
159	The BTE test failed for the hub chip.	IP34 motherboard
160	The BTE for the hub chip hung while diagnostics were running.	IP34 motherboard
161	The hub has a real-time clock error.	IP34 motherboard
162	A reset propagation error occurred.	IP34 motherboard
	Generic Error Codes	
240	No CPU is available.	Anywhere
241	Returning to POD mode.	Anywhere
242 - 247	A PROM panic occurred.	Anywhere
248 - 252	An NMI occurred.	Anywhere
	PS/2 Error Codes	
253	The IOC3 pckm register test failed.	IP34 motherboard, keyboard, or mouse

Table 3-7 (continued) diag_rc Values

Value	Error Description	Failing Component
254	Two of the same PS/2 devices cannot exist.	IP34 motherboard, keyboard, or mouse
255	No PS/2 devices were found.	IP34 motherboard, keyboard, or mouse

3.4.6 System Configuration and Diagnostics Summary

After the power-on diagnostics and discovery process complete, the BaseIO PROM code displays the system configuration and diagnostics summary:

```
**** System Configuration and Diagnostics Summary ****
CONFIG:
         No. of NODEs enabled    = 1
         No. of NODEs disabled   = 0
         No. of CPUs enabled     = 1
         No. of CPUs disabled    = 0
         Mem enabled             = 1024 MB
         Mem disabled            = 0 MB
         No. of RTRs enabled     = 0
         No. of RTRs disabled    = 0

DIAG RESULTS:
         ALL DIAGS PASSED.
**** End System Configuration and Diagnostics Summary ****
```

Note: The `System Configuration and Diagnostics Summary` information appears only when the *console* PROM environment variable is set to `d`. To see this information, you must connect an external system running the L3 system controller software to the diagnostic (L1) port or connect a terminal to the L1 system controller DB-9 connector on the motherboard.

After the boot process prints this screen, control passes to the PROM monitor, which displays the `System Maintenance Menu` or autoboots the system, depending on how the PROM monitor is configured.

3.4.7 Example Output

```
Starting PROM Boot process


IP35 PROM SGI Version 6.101 TEST built 10:39:43 AM Nov 15, 2001
Running in DDR mode
Testing/Initializing memory ..............            DONE
Copying PROM code to memory ..............            DONE
Discovering local IO .....................            DONE
Discovering NUMAlink connectivity .........
Local hub NUMAlink is down.
*** Local network link down
DONE
Found 1 objects (1 hubs, 0 routers) in 5894 usec
Waiting for peers to complete discovery....            DONE
No other nodes present; becoming global master
Global master is /hw/rack/001/bay/01
Checking partitioning information .........            DONE
No other nodes present; becoming partition master
Loading BASEIO prom ......................
INFO: console subchannel changed:  001a01 console
DONE

BASEIO PROM Monitor SGI Version 6.101 TEST built 10:36:15 AM Nov 15, 2001 (BE64)
1 CPUs on 1 nodes found.
Automatic update of PROM environment disabled
Installing PROM Device drivers ............
WARNING: AddChild revision(3) different from system revision(0).

Walking SCSI Adapter 0, (pci id 1)
1+ Device Vendor Product: SEAGATE ST318404LW
2- 3- 4- 5- 6- 7- 8- 9- 10- 11- 12- 13- 14- 15- = 1 device(s)


Walking SCSI Adapter 1, (pci id 1)
1- 2- 3- 4+ Device Vendor Product: TOSHIBA DVD-ROM SD-M1401
5- 6- 7- 8- 9- 10- 11- 12- 13- 14- 15- = 1 device(s)

Initializing PROM Device drivers ..........            DONE
Checking hardware inventory ...............             DONE

**** System Configuration and Diagnostics Summary ****
CONFIG:
         No. of NODEs enabled    = 1
         No. of NODEs disabled   = 0
         No. of CPUs enabled     = 1
         No. of CPUs disabled    = 0
         Mem enabled             = 512 MB
         Mem disabled            = 0 MB
         No. of RTRs enabled     = 0
         No. of RTRs disabled    = 0

DIAG RESULTS:
         ALL DIAGS PASSED.
**** End System Configuration and Diagnostics Summary ****
```

3.5 IP35 PROM Tests

The IP35 PROM tests check the PIMM, Bedrock, memory, PROM image, PCI slots, and serial port.

Note: The term “IP35 PROM” is used because the power-on diagnostics were originally developed for IP35 systems.

3.5.1 CPU Tests

- Register test
- Primary instruction cache test
- Primary data cache test

3.5.1.1 Register Test

The register test checks for stuck bits in the processor registers. This test uses the following algorithm:

1. Write 0xa5a5 and 0x0000 patterns to the BSR registers.
2. Read the data back from the BSR registers.
3. Compare the data that was read back with the data that was written.

This test runs automatically during the boot process. You cannot run this test from the `POD>` prompt.

The failing LED pattern is 0x81. If this test fails, replace the PIMM.

3.5.1.2 Primary Instruction Cache Test

The primary instruction cache test (`cache_test_i`) writes to and reads from the primary instruction cache.

The primary instruction cache test uses the following algorithm:

1. Write 0x5555 and 0x0000 data patterns to the primary instruction cache.
2. Read the data back from the primary instruction cache.
3. Compare the data that was read with the data that was written.

This test runs automatically during the boot process. You cannot run this test from the `POD>` prompt.

The failing LED pattern is 0x83. The failing diag_rc values are 9, 10, 11, 12, 15, 22, and 27. If this test fails, replace the PIMM.

3.5.1.3 Primary Data Cache Test

The primary data cache test (`cache_test_d`) writes to and reads from the primary data cache.

This test runs automatically during the boot process. You cannot run this test from the `POD>` prompt.

The failing LED pattern is 0x84. The failing diag_rc values are 1, 2, 3, 4, 13, 17, 21, and 26. If this test fails, replace the PIMM.

3.5.2 Secondary Cache Test

The secondary cache test (`cache_test_s`) walks data patterns across the secondary data cache to detect stuck bits in the secondary cache SRAMs. This test uses the following algorithm:

1. Write data patterns (0xaaaa and 0x5555 in normal mode; 0xaaaa, 0x5555, 0x0000, and 0xffff in heavy and manufacturing modes) to the secondary cache SRAMs.
2. Read the data back from the secondary cache SRAMs.
3. Compare the data that was read with the data that was written.

This test runs automatically during the boot process. You cannot run this test from the `POD>` prompt.

The test displays the `RSLT cache_test_s FAIL` message. The failing LED values are 0x85, 0x86, 0x87, 0x88, 0x89, 0x8A, 0x8B, and 0x8C. The failing diag_rc values are 5, 6, 7, 8, 14, 19, 20, 23, and 25. If this test fails, replace the PIMM.

3.5.3 Bedrock Tests

The following Bedrock tests are available:

- Hub interrupt test
- Hub local test
- Hub config test
- Hub UART test
- Get chipid test
- GClk logic test
- BTE test

3.5.3.1 Hub Interrupt Test

The hub interrupt test (`hub_intrpt_diag`) verifies that the processor can correctly receive interrupts. This test checks the hardware in the processor that is dedicated to handling any interrupts or exceptions. It also tests the hub (Bedrock) chip interrupt registers.

This test runs automatically during the boot process. You cannot run this test from the `POD>` prompt.

The test displays the `RSLT hub_int_diag FAIL` failure message. The failing LED pattern is 0x90. The failing diag_rc value is 157. If this test fails, replace the PIMM or IP34 motherboard. (The cause of the failure is most likely the CPU on the PIMM.)

3.5.3.2 Hub Local Test

The hub local test tests the hub (Bedrock).

This test runs automatically during the boot process. You cannot run this test from the `POD>` prompt.

The failing LED value is 0x91. If this test fails, replace the IP34 motherboard.

3.5.3.3 Hub Config Test

The hub config test tests the hub (Bedrock).

This test runs automatically during the boot process. You cannot run this test from the `POD>` prompt.

The failing LED value is 0x92. If this test fails, replace the IP34 motherboard.

3.5.3.4 Hub UART Test

The hub UART test tests the hub (Bedrock).

This test runs automatically during the boot process. You cannot run this test from the `POD>` prompt.

The failing LED value is 0x95. If this test fails, replace the IP34 motherboard.

3.5.3.5 GClk Logic Test

The GClk logic test (`rt_clock_test`) checks the functionality of the hub global clock logic.

This test runs automatically during the boot process. You cannot run this test from the `POD>` prompt.

If a failure occurs, this test prints out a `RSLT rt_clock_test FAIL` failure message.

If this test fails, replace the IP34 motherboard.

3.5.3.6 Block Transfer Engine (BTE) Test

The BTE test (`hub_bte_diag`) checks the BTE logic in the hub (Bedrock) ASIC. This test uses the following algorithm:

1. Program both BTEs on the hub chip.
2. Have both BTEs on the hub chip perform local transfer operations.
3. Compare the results.

In all modes, this test performs a simple block copy transfer and then checks all basic modes of the BTE logic, including basic copy, termination of a copy in progress, zero-fill mode, poison mode, interrupt mode, and notification mode.

This test runs automatically during the boot process. You cannot run this test from the `POD>` prompt.

If a failure occurs, this test prints a `RSLT hub_bte_diag FAIL` message. The failing diag_rc values are 159 and 160. The boot process continues even if this test detects an error.

Before you replace any hardware components, determine whether the memory tests passed or failed. If the memory tests failed, replace the memory DIMMs. If the memory tests did not fail, replace the IP34 motherboard.

3.5.4 PROM Checksum Test

The PROM checksum test (`prom_checksum`) verifies the copy of PROM code in memory. If it fails when it is verifying the copy in memory, this test verifies the copy of the PROM code that is stored in the PROM.

This test uses the following algorithm:

1. Total the number of words in the PROM code.
2. Compare the sum with an expected value.

This test runs automatically during the boot process. You cannot run this test from the `POD>` prompt.

If this test fails, it displays the `RSLT prom_checksum FAIL` failure message. The failing diag_rc values are 40, 41, 42, 46, 242, 243, 244, 245, 246, and 247.

If this test fails, the failing FRU is either the IP34 motherboard or a memory DIMM. Possible causes are a bad PROM, a bad copy of the software image in the PROM, bad bit(s) in memory, a transient error that occurred when the PROM image was copied from the PROM into memory, or a transient error in the processor that occurred when the processor computed the checksum. The most likely cause is one or more bad bits in memory.

3.5.5 Memory Tests

- Memory configuration test
- Backdoor directory space test
- Memory test

3.5.5.1 Memory Configuration Test

The memory configuration test verifies that the system memory configuration is valid.

This test runs automatically during the boot process. You cannot run this test from the `POD>` prompt.

If this test fails, the failing LED values are `0x98` or `0xa9`. The test also prints one of the following messages:

```
No Memory Found
```

`nasid` *node_number*`: disabling bank` *bank_number*

If this test fails, the failing FRU is the DIMM that the test indicates**.** A failure of this test most likely indicates a bad memory configuration. It could also indicate a hardware failure in a hub-to-memory interconnect. The system does not boot if this test fails.

Note: This test always prints the following message:
`Total memory configured:` *number* `MB`.

3.5.6 Backdoor Directory Space Test

The backdoor directory space test (`dirtest`) checks the memory that contains the backdoor directory space.

The directory memory test uses the following algorithm:

1. Clear the error registers.
2. Test the store and load functions. For each doubleword address in the memory range that is being tested, perform the following actions:
   - Perform a store operation with a data pattern that contains all 0x5 values, immediately perform load operations to get the data back, and compare the loaded data with the stored data.
   - Perform a store operation with a data pattern that contains all 0xa values, immediately perform load operations to get the data back, and compare the loaded data with the stored data.

   This test may find problems with RAM cells that take too long to change states.
3. Test memory with alternating 0x5 and 0xa data patterns. (Write the data patterns, read the data back, and compare the data that was read back with the data that was written.)
4. Test memory with alternating 0xa and 0x5 data patterns. (Write the data patterns, read the data back, and compare the data that was read back with the data that was written.)
5. Test memory with alternating 0xc3c3c3c3c3c3c3c3 and 0x3c3c3c3c3c3c3c3c doubleword patterns. (Write the data patterns, read the data back, and compare the data that was read back with the data that was written.)
6. Test memory with alternating 0x3c3c3c3c3c3c3c3c and 0xc3c3c3c3c3c3c3c3 doubleword patterns. (Write the data patterns, read the data back, and compare the data that was read back with the data that was written.)
7. Test memory with random data patterns. (Write a random data pattern, read the data back, and compare the data that was read back with the data that was written.)
8. Test memory with address data patterns. (Write each doubleword with the address of the location being written, read the data back, and compare the data that was read back with the data that was written.)

Memory is left-filled with a data pattern of incrementing doublewords. The `maxerr` command specifies the number of errors that the test should display for each phase before it stops the current phase and continues with the next phase of the test.

This test runs automatically during the boot process, or you can use the `dirtest` command at the POD prompt to run this test. The `dirtest` command tests a range of directory memory. This command uses the following syntax:

```
POD> dirtest start len
```

The *start* variable specifies a memory address that this command uses by masking off the lower 40 bits and translating the value into a corresponding directory address. The *len*

variable specifies the size of the area to test (in bytes). The *start* and *len* variables must be multiples of 4,096 (0x1000).

Note: You should verify that the processor, the system interface bus, and the hub registers are functional before you manually run this test. Enter **`ecc off`** to disable ECC checking before you test directory memory.

The failing diag_rc value for this test is 44. The test also displays one of the following messages:

```
----- MEMORY FAILURE (caught exception node slot slot_number) -----
Address address
CAUSE cause_register_value

-----MEMORY FAILURE (miscompare node slot slot_number) -----
Address address
CAUSE cause_register_value
```

If this test fails, the failing FRU is the DIMM that the test specifies. Failure of this test indicates one of the following hardware failures: a directory DIMM had a soft or hard failure (most likely), a hub-to-directory memory interconnect failed, or the hub failed (least likely).

3.5.6.1 Memory Test

The memory test (`memtest`) checks the memory address and data lines.

The memory test uses the following algorithm:

1. Clear the error registers.
2. Test the store and load functions. For each doubleword address in the memory range that is being tested, perform the following actions:
    1. Perform a store operation with a data pattern that contains all 0x5 values, then perform load operations to get the data back, and then compare the loaded data with the stored data.
    2. Perform a store operation with a data pattern that contains all 0xa values, then perform load operations to get the data back, and then compare the loaded data with the stored data.

    This test may find problems with RAM cells that take too long to change states.
3. Test memory with alternating 0x5 and 0xa data patterns. (Write the data patterns, read the data back, and compare the data that was read back with the data that was written.)
4. Test memory with alternating 0xa and 0x5 data patterns. (Write the data patterns, read the data back, and compare the data that was read back with the data that was written.)
5. Test memory with alternating 0xc3c3c3c3c3c3c3c3 and 0x3c3c3c3c3c3c3c3c doubleword patterns. (Write the data patterns, read the data back, and compare the data that was read back with the data that was written.)
6. Test memory with alternating 0x3c3c3c3c3c3c3c3c and 0xc3c3c3c3c3c3c3c3 doubleword patterns. (Write the data patterns, read the data back, and compare the data that was read back with the data that was written.)
7. Test memory with random data patterns. (Write a random data pattern, read the data back, and compare the data that was read back with the data that was written.)
8. Test memory with address data patterns. (Write each doubleword with the address of the location being written, read the data back, and compare the data that was read back with the data that was written.)

Memory is left-filled with a data pattern of incrementing doublewords. The *maxerr* command specifies the number of errors that the test should display for each phase before it stops the current phase and continues with the next phase of the test.

This test runs automatically during the boot process, or you can use the `memtest` command from the `POD>` prompt to run this test. The `memtest` command tests a range of memory. This command uses the following syntax:

`POD>` **`memtest`** *start len*

The *start* variable specifies the starting address of the memory range; this address can be in cached or uncached memory. The *len* command specifies the size of the range of memory to test (in bytes). *start* and *len* must be multiples of 4,096 (0x1000).

Examples:

```
POD> memtest u:32m 1m
POD> memtest 0x9600000002000000 0x100000
```

These examples are equivalent. These examples test 1 MB of uncached memory space, starting at an offset of 32 MB.

```
POD> memtest c:32m 32m
```

This example tests the second 32 MB of cached memory.

The failing diag_rc value for this test is 43. It also displays `RSLT memtest FAIL` or one of the following messages:

`-----MEMORY FAILURE (caught exception node slot` *slot_number*`) -----`
`Address` *address*
`CAUSE` *cause_register_value*

`-----MEMORY FAILURE (miscompare node slot` *slot_number*`) -----`
`Address` *address*
`Actual` *actual_value*
`Expected` *expected_value*
`Difference` *difference_value*

If this test fails, the failing FRU is the DIMM that the test specifies. A failure of this test indicates a soft or hard failure in a memory chip (most likely), a failing hub chip-to-memory interconnect, or a bad hub chip (least likely).

3.5.7 Xbridge Tests

- XBOW test
- Bridge test

3.5.7.1 XBOW Test

The XBOW test (`xbow_sanity`) checks the XBOW portion of the Xbridge ASIC for valid reset values in the XBOW registers and tests the XBOW interrupt hardware.

Note: If a hub-link error occurs during the boot sequence, this test may not run because the link was not detected.

This test uses the following algorithm:

1. Read the XBOW registers and verify that they contain valid reset values.
2. Create an illegal access by writing to a read-only register, which causes an XBOW interrupt.
3. Verify that the interrupt occurred on the XBOW and the hub.

This test runs automatically during the boot process, or you can run it from the `POD>` prompt. To run this test from the `POD>` prompt, enter the following command:

```
POD> dgxbow [mn | mh | mm] [n nasid]
```

The `m` option specifies the mode in which the test should run. Enter `mn` to run the test in normal mode, enter `mh` to run the test in heavy mode, or enter `mm` to run the test in manufacturing mode. (The default mode is normal mode.)

The `n` option specifies the node ID (NASID) of the node on which the test should run. (The default node is the node from which you entered the command.)

For example, enter the following command to run the XBOW test in heavy mode on node 1.

```
POD> dgxbow mh n1
```

The failing diag_rc values are 50 and 80. The XBOW test also returns the following failure messages:

- `xbow_sanity: Bad xbow id reg value` *value*
- `xbow_sanity: Bad Xbow id Exp:` *expected* `Recv:` *received*
- `xbow_sanity: Reg write Read miscompare. Exp:` *expected*`: Recv:` *received*
- `xbow_sanity: Register access violation not detected.`
- `xbow_sanity: Status Register Clear on Read not working!!`
- `xbow_sanity: Interrupt did not reach PI_INT_PEND0. exp:` *expected* `recv:` *received*
- `RSLT xbow_sanity FAIL`

If this test fails, replace the IP34 motherboard.

3.5.7.2 Bridge Test

The bridge test (`bridge_sanity`) checks the bridge portion of the Xbridge ASIC for valid reset values in the registers and tests the bridge interrupt hardware.

This test uses the following algorithm:

1. Read the bridge registers and verify that they contain valid reset values.
2. Force illegal accesses to the SSRAM to cause bridge interrupts.
3. Verify that the interrupts occurred on both the Xbridge and the hub (Bedrock).

This test runs automatically during the boot process, or you can run it from the `POD>` prompt. To run this test from the `POD>` prompt, enter the following command:

```
POD> dgbrdg [mn | mh | mm] [nnasid] [bbus]
```

The `m` option specifies the mode in which the test should run. Enter `mn` to run the test in normal mode, enter `mh` to run the test in heavy mode, or enter `mm` to run the test in manufacturing mode. (The default mode is normal mode.)

The `n` option specifies the node ID of the node on which the test should run. (The default node is the node from which you entered the command.)

The `b` option specifies the slot number of the board with the Xbridge that you want to test.

For example, to run this test in heavy mode on bus 4 of the Xbridge ASIC enter:

```
POD> dgbrdg mh b4
```

The failing diag_rc values are 51 and 81. This test also returns the following error messages:

- `bridge_sanity: Bad bridge id register value (LSB != 1)`
- `bridge_sanity: Bad bridge id Exp:` *expected* `Recv:` *received*
- `bridge_sanity: PCI bit set to 0 sn status reg.`
- `bridge_sanity: Timeout val reg exp:` *expected* `recv:` *received*.
- `bridge_sanity: Bridge control reg exp:` *expected* `recv:` *received*.
- `bridge_sanity: Bad reset val in dev reg exp:` *expected* `recv:` *received*.
- `bridge_sanity: Interrupt status register value non-zero:` *value*.
- `bridge_sanity: Bridge(wr): Exp:` *expected* `Recv:` *received*.
- `bridge_sanity: Interrupt status reg bit not set: exp:` *expected* `recv:` *received*.
- `bridge_sanity: Interrupt` *interrupt* `did not reach PI_INT_PEND0: exp:` *expected* `recv:` *received*.
- `bridge_sanity: Interrupt` *interrupt* `did not occur.`

This test always returns the following message when the test detects an error:

- `RSLT bridge_sanity FAIL`

If this test fails, replace the IP34 motherboard.

3.5.8 PCI Tests

- PCICONFIG test
- PCI bus test

3.5.8.1 PCICONFIG Test

The PCICONFIG test (`io_config_space`) verifies the PCI configuration space.

This test runs automatically during the boot process, or you can run it from the `POD>` prompt. To run this test from the `POD>` prompt, enter the following command:

`POD>` **`dgconf`** `[mn | mh | mm] [n`*`nasid`*`] [b`*`bus`*`]`

The `m` option specifies the mode in which the test should run. Enter `mn` to run the test in normal mode, enter `mh` to run the test in heavy mode, or enter `mm` to run the test in manufacturing mode. (The default mode is normal mode.)

The `n` option specifies the node ID (NASID) of the node on which the test should run. (The default node is the node from which you entered the command.)

The `b` option specifies the location of the board that you want to test.
(The default bus number is 3, which selects slot IO1.)

For example, enter the following command to run the `dgconf` test in heavy mode on bus 4:

`POD>` **`dgconf mh b4`**

This test returns the following failure messages:

- `io_config_space: IOC3 PCI_SCR reg reset val exp: expected recv: received`
- `io_config_space: IOC3 PCI_ADDR reg reset val exp: expected recv: received`
- `io_config_space: IOC3 PCI_ADDR reg write read error. wrote: value read: value`
- `io_config_space: PCI device` *`device`* `QL command status reg reset val exp: expected recv: received`
- `io_config_space: PCI device` *`device`* `QL class code and revision read as value`
- `io_config_space: PCI device` *`device`* `QL base address reg reset val exp: expected recv: received`
- `io_config_space: PCI device` *`device`* `QL base addr write read error. wrote: value read: value`

This test always returns the following message when the test detects an error:

- `RSLT io_config_space FAIL`

If this test fails, replace the IP34 motherboard.

3.5.8.2 PCI Bus Test

The PCI bus test (`pcibus_sanity`) checks the PCI bus with walking data tests. The test is written with the assumption that the processor, system interface bus, hub (Bedrock), and Xbridge ASIC have been tested. This test uses the following algorithm:

1. Write a walking 1's pattern in the 32-bit Ethernet low-address register.
2. Read the data back.
3. Compare the data that was read back with the data that was written.
4. Write a walking 0's pattern in the 32-bit Ethernet low-address register.
5. Read the data back.
6. Compare the data that was read back with the data that was written.

This test runs automatically during the boot process, or you can run it from the `POD>` prompt. To run this test from the `POD>` prompt, enter the following command:

`POD>` **`dgpci`** `[mn | mh | mm] [n`*`node`*`] [b`*`bus`*`] [p`*`pcinum`*`]`

The `m` option specifies the mode in which the test should run. Enter `mn` to run the test in normal mode, enter `mh` to run the test in heavy mode, or enter `mm` to run the test in manufacturing mode. (The default mode is normal mode.)

The `n` option specifies the node ID of the node on which the test should run. (The default node is the node from which you entered the command.)

The `b` option specifies the I/O slot number of the board that you want to test.

The `p` option specifies the PCI device number of the IOC3 ASIC that you want to test.

For example, enter the following command to run the PCI bus test in normal mode:

`POD>` **`dgpci mn`**

The failing diag_rc values are 54 and 83. The PCI bus test also returns the following failure messages:

- `pcibus_sanity: IOC3 enet hashed low addr reg reset val exp:` *expected* `recv:` *received*
- `pcibus_sanity: PCI walk one test. exp:` *expected* `recv:` *received*
- `pcibus_sanity: PCI walk zero test. exp:` *expected* `recv:` *received*

This test always returns the following message when the test detects an error:

- `RSLT pcibus_sanity FAIL`

If this test fails, replace the IP34 motherboard.

3.5.9 Serial Port Tests

- Serial programmed I/O test
- Serial DMA test

3.5.9.1 Serial Programmed I/O Test

The serial programmed I/O (PIO) test (`serial_pio`) uses the internal loopback features of the SuperIO chip to test the serial PIO functionality. This test uses the following algorithm:

1. Reset the SuperIO chip.
2. Fill the transmit FIFO with data.
3. Read the receive FIFO in loopback mode.
4. Compare the data in the receive FIFO with the data in the transmit FIFO.

This test runs automatically during the boot process, or you can run it from the `POD>` prompt. To run this test from the `POD>` prompt, enter the following command:

`POD>` **`dgspio`** `[mn | mh | mm | mx] [n`*`node`*`] [b`*`bus`*`] [p`*`pcinum`*`]`

Note: Before you can run this test in external loopback mode (by using the `mx` command-line option), you must attach loopback connectors to the serial ports that you will test.

The `m` option specifies the mode in which the test should run. Enter `mn` to run the test in normal mode, enter `mh` to run the test in heavy mode, enter `mm` to run the test in manufacturing mode, or enter `mx` to run the test in external loopback mode. (The default mode is normal mode.)

The `n` option specifies the node ID of the node on which the test should run. (The default node is the node from which you entered the command.)

The `b` option specifies the I/O slot number of the board that you want to test.

The `p` option specifies the PCI device number of the IOC3 ASIC that you want to test.

For example, enter the following command to run the serial PIO test in heavy mode.

`POD>` **`dgspio mh`**

The failing diag_rc values are 55, 56, 57, 58, 59, 60, and 84. This test also returns the following failure messages:

- `serial_pio: UARTA read timed out on LSR<Data Ready>`
- `serial_pio: UARTA error. Chars in FIFO after reset .....`
- `serial_pio: UARTA sent:` *value* `recv:` *value*

  Note: This message also appears if you do not install external loopback plugs on the ports before you run the test in external loopback mode.

- `serial_pio failed with` *number* `errors`

This test always returns the following message when the test detects an error:

- `RSLT serial_pio FAIL`

If this test fails, replace the IP34 motherboard.

3.5.9.2 Serial DMA Test

The serial DMA test (`serial_dma`) verifies that serial DMA can be performed.

This test uses the following algorithm:

1. Set up the ring buffers.
2. Set up DMA in loopback mode.
3. Perform DMA operation through UART channel A.
4. Verify the results.

This test runs automatically during the boot process, or you can run it from the `POD>` prompt. To run this test from the `POD>` prompt, enter the following command:

```
POD> dgsdma [mn | mh | mm | mx] [nnode] [bbus] [ppcinum]
```

Note: The CPU must be in Cac mode to run this test. If you attempt to run this test from Dex mode or Unc mode, the test displays a message that indicates you should enter Cac mode (enter **`go cac`**) before you run the test.

The `m` option specifies the mode in which the test should run. Enter `mn` to run the test in normal mode, enter `mh` to run the test in heavy mode, enter `mm` to run the test in manufacturing mode, or enter `mx` to run the test in external loopback mode. (The default mode is normal mode.)

Note: Before you can run this test in external loopback mode (by using the `mx` command-line option), you must attach loopback connectors to the serial ports that you will test.

The `n` option specifies the node ID of the node on which the test should run. (The default node is the node from which you entered the command.)

The `b` option specifies the bus number of the I/O device that you want to test.

The `p` option specifies the PCI device number of the IOC3 ASIC that you want to test.

For example, enter the following command to run the serial DMA test in heavy mode.

```
POD> dgsdma mh b4
```

The failing diag_rc values are 61, 62, and 85. This test also returns the following failure messages:

- `serial_dma: PCI: could not get the ioc3 base addr`
- `serial_dma: Serial DMA exp:` *expected* `recv:` *received*

  Note: This message also appears if you do not install external loopback plugs on the ports before you run the test in external loopback mode.

- `serial_dma: Timeout on uartA dma ....`

This test always returns the following message when the test detects an error:

- `RSLT serial_dma FAIL`

If this test fails, replace the IP34 motherboard.

3.6 BaseIO PROM Tests

The BaseIO PROM tests verify the SCSI controller, keyboard and mouse, graphics, and Ethernet hardware.

3.6.1 SCSI Tests

The SCSI test verifies the SCSI controller for the system boot disk.

The SCSI driver in the BaseIO PROM automatically calls the controller tests when the boot sequence installs the motherboard-based SCSI devices. You can also manually run them from the `>>` prompt, which you can access by selecting the `Enter Command Monitor` option from the `System Maintenance Menu`. You cannot run these tests from the `POD>` prompt.

When the tests run automatically, they use the mode that is set by the virtual debug switch settings. When you run the tests manually, the command-line options that you specify override the mode that is set by the virtual debug switches.

The `diag_fibre` command runs the SCSI controller tests from the command monitor prompt (`>>`) and uses the following syntax:

`diag_fibre [-N` *nasid*`] [-m` *moduleid* `-b` *bus*`] [-p` *pcidev*`] [-h] [-v] [-t` *testname*`] [-r` *count*`| -R` *count*`]`

`-N` *nasid*	Specifies the NASID number of the module to test. This option supersedes the NASID that is indicated by the module and slot values.
`-m` *moduleid*	Specifies the module number of the module to test.
`-b` *bus*	Specifies the bus name.
`-p` *pcidev*	Specifies the PCI device number of the target IOC3.
`-h`	Runs the diagnostic in heavy mode.
`-v`	Runs the diagnostic in verbose mode, which returns detailed status and error messages.
`-t` *testname*	Runs only the specified test. Valid values for *testname* are `ram`, `dma`, and `controller`. (The default is to run all tests.)
`-r` *count*	Repeats the test *count* times. The test stops when it detects a failure.
`-R` *count*	Repeats the test *count* times. The test continues to execute when it detects a failure.

3.6.1.1 SCSI Controller SSRAM Test

The SCSI controller SSRAM test (`scsi_ram`) verifies that data can be loaded into controller SSRAM memory and read back successfully. It also verifies the "mailbox" mechanism before it performs this test. This test uses the following algorithm:

1. Verify the controller "mailbox" mechanism.
2. Write data into the controller SSRAM (one word at a time).
3. Read data back from the controller SSRAM (one word at a time).
4. Compare the data that was read back with the data that was written.

This test runs automatically during the boot process, or you can run it from the `>>` prompt. To run the fibre channel controller SSRAM test from the `>>` prompt, enter the **`diag_fibre -t ram`** command and any other command-line options that you want to use. For example:

`>>` **`diag_fibre -t ram`** `[-N` *nasid*`] [-m` *moduleid* `-b` *bus*`] [-p` *pcidev*`] [-h] [-v] [-t` *testname*`][-r` *count* `|` `-R` *count*`]`

If this test fails, the failing diag_rc values are 63, 64, 65, 66, and 86. This test also returns the following failure messages:

- `scsi_ram: SCSI mailbox error. erp:` *expected* `recv:` *received*
- `scsi_ram: QL: wrote:` *write_value* `read` *read_value* `at addr:` *address*

This test always returns the following message when the test detects an error:

- `RSLT scsi_ram FAIL`

If this test fails, replace the IP34 motherboard.

3.6.1.2 SCSI Controller DMA Test

The SCSI controller DMA test (`scsi_dma`) verifies that the controller DMA mechanism is functional. This test uses the following algorithm:

1. Use a SCSI channel controller DMA operation to write data to the SCSI channel controller SSRAM with data.
2. Use a SCSI channel controller DMA operation to read the data that is in SCSI fibre channel controller SSRAM.
3. Compare the data that was read with the data that was written.

This test runs automatically during the boot process, or you can run it from the `>>` prompt. To run the SCSI DMA test from the `>>` prompt, enter the **`diag_fibre -t dma`** command and any other command-line options that you want to use. For example:

`>>` **`diag_fibre -t dma`** `[-N` *nasid*`] [-m` *moduleid* `-b` *bus*`] [-p` *pcidev*`] [-h] [-v] [-t` *testname*`][-r` *count* `|` `-R` *count*`]`

Note: The CPU from which you run this test must be in Cac mode. If you attempt to run this test from a CPU that is in Dex mode or Unc mode, the test displays a message that indicates you should enter Cac mode (type **`go cac`**) before you run the test.

If this test fails, the failing diag_rc values are 67, 68. 69, and 87. This test also returns the following failure messages:

- `scsi_dma: LOAD SSRAM command failed`
- `scsi_dma: DUMP SSRAM command failed`
- `scsi_dma: DMA failure exp` *expected* `recv` *received* `at` *location*

This test always returns the following message when the test detects an error:

- `RSLT scsi_dma FAIL`

If this test fails, replace the IP34 motherboard.

3.6.1.3 SCSI Controller Self-test

The SCSI controller self-test (`scsi_controller`) uses the controller self-test firmware to test the controller. This test uses the following algorithm:

1. Load the SCSI controller self-test firmware.
2. Perform the controller self-test. The self-test tests the RISC unit, the SXP FIFOs, and the fibre SCSI bus.

This test runs automatically during the boot process, or you can run it from the `>>` prompt. To run the fibre channel controller self-test from the `>>` prompt, enter the **`diag_fibre -t controller`** command and any other command-line options that you want to use. For example:

`>>` **`diag_fibre -t controller`** `[-N` *nasid*`] [-m` *moduleid* `-b` *bus*`] [-p` *pcidev*`] [-h] [-v] [-t` *testname*`][-r` *count* `|` `-R` *count*`]`

If this test fails, the failing diag_rc values are 70, 71, 72, and 88. This test also returns the following failure messages:

- `scsi_controller: EXECUTE FIRMWARE Command Failed.`
- `scsi_controller: write data failure at` *count* `word`
- `scsi_controller: Bad status val ==>` *value* `....`

This test always returns the following message when the test detects an error:

- `RSLT scsi_controller FAIL`

If this test fails, replace the IP34 motherboard.

3.6.2 Keyboard and Mouse Test

The keyboard and mouse test (`pckm`) verifies the keyboard, mouse, and PS/2 ports.

This test runs automatically during the boot process, or you can run it from the `POD>` prompt. To run this test from the `POD>` prompt, enter the following command:

```
POD> dgpckm [mn | mm]
```

Note: This is the only BaseIO PROM-based test that runs from the `POD>` prompt and not the `>>` prompt.

The `m` option specifies the mode in which the test should run. Enter `mn` to run the test in normal mode or enter `mm` to run the test in manufacturing mode. (The default mode is normal mode.)

The failing diag_rc values are 52, 73, 74, 253, 254, and 255. This test also returns the following failure messages:

- `pckm: could not get the ioc3 based addr`
- `pckm: IOC3 pckm register test failed`
- `Timeout during keyboard self-test`
- `LEDS test : keyboard doesn't answer`
- `Keyboard failed self-test`
- `Timeout during mouse self-test`
- `Mouse failed self-test`
- `Cannot have two of the SAME PS/2 devices`
- `No PS/2 devices found`

This test always returns the following messages when the test detects an error:

- `Failed pckm diagnostics`
- `RSLT pckm FAIL`

If this test fails, replace the keyboard, mouse, or IP34 motherboard.

Note: Before you replace any hardware, swap the keyboard and mouse connections and run the test again. If the motherboard is failing, the same port fails with a different component connected. If the keyboard or mouse is failing, the error message follows the component to the other port.

3.6.3 Graphics Tests

The graphics driver in the BaseIO PROM automatically calls the graphics tests when the boot sequence installs the graphics device. You can also run the graphics tests from the `>>` prompt, which you can access by selecting the `Enter Command Monitor` option from the `System Maintenance Menu`. You cannot run these tests from the `POD>` prompt.

The graphics tests include:

- `info`, which reads basic configuration information from the graphics board and compares it to expected results. If this test fails, the failing diag_rc value is 131.
- `xtalk`, which verifies that the Xtalk link is operating properly by checking the link status of the Xbow for the graphics board (port D). Then, it writes walking bit patterns to Xtalk-related registers on the graphics card, reads data back from the registers, and compares the results. If this test fails, the failing diag_rc value is 132.
- `buzz`, which writes walking bit patterns to registers in the Buzz ASIC that are set during initialization of the graphics hardware. Then, it reads the data in the registers and compares the data that was written with the data that was read. If this test fails, the failing diag_rc value is 134.

  Note: Only the `info`, `xtalk`, and `buzz` tests run during the boot process. The `dcb` and `analog` tests require initialization of the graphics board before they can run. You can manually run the `dcb` and `analog` tests from the >> prompt.

- `dcb`, which tests the display control bus (DCB) that connects the Buzz ASIC and pixel blaster and jammer (PB and J) ASIC. This test writes walking bit patterns to the DCB, reads data back, and compares the results. If this test fails, the failing diag_rc value is 133.

  Note: The `dcb` test modifies the PB and J ASIC configuration information, which causes the screen to go blank. The screen remains blank for several seconds while the graphics hardware reinitializes after the test completes.

- `analog`, which tests the three analog lines (PBJ_ANAL_RED_H, PBJ_ANAL_GRN_H, and PBJ_ANAL_BLUE_H) that connect the PB and J ASIC to the monitor by using the red, green, and blue comparator output bits of the DAC control register. You must attach a monitor to the workstation to use this test. If this test fails, the failing diag_rc value is 135.

When the graphics tests run automatically, they use the mode that is set by the virtual debug switches. When you run the tests manually, the command-line options that you specify override the mode that is set by the virtual debug switch settings.

The `diag_gfx` command runs the graphics tests from the `>>` prompt and uses the following syntax:

`diag_gfx` `[-t <`*`testname`*`>] [-r <`*`count`*`> | -R <`*`count`*`>] [-h] [-v]`

`-t` *testname*	Runs only the specified test. (Valid values for *testname* are `info`, `xtalk`. `buzz`, `dcb`, and `analog`.)
`-r` *count*	Repeats the test *count* times. The test stops when it detects a failure.
`-R` *count*	Repeats the test *count* times. The test continues executing when it detects a failure.
`-h`	Runs the diagnostic in heavy mode.
`-v`	Runs the diagnostic in verbose mode, which returns detailed status and error messages.

If one of the graphics tests fails, the failing diag_rc value is 131, 132, 133, 134, or 135. These tests also generate the following failure messages:

- `Odyssey configuration is of undefined type`
- `Failed xtalk tests`
- `Failed Buzz registers tests`
- `Failed Analog Output Tests`
- `Failed DCB bus tests`

These tests always return the following messages when they detect an error:

- `RSLT gfx FAIL`
- `Failed gfx diagnostics`

If any of the graphics tests fail, replace the graphics board.

3.6.4 Ethernet Tests

The Ethernet driver in the BaseIO PROM automatically calls the Ethernet tests when the boot sequence installs the Ethernet device. You can also run the Ethernet tests from the `>>` prompt, which you can access by selecting the `Enter Command Monitor` option from the `System Maintenance Menu`. You cannot run these tests from the `POD>` prompt.

When the tests run automatically, they use the mode that is set by the virtual debug switches. When you run the tests manually, the command-line options that you specify override the mode that is set by the virtual debug switch settings.

The `diag_enet` command runs the Ethernet tests from the `>>` prompt and uses the following syntax:

`diag_enet` `[-N <`*nasid*`>] [-m <`*moduleid*`> <-b` *bus*`>] [-p <`*pcidev*`>] [-h] [-e|-x] [-v] [-t <`*testname*`>] [-r <`*count*`>] [-R <`*count*`>]`

`-N` *nasid*	Specifies the NASID number of the module to test. This option supersedes the NASID that is indicated by the *moduleid* and *bus* values.
`-m` *moduleid*	Specifies the module number of the module to test.
`-b` *bus*	Specifies the bus name.
`-p` *pcidev*	Specifies the PCI device number of the target IOC3.
`-h`	Runs the diagnostic in heavy mode.
`-e or -x`	Runs the diagnostic in external loopback mode.
`-v`	Runs the diagnostic in verbose mode, which returns detailed status and error messages.
`-t` *testname*	Runs only the specified test. Valid values for testname are `ssram`, `txclk`, `phyreg`, `nic`, `ioc3_loop`, `phy_loop`, `tw_loop`, `ext_loop`, `ext_loop_10`, `ext_loop_100`, and `xtalk`. (The default is to run all tests.)
`-r` *count*	Repeats the test *count* times. The test stops when it detects a failure.
`-R` *count*	Repeats the test *count* times. The test continues executing when it detects a failure.

The Ethernet diagnostics are written with the assumption that the following hardware has been tested and is functional:

- Non-Ethernet-specific logic on the IOC3 ASIC
- PCI bus address and data
- Xbridge ASIC
- Xbridge ASIC, and all IP34 motherboard components

Note: When the boot sequence runs the Ethernet diagnostics, all of these components have already been tested by other POD diagnostics. However, the actions that the Ethernet diagnostics perform stress these components harder than the other diagnostics, so it is possible that a previously undetected failure in one of these components could cause the Ethernet diagnostics to fail.

The Ethernet diagnostic tests are as follows:

- Comprehensive Ethernet test
- Ethernet SSRAM test
- TX clock test
- PHY register test
- IOC3 internal loopback test
- PHY chip internal loopback test
- Twister chip internal loopback test
- External loopback DMA test
- External loopback DMA (10 Mb/s) test
- External loopback DMA (100 Mb/s) test
- Xtalk stress test

3.6.4.1 Comprehensive Ethernet Test

The comprehensive Ethernet test (`enet_all`) runs all of the Ethernet tests in an order that optimizes fault isolation.

Note: If a failure of one of the diagnostics implies failures in subsequent diagnostics, this test does not run the diagnostics that will fail.

This test uses the following algorithm to control the execution of the other Ethernet tests:

1. Run the SSRAM test.
2. Run the TX clock test.
3. Run the PHY register test.
4. If the SSRAM test passed, run the IOC3 internal loopback DMA test.
5. If the IOC3 internal loopback DMA test passes and the TX clock test passes and the PHY register test passes, run the PHY internal loopback DMA test.
6. If the PHY internal loopback DMA test passes, run the twister chip internal loopback DMA test.
7. If the PHY internal loopback DMA test passes and external loopback node is selected, run the external loopback DMA test.
8. If the PHY internal loopback DMA test passes, external loopback DMA test passes (or was not run), and heavy mode or external loopback mode was selected; run the Xtalk stress test.

For detailed information about each of the tests, refer to the corresponding description later in this document.

This test runs automatically during the boot process, or you can run it from the >> prompt. To manually run this test from the >> prompt, enter the **`diag_enet`** [*test options*] command and any other command-line options that you want to use.

Note: Ensure that the following hardware is functional before you run this test:

- Non-Ethernet-specific logic on the IOC3 ASIC
- PCI bus address/data
- Xbridge ASIC
- Motherboard components

If this test fails, the failing diag_rc value is 101. This test also generates the following failure messages:

- `enet_all: SSRAM test failed.`

  The SSRAM test failed. Refer to Section 3.6.4.2, “Ethernet SSRAM Test,” for more information about the failure messages that the SSRAM test returns.

- `enet_all: Ethernet PHY chip register test failed.`

  The PHY register test failed. Refer to Section 3.6.4.3, “PHY Register Test,” for more information about the failure messages that the PHY register test returns.

- `enet_all: IOC3 internal loopback DMA test failed.`

  The IOC3 internal loopback DMA test failed. Refer to Section 3.6.4.5, “IOC3 Internal Loopback Test,” for more information about the failure messages that the IOC3 internal loopback test returns.

- `enet_all: Ethernet PHY chip internal loopback DMA test failed.`

  The PHY chip internal loopback DMA test failed. Refer to Section 3.6.4.6, “PHY Chip Internal Loopback Test,” for more information about the failure messages that the PHY chip internal loopback DMA test returns.

- `enet_all: Ethernet Twister chip internal loopback DMA test failed.`

  The twister chip internal loopback DMA test failed. Refer to Section 3.6.4.7, “Twister Chip Internal Loopback Test,” for more information about the failure messages that the twister chip internal loopback DMA test returns.

- `enet_all: Ethernet external loopback DMA test failed.`

  The external loopback DMA test failed. Refer to Section 3.6.4.8, “External Loopback DMA Test,” for more information about the failure messages that the external loopback DMA test returns.

- `enet_all: Xtalk stress test failed.`

  The Xtalk stress test failed. Refer to Section 3.6.4.11, “Xtalk Stress Test,” for more information about the failure messages that the Xtalk stress test returns.

3.6.4.2 Ethernet SSRAM Test

The Ethernet SSRAM test (`enet_ssram`) checks the 128-KB Ethernet packet-buffering SSRAM, the SSRAM's interface to the IOC3 ASIC, and the parity generation and checking logic in the IOC3. This test uses the following algorithm:

1. Perform an address walk test:
    1. Write a test pattern to addresses 0, 1, 2, 4, 8, ...
    2. Read the pattern back and verify that the data that was read back matches the data that was written.
2. Perform a parity test:
    1. Write a value with good parity = 0 to memory location 0.
    2. Read the value back, verify that the value that was read back equals the value that was written, and verify that the error bit did not set.
    3. Write a value with good parity = 1 to memory location 0.
    4. Read the value back, verify that the value that was read back equals the written value, and verify that the error bit did not set.
    5. Write a value with bad parity = 0 to memory location 0.
    6. Read the value back, verify that the value that was read back equals the value that was written, and verify that the error bit set.
    7. Write a value with bad parity = 1 to memory location 0.
    8. Read the value back, verify that the value that was read back equals the value that was written, and verify that the error bit set.
3. In heavy and manufacturing modes only, perform a data pattern test:
    1. Write a test pattern to all memory addresses, alternating between low and high addresses.
    2. Read the data back and verify that the data that was read back matches the data that was written.

In normal mode, the address walk test and parity tests are the only tests that run.
In heavy and manufacturing modes, all tests run.

This test runs automatically during the boot process, or you can run it from the `>>` prompt. To run the Ethernet SSRAM test from the `>>` prompt, enter:

`>>` **`diag_enet -t ssram`** `[-N <`*`nasid`*`>] [-m <`*`moduleid`*`> <-b `*`bus`*`>] [-p <`*`pcidev`*`>] [-h] [-e|-x] [-v] [-r <`*`count`*`>] [-R <`*`count`*`>]`

Refer to "Ethernet Tests" on page 3-52 for descriptions of all available options.

Note: Ensure that the following hardware is functional before you run this test:

- Non-Ethernet-specific logic on the IOC3 ASIC
- PCI bus address / data
- Xbridge ASIC
- Motherboard components

If this test fails, the failing diag_rc value is 102. This test also generates the following failure messages:

- `enet_ssram: could not get the ioc3 base addr`
- `enet_ssram: ssram address walk error. Ex:` *expected_value* `Recv:` *received_value*
- `enet_ssram: ssram parity test error. Exp:` *expected_value* `Recv:` *received_value*
- `enet_ssram: IOC3 ssram test failed...`
- `enet_ssram: **Data miscompare(A)** offset:` *offset_value* `exp:` *expected_value* `recv:` *received_value*

This test always returns the following message when the test detects an error:

- `RSLT enet_ssram FAIL`

If this test fails, replace the IP34 motherboard.

3.6.4.3 PHY Register Test

The PHY register test (`enet_phy_reg`) checks the reset values of registers on the PHY chip. This test also verifies the capability to read and write selected bits in selected registers on the PHY chip; this test checks only the bits and registers that the Ethernet driver and the other Ethernet diagnostics use. This test uses the following algorithm:

1. Verify the presence of the PHY chip.
2. If the PHY chip is present, perform the following actions:
    1. Reset the PHY chip with a software reset.
    2. Verify the reset values of all registers in the PHY chip and generate failure messages for any miscompares that are detected.
    3. Read and write data to selected bits in registers 0, 23, and 24. Compare the data that is read back to the written data and generate failure messages for any miscompares that are detected.

This test runs automatically during the boot process, or you can run it from the `>>` prompt. To run the PHY register test from the `>>` prompt, enter:

```
>> diag_enet -t phyreg [-N <nasid>] [-m <moduleid> <-b bus>]
[-p <pcidev>] [-h] [-e|-x] [-v] [-r <count>] [-R <count>]
```

Refer to “Ethernet Tests” on page 3-52 for descriptions of all available options.

Note: Ensure that the following hardware is functional before you run this test:

- Non-Ethernet-specific logic on the IOC3 ASIC
- PCI bus address / data
- Xbridge ASIC
- Motherboard components

If this test fails, the failing diag_rc values are 103, 104, 105, and 106. This test also generates the following failure messages:

- `enet_phy_reg: Read of PHY register 0 failed (returned 0xffff).`
- `enet_phy_reg: Timed out waiting for PHY to rest.`
- `enet_phy_reg:  Timed out on MICR_BUSY before read.`
  `enet_phy_reg:  Timed out on MICR_BUSY during read.`
- `enet_phy_reg: Timed out on MICR_BUSY before write.`
  `enet_phy_reg: Timed out on MICR_BUSY during write.`
- `enet_phy_reg: Warning - PHY Identifier register #1 value` *value* `is not supported by this diag!`
  `enet_phy_reg: Warning - PHY Identifier register #2 value` *value* `is not supported by this diag!`
  `enet_phy_reg: Warning - PHY revision register value` *value* `is not supported by this diag!`
- `enet_phy_reg: Warning - unexpected Reg` *register* `reset value, exp:` *expected* `mask:` *mask_value* `got:` *received*

- `enet_phy_reg: Reg 0 value wrong, exp:` *expected* `mask:` *mask_value* `got:` *received*
  `enet_phy_reg: Reg 23 value wrong, exp:` *expected* `mask:` *mask_value* `got:` *received*
  `enet_phy_reg: Reg 24 value wrong, exp:` *expected* `mask:` *mask_value* `got:` *received*

This test always returns the following message when the test detects an error:

- `RSLT enet_phy_reg FAIL`

If this test fails, replace the IP34 motherboard.

3.6.4.4 TX Clock Test

The TX clock test (`enet_tx_clk`) verifies the presence of the TX clock that is sent from the PHY chip to the IOC3 ASIC. This test reads the IOC3 ETCSR register and checks the NO_TX_CLK bit. This test uses the following algorithm:

1. Read the IOC3 ETCSR register.
2. Check the NO_TX_CLK bit.

This test runs the same way in normal and heavy modes. Manufacturing mode returns more status information than normal and heavy modes.

This test runs automatically during the boot process, or you can run it from the `>>` prompt. To run the TX clock test from the `>>` prompt, enter the **`diag_enet -t txclk`** command and any other command-line options that you want to use. For example:

`>>` **`diag_enet -t txclk`** `[-N <`*nasid*`>] [-m <`*moduleid*`> <-b` *bus*`>] [-p <`*pcidev*`>] [-h] [-e|-x] [-v] [-r <`*count*`>] [-R <`*count*`>]`

Refer to “Ethernet Tests” on page 3-52 for descriptions of all available options.

Note: Ensure that the following hardware is functional before you run this test:

- Non-Ethernet-specific logic on the IOC3 ASIC
- PCI bus address / data
- Xbridge ASIC
- Xtalk interface between the bridge and XBOW

This test displays the following failure message:

```
enet_tx_clk: IOC3 is not seeing the PHY TX clock.
```

This test always returns the following message when the test detects an error:

- `RSLT enet_tx_clk FAIL`

If this test fails, replace the IP34 motherboard.

3.6.4.5 IOC3 Internal Loopback Test

The IOC3 internal loopback test (`enet_ioc3_loop`) verifies Ethernet DMA by using the internal loopback features of the IOC3 ASIC. This test uses the following algorithm:

1. Allocate memory for the TX descriptor ring and write the base address to the ETBR registers in the IOC3.
2. Allocate memory for the TX data buffers and write their addresses to the descriptor ring entries and to the command and buffer count fields.
3. Generate test data and write it to the TX buffers. (The test pattern is designed to stress PCI SSO.)
4. Allocate memory for the RX buffer pointer ring and write the base address to the ERBR registers on the IOC3.
5. Allocate memory for the RX buffers.
6. Enable error interrupts on the Xbridge ASIC.
7. Reset the IOC3 Ethernet DMA engines, initialize the IOC3 registers for IOC3 loopback DMA, and enable the TX_EMPTY interrupt.
8. Write to the ring pointers in a way that causes the DMA to occur in one burst of 10 packets (in normal mode) or one burst of 495 packets (in heavy and manufacturing modes) and then stop.
9. Write to the EMCR register on the IOC3 to enable DMA.
10. Execute a polling loop to wait for the IOC3 to post TX_EMPTY in the EISR. During the polling loop, monitor for unexpected error conditions. Time out if this takes too long.
11. Check the Xbridge ASIC interrupt status register (ISR) to verify that the interrupt went from the IOC3 to the bridge.
12. Wait for the valid bit to set in the RX buffer for the last packet. While waiting for this to occur, the test monitors for unexpected error conditions.
13. Time out if it takes too long for the valid bit to set in the RX buffer for the last packet.
14. Check all received packets: Check the status field for each packet and also verify that the data matches the data that was transmitted.
15. Clean up: Free the allocated memory, restore interrupt enables, and reset the PHY chip.

In normal mode, this test transmits and receives 10 packets; each packet contains 301 bytes of data. In heavy and manufacturing modes, this test transmits and receives 495 packets; each packet contains 1,501 bytes of data.

This test runs automatically during the boot process, or you can run it from the >> prompt. To run the IOC3 internal loopback test from the >> prompt, enter:

```
>> diag_enet -t ioc3_loop [-N <nasid>] [-m <moduleid> <-b bus>]
[-p <pcidev>] [-h] [-e|-x] [-v] [-r <count>] [-R <count>]
```

Refer to “Ethernet Tests” on page 3-52 for descriptions of all available options.

Note: Ensure that the following hardware is functional before you run this test:

- Non-Ethernet-specific logic on the IOC3 ASIC
- PCI bus address/data
- Xbridge ASIC
- Motherboard components

If this test fails, the failing diag_rc value is 52. This test also returns the following failure messages:

- `enet_ioc3_loop: Timed out on EMCR_ARB_DIAG_IDLE trying to reset enet.`
- `enet_ioc3_loop: Unexpected EISR condition while waiting for TX_EMPTY (`*`string1`*`: `*`string2`*`).`
  `enet_ioc3_loop:  EISR value =` *`e_value`*
  `enet_ioc3_loop:  Elapsed time =` *`time`* `us`
- `enet_ioc3_loop: Timed out waiting for TX_EMPTY interrupt (`*`string`*`: `*`string`*`).`
- `enet_ioc3_loop: Bridge ISR bit` *`value`* `did not set after TX_EMPTY interrupt (`*`string`*`: `*`string`*`).`
  `enet_ioc3_loop:  ==> Bridge ISR =` *`value`*
- `enet_ioc3_loop: Timed out polling RX buffer valid bit. (`*`string`*`: `*`string`*`).`
  `enet_ioc3_loop:` *`value`* `of` *`value`* `packets were received.`
- `enet_ioc3_loop: Unexpected EISR condition while waiting for RX buffer valid bit (`*`string`*`: `*`string`*`)`
  `enet_ioc3_loop: EISR value =` *`e_value`*
- `enet_ioc3_loop: RX data miscompare on packet` *`value`* `first word.`
  `enet_ioc3_loop:  ==> exp:` *`expected`* `recv:` *`received`* `(`*`string`*`: `*`string`*`).`
- `enet_ioc3_loop: RX data miscompare on packet` *`value`* `status word.`
  `enet_ioc3_loop:  ==> exp:` *`expected`* `recv:` *`received`* `(`*`string`*`: `*`string`*`).`
- `enet_ioc3_loop: RX data miscompare on packet` *`value`* `DW` *`value.`*
  `enet_ioc3_loop:  ==> exp:` *`expected`* `recv:` *`received`* `(`*`string`*`: `*`string`*`).`
- `enet_ioc3_loop: RX data miscompare on packet` *`value`* `byte` *`value`*`.`
  `enet_ioc3_loop:  ==> exp:` *`expected`* `recv:` *`received`* `(`*`string`*`: `*`string`*`).`

This test always returns the following message when the test detects an error:

- `RSLT enet_ioc3_loop FAIL`

If this test fails, replace the IP34 motherboard.

3.6.4.6 PHY Chip Internal Loopback Test

The PHY chip loopback test (`enet_phy_loop`) checks Ethernet DMA by using the internal loopback features of the PHY chip. The PHY chip loopback DMA runs at 10 Mb/s and also at 100 Mb/s. This test uses the following algorithm twice (once for 10-Mb/s testing and then again for 100-Mb/s testing):

1. Allocate memory for the TX descriptor ring and write the base address to the ETBR registers on the IOC3.
2. Allocate memory for the TX data buffers and write their addresses to the descriptor ring entries and to the command and buffer count fields.
3. Generate test data and write it to the TX buffers. (The test pattern is designed to stress PCI SSO.)
4. Allocate memory for the RX buffer pointer ring and write the base address to the ERBR registers on the IOC3.
5. Allocate memory for the RX buffers.
6. Enable error interrupts on the Xbridge ASIC.
7. Reset the IOC3 Ethernet DMA engines, initialize the IOC3 registers for PHY loopback DMA, and enable the TX_EMPTY interrupt.
8. Write to the ring pointers in a way that causes the DMA to occur in one burst of 10 packets (in normal mode) or one burst of 495 packets (in heavy and manufacturing modes) and then stop.
9. Reset the PHY chip and initialize it for internal loopback at 10 Mb/s (during first test sequence) or 100 Mb/s (during second test sequence).
10. Write to the EMCR register on the IOC3 to enable DMA.
11. Wait for the IOC3 to post TX_EMPTY in the EISR. During the polling loop, monitor for unexpected error conditions. Time out if this takes too long.
12. Check the Xbridge ASIC interrupt status register to verify that the interrupt went from the IOC3 to the bridge.
13. Wait for the valid bit to set in the RX buffer for the last packet. While waiting for this to occur, the test monitors for unexpected error conditions.
14. Time out if it takes too long for the valid bit to set in the RX buffer for the last packet.
15. Check all received packets: Check the status field for each packet and also verify that the data matches the data that was transmitted.
16. Clean up: free the allocated memory, restore interrupt enables, and reset the PHY chip.

In normal mode, this test transmits and receives 10 packets; each packet contains 301 bytes of data. In heavy and manufacturing modes, this test transmits and receives 495 packets; each packet contains 1,501 bytes of data.

This test runs automatically during the boot process, or you can run it from the >> prompt. To run the PHY chip internal loopback test from the >> prompt, enter:

>> **diag_enet -t phy_loop** [-N <*nasid*>] [-m <*moduleid*> <-b *bus*>] [-p <*pcidev*>] [-h] [-e|-x] [-v] [-r <*count*>] [-R <*count*>]

Refer to "Ethernet Tests" on page 3-52 for descriptions of all available options.

Note: Ensure that the following hardware is functional before you run this test:

- Non-Ethernet-specific logic on the IOC3 ASIC
- PCI bus address / data
- Xbridge ASIC
- Motherboard components

Note: Also ensure that the following Ethernet tests pass before you run this test:

- IOC3 internal loopback test
- PHY register test

The PHY chip internal loopback test returns the same messages as the IOC3 internal loopback test. If the IOC3 internal loopback test passed, any failures that are listed as "difficult to isolate" are most likely to be caused by a fault in the IOC3 chip to PHY chip wiring, the PHY chip, or the IOC3 ASIC.

This test also returns the following failure message:

- enet_phy_loop: Unexpected EISR condition while waiting for TX_EMPTY (*string*: *string*).
  enet_phy_loop: EISR value = *e_value*
  enet_phy_loop: Elapsed time = *value* us

This test always returns the following message when the test detects an error:

- RSLT enet_phy_loop FAIL

If this test fails, replace the IP34 motherboard.

3.6.4.7 Twister Chip Internal Loopback Test

The twister chip internal loopback test (`enet_tw_loop`) verifies Ethernet DMA by using the internal loopback features of the twister chip. The twister chip loopback DMA runs at 100 Mb/s.

This test uses the following algorithm:

1. Allocate memory for the TX descriptor ring and write the base address to the ETBR registers on the IOC3.
2. Allocate memory for the TX data buffers and write their addresses to the descriptor ring entries and to the command and buffer count fields.
3. Generate test data and write it to the TX buffers. (The test pattern is designed to stress PCI SSO.)
4. Allocate memory for the RX buffer pointer ring and write the base address to the ERBR registers on the IOC3.
5. Allocate memory for the RX buffers.
6. Enable error interrupts on the Xbridge ASIC.
7. Reset the IOC3 Ethernet DMA engines, initialize the IOC3 registers for twister loopback DMA, and enable the TX_EMPTY interrupt.
8. Write to the ring pointers in a way that causes the DMA to occur in one burst of 10 packets (in normal mode) or one burst of 495 packets (in heavy and manufacturing modes) and then stop.
9. Reset the PHY chip and initialize it for twister loopback at 100 Mb/s.
10. Write to the EMCR register on the IOC3 to enable DMA.
11. Wait for the IOC3 to post TX_EMPTY in the EISR. During the polling loop, monitor for unexpected error conditions. Time out if this takes too long.
12. Check the Xbridge ASIC ISR to verify that the interrupt went from the IOC3 to the Xbridge.
13. Wait for the valid bit to set in the RX buffer for the last packet. While the test waits for this to occur, it monitors for unexpected error conditions.
14. Time out if it takes too long for the valid bit to set in the RX buffer for the last packet.
15. Check all received packets: Check the status field for each packet and also verify that the data matches the data that was transmitted.
16. Clean up: free the allocated memory, restore interrupt enables, and reset the PHY chip.

In normal mode, this test transmits and receives 10 packets; each packet contains 301 bytes of data. In heavy and manufacturing modes, this test transmits and receives 495 packets; each packet contains 1,501 bytes of data.

This test runs automatically during the boot process, or you can run it from the >> prompt. To run the twister chip internal loopback test from the >> prompt, enter:

```
>> diag_enet -t tw_loop [-N <nasid>] [-m <moduleid> <-b bus>] [-p <pcidev>]
[-h] [-e|-x] [-v] [-r <count>] [-R <count>]
```

Refer to “Ethernet Tests” on page 3-52 for descriptions of all available options.

Note: Ensure that the PHY chip internal loopback test runs without errors before you run this test, and ensure that the following hardware is functional before you run this test:

- Non-Ethernet-specific logic on the IOC3 ASIC
- PCI bus address / data
- Xbridge ASIC
- Motherboard components

The twister chip internal loopback test returns the same messages that the PHY chip internal loopback test returns. Failures that are listed as “difficult to isolate” are most likely to be caused by faulty wiring between the PHY chip and the twister chip.

This test always returns the following message when the test detects an error:

- `RSLT enet_tw_loop FAIL`

If this test fails, replace the IP34 motherboard.

3.6.4.8 External Loopback DMA Test

The external loopback DMA test (`enet_ext_loop`) verifies Ethernet DMA by using an external loopback cable or connector. This test verifies the autonegotiation feature and verifies the DMA at 10 Mb/s and 100 Mb/s. This test uses the following algorithm twice (once at 10 Mb/s and then again at 100 Mb/s):

1. Allocate memory for the TX descriptor ring and write the base address to the ETBR registers on the IOC3.
2. Allocate memory for the TX data buffers and write their addresses to the descriptor ring entries and to the command and buffer count fields.
3. Generate test data and write it to the TX buffers. (The test pattern is designed to stress PCI SSO.)
4. Allocate memory for the RX buffer pointer ring and write the base address to the ERBR registers on the IOC3.
5. Allocate memory for the RX buffers.
6. Enable error interrupts on the Xbridge ASIC.
7. Reset the IOC3 Ethernet DMA engines, initialize the IOC3 registers for external loopback DMA, and enable the TX_EMPTY interrupt.
8. Write to the ring pointers in a way that causes the DMA to occur in one burst of 10 packets (in normal mode) or one burst of 495 packets (in heavy and manufacturing modes) and then stop.
9. Reset the PHY chip.
10. Test the autonegotiation features of the hardware.
11. Write to the EMCR register on the IOC3 to enable DMA.
12. Wait for the IOC3 to post TX_EMPTY in the EISR. During the polling loop, monitor for unexpected error conditions. Time out if this takes too long.
13. Check the Xbridge ASIC ISR to verify that the interrupt went from the IOC3 to the Xbridge.
14. Wait for the valid bit to set in the RX buffer for the last packet. While the test waits for this to occur, it monitors for unexpected error conditions.
15. Time out if it takes too long for the valid bit to set in the RX buffer for the last packet.
16. Check all received packets: Check the status field for each packet and also verify that the data matches the data that was transmitted.
17. Clean up: free the allocated memory, restore interrupt enables, and reset the PHY chip.

In normal mode, this test transmits and receives 10 packets; each packet contains 301 bytes of data. In heavy mode, this test transmits and receives 495 packets; each packet contains 1,501 bytes of data. This test does not run in manufacturing mode.

This test does not run during the boot sequence. It runs only from the command line. To run this test from the >> prompt, enter:

```
>> diag_enet -t ext_loop [-N <nasid>] [-m <moduleid> <-b bus>] [-p <pcidev>]
[-h] [-e|-x] [-v] [-r <count>] [-R <count>]
```

Refer to "Ethernet Tests" on page 3-52 for descriptions of all available options.

Note: Ensure that the twister chip internal loopback test runs without errors, and ensure that the following hardware is functional before you run this test:

- Non-Ethernet-specific logic on the IOC3 ASIC
- PCI bus address / data
- Xbridge ASIC
- Motherboard components

This test returns the same failure information as the twister loopback test. If the twister loopback test passed, then failures that were "difficult to isolate" are most likely caused by one of the following conditions:

- The connections between the PHY and the magnetics (for autonegotiation or 10 Mb / s DMA test).
- The connections between the twister chip and the magnetics (for 100 Mb / s DMA test).
- The RJ45 connector.
- The loopback cable / connector.

If the loopback cable and connector are good, the failing FRU is the IP34 motherboard. The following messages, which indicate that the autonegotiation test failed, are returned only with the external loopback test and indicate a failure of the autonegotiation test:

- `enet_ext_loop: Phy did not complete auto-negotiation with itself via external loopback cable.`
- `enet_ext_loop: PHY reg 6 LP_AN_ABLE bit not set after auto-negotiation.`
  `==>  Reg 0 =` *value*
  `==>  Reg 4 =` *value*
  `==>  Reg 5 =` *value*
  `==>  Reg 6 =` *value*
  `*** Are you sure there is an external loopback? ***`
- `enet_ext_loop: PHY reg 5 mode support bits (9:5) do not match reg 4 mode support bits after autonegotiation. Reg 4:` *value* `Reg 5:` *value*

This test always returns the following message when the test detects an error:

- `RSLT enet_ext_loop FAIL`

If this test fails, replace the IP34 motherboard.

3.6.4.9 External Loopback DMA (10 Mb/s) Test

The external loopback DMA (10 Mb/s) test (`enet_10_ext_loop`) is a version of the external loopback DMA test; this version has the following differences: it does not test autonegotiation and it runs only at 10 Mb/s. This test uses the following algorithm:

1. Allocate memory for the TX descriptor ring and write the base address to the ETBR registers on the IOC3.
2. Allocate memory for the TX data buffers and write their addresses to the descriptor ring entries and to the command and buffer count fields.
3. Generate test data and write it to the TX buffers. (The test pattern is designed to stress PCI SSO.)
4. Allocate memory for the RX buffer pointer ring and write the base address to the ERBR registers on the IOC3.
5. Allocate memory for the RX buffers.
6. Enable error interrupts on the Xbridge ASIC.
7. Reset the IOC3 Ethernet DMA engines, initialize the IOC3 registers for external loopback DMA, and enable the TX_EMPTY interrupt.
8. Write to the ring pointers in a way that causes the DMA to occur in one burst of 10 packets (in normal mode) or one burst of 495 packets (in heavy and manufacturing modes) and then stop.
9. Reset the PHY chip.
10. Write to the EMCR register on the IOC3 to enable DMA.
11. Wait for the IOC3 to post TX_EMPTY in the EISR. During the polling loop, monitor for unexpected error conditions. Time out if this takes too long.
12. Check the Xbridge ASIC ISR to verify that the interrupt went from the IOC3 to the bridge.
13. Wait for the valid bit to set in the RX buffer for the last packet. While the test waits for this to occur, it monitors for unexpected error conditions.
14. Time out if it takes too long for the valid bit to set in the RX buffer for the last packet.
15. Check all received packets: Check the status field for each packet and also verify that the data matches the data that was transmitted.
16. Clean up: free the allocated memory, restore interrupt enables, and reset the PHY chip.

In normal mode, this test transmits and receives 10 packets; each packet contains 301 bytes of data. In heavy mode, this test transmits and receives 495 packets; each packet contains 1,501 bytes of data. This test does not run in manufacturing mode.

This test does not run during the boot sequence. It runs only from the command line. To run this test from the >> prompt, enter:

```
>> diag_enet -t ext_loop_10 [-N <nasid>] [-m <moduleid> <-b bus>]
[-p <pcidev>] [-h] [-e|-x] [-v] [-r <count>] [-R <count>]
```

Refer to “Ethernet Tests” on page 3-52 for descriptions of all available options.

Note: Ensure that the following hardware is functional before you run this test:

- Non-Ethernet-specific logic on the IOC3 ASIC
- PCI bus address / data
- Xbridge ASIC
- Motherboard components

Note: Also ensure that the twister chip internal loopback test runs without errors before you run this test. Install a loopback cable or connector before you run this test.

This test returns the same failure messages as the external loopback DMA test.

This test always returns the following message when the test detects an error:

- `RSLT enet_10_ext_loop FAIL`

If this test fails, replace the IP34 motherboard.

3.6.4.10 External Loopback DMA (100 Mb/s) Test

The external loopback DMA (100 Mb/s) test (`enet_100_ext_loop`) is a version of the external loopback DMA test; the differences are that it does not test autonegotiation and it runs only at 100 Mb/s. This test uses the following algorithm:

1. Allocate memory for the TX descriptor ring and write the base address to the ETBR registers on the IOC3.
2. Allocate memory for the TX data buffers and write their addresses to the descriptor ring entries and to the command and buffer count fields.
3. Generate test data and write it to the TX buffers. (The test pattern is designed to stress PCI SSO.)
4. Allocate memory for the RX buffer pointer ring and write the base address to the ERBR registers on the IOC3.
5. Allocate memory for the RX buffers.
6. Enable error interrupts on the Xbridge ASIC.
7. Reset the IOC3 Ethernet DMA engines, initialize the IOC3 registers for external loopback DMA, and enable the TX_EMPTY interrupt.
8. Write to the ring pointers in a way that causes the DMA to occur in one burst of 10 packets (in normal mode) or one burst of 495 packets (in heavy and manufacturing modes) and then stop.
9. Reset the PHY chip.
10. Write to the EMCR register on the IOC3 to enable DMA.
11. Wait for the IOC3 to post TX_EMPTY in the EISR. During the polling loop, monitor for unexpected error conditions. Time out if this takes too long.
12. Check the Xbridge ASIC ISR to verify that the interrupt went from the IOC3 to the bridge.
13. Wait for the valid bit to set in the RX buffer for the last packet. While the test waits for this to occur, it monitors for unexpected error conditions.
14. Time out if it takes too long for the valid bit to set in the RX buffer for the last packet.
15. Check all received packets: Check the status field for each packet and also verify that the data matches the data that was transmitted.
16. Clean up: free the allocated memory, restore interrupt enables, and reset the PHY chip.

In normal mode, this test transmits and receives 10 packets; each packet contains 301 bytes of data. In heavy mode, this test transmits and receives 495 packets; each packet contains 1,501 bytes of data. This test does not run in manufacturing mode.

This test does not run during the boot sequence. It runs only from the command line. To run this test from the >> prompt, enter:

```
>> diag_enet -t ext_loop_100 [-N <nasid>] [-m <moduleid> <-b bus>]
[-p <pcidev>] [-h] [-e|-x] [-v] [-r <count>] [-R <count>]
```

Refer to “Ethernet Tests” on page 3-52 for descriptions of all available options.

Note: Ensure that the following hardware is functional before you run this test:

- Non-Ethernet-specific logic on the IOC3 ASIC
- PCI bus address / data
- Xbridge ASIC
- Motherboard components

Note: Also ensure that the twister chip internal loopback test runs without errors before you run this test. You should install a loopback cable or connector before you run this test.

This test returns the same failure messages as the external loopback DMA test.

This test always returns the following message when the test detects an error:

- `RSLT enet_100_ext_loop FAIL`

If this test fails, replace the IP34 motherboard.

3.6.4.11 Xtalk Stress Test

The Xtalk stress test (`enet_xtalk_stress`) is a 100-Mb/s DMA test that uses a continuous stream of DMA for 1 to 5 seconds, instead of a single burst of 10 or 495 packets like the other DMA tests. This test does not check the received data, but it does check the IOC3 and Xbridge interrupt registers for errors. This test uses a data pattern that is designed to stress SSO on the LLP links in the Xtalk path. This test uses the following algorithm:

1. Allocate memory for the TX descriptor ring and write the base address to the ETBR registers on the IOC3.
2. Allocate memory for the TX data buffers and write their addresses to the descriptor ring entries and to the command and buffer count fields.
3. Generate test data and write it to the TX buffers. (The test pattern is designed to stress LLP.)
4. Allocate memory for the RX buffer pointer ring and write the base address to the ERBR registers on the IOC3.
5. Allocate memory for the RX buffers.
6. Enable error interrupts on the Xbridge ASIC.
7. Reset the IOC3 Ethernet DMA engines, initialize the IOC3 registers for twister loopback DMA, and enable the TX_EMPTY interrupt.
8. Write to the ring pointers in a way that causes the DMA to occur in one burst of 10 packets (in normal mode) or one burst of 495 packets (in heavy and manufacturing modes) and then stop.
9. Reset the PHY chip and initialize it for twister loopback at 100 Mb/s.
10. Write to the EMCR register on the IOC3 to enable DMA.
11. Check for errors in the IOC3 EISR and the bridge ISR and chase the consume pointers. (In normal mode, repeat this step for 1 second. In heavy and manufacturing modes, repeat this step for 5 seconds.)
12. Wait for the IOC3 to post TX_EMPTY in the EISR. During the polling loop, monitor for unexpected error conditions. Time out if this takes too long.
13. Check the Xbridge ASIC ISR to verify that the interrupt went from the IOC3 to the bridge.
14. Wait for the valid bit to set in the RX buffer for the last packet. While the test waits for this to occur, it monitors for unexpected error conditions.
15. Time out if it takes too long for the valid bit to set in the RX buffer for the last packet.
16. Check all received packets: Check the status field for each packet and also verify that the data matches the data that was transmitted.
17. Clean up: free the allocated memory, restore interrupt enables, and reset the PHY chip.

In normal mode, this test performs 1 second of DMA operations. In heavy and manufacturing modes, this test performs 5 seconds of DMA operations.

This test does not run as part of the boot sequence in normal mode; it does run during the boot sequence in heavy and manufacturing modes. To run the Xtalk stress test from the >> prompt, enter:

```
>> diag_enet -t xtalk [-N <nasid>] [-m <moduleid> <-b bus>] [-p <pcidev>]
[-h] [-e|-x] [-v] [-r <count>] [-R <count>]
```

Refer to “Ethernet Tests” on page 3-52 for descriptions of all available options.

If you use the `-e` option, this test uses the external loopback path instead of the twister internal loopback. If you use the `-r` option, this test prints the retry counts every 50 seconds. When you run this test manually from the command line, the command reads and clears out the LLP retry count registers on the hub and Xbridge before and after the test runs.

Note: Ensure that the following hardware is functional before you run this test:

- Non-Ethernet-specific logic on the IOC3 ASIC
- PCI bus address / data
- Xbridge ASIC
- Motherboard components

Note: Also ensure that the twister chip internal loopback test runs without errors before you run this test.

The failure messages are the same as the failure messages for the twister chip internal loopback test. This test also includes the following messages:

- `enet_xtalk_stress: RX CONSUME did not increment from Xtalk stress loop` *value* `to` *value*.
  `enet_xtalk_stress: TX CONSUME did not increment from Xtalk stress loop` *value* `to` *value*.
- `enet_xtalk_stress: Unexpected EISR condition in Xtalk stress loop` *value*.
  `enet_xtalk_stress:  EISR value =` *e_value*
- `enet_xtalk_stress: Unexpected Bridge ISR condition in Xtalk stress loop` *value*.
  `enet_xtalk_stress:  Bridge ISR value =` *value*

This test always returns the following message when the test detects an error:

- `RSLT enet_xtalk_stress FAIL`

If this test fails, replace the IP34 motherboard.

Chapter 4

Offline Diagnostics

Offline diagnostics are tests that run under the scalable micro-diagnostic kernel (sMDK). sMDK is a stand-alone diagnostic environment that runs in kernel mode. sMDK provides test access to hardware components that cannot be accessed from a user program that is running under the operating system.

Offline diagnostics require full use of the system: you must bring down (reboot) the system to run them. No operating system can be running in the system when you use the offline diagnostics.

Use the offline diagnostics to isolate failures for which the online diagnostics do not provide enough error information or to test hardware that the online diagnostics cannot test.

Table 4-1 describes the offline diagnostics that are available.

Table 4-1 Offline Diagnostics

Name	Description	Release[a]
`CPU_Test`	CPU test	Customer and internal
`sct`	Secondary cache test	Customer and internal
`scct`	Cache coherency test	Internal
`nmem`	Memory test	Customer and internal
`ndir`	Directory memory test	Internal
`io8bt`	I/O test (checks the following hardware on the IP34 motherboard: IOC3, USB, SCSI controller components, and PCI slot, parallel port, keyboard port, and mouse port)	Customer and internal
`xbg`	Xbridge ASIC and the PCI slot test	Customer and internal

a. The “Release” column indicates the software release that contains each offline diagnostic. A “Customer” entry indicates that the diagnostic is available on a *Customer Diagnostics* CD. An “Internal” entry indicates that the diagnostic is available on an *Internal Support Tools* CD. Refer to “Offline Diagnostic Distribution” on page 4-2 for more information.

The offline diagnostic tests are divided into the following categories:

- CPU test (`CPU_Test`)
- Secondary cache tests (`sct` and `scct`)
- Memory tests (`nmem` and `ndir`)
- I/O tests (`io8bt` and `xbg`)

Figure 4-1 shows the hardware components that the offline diagnostics test when you run them automatically.

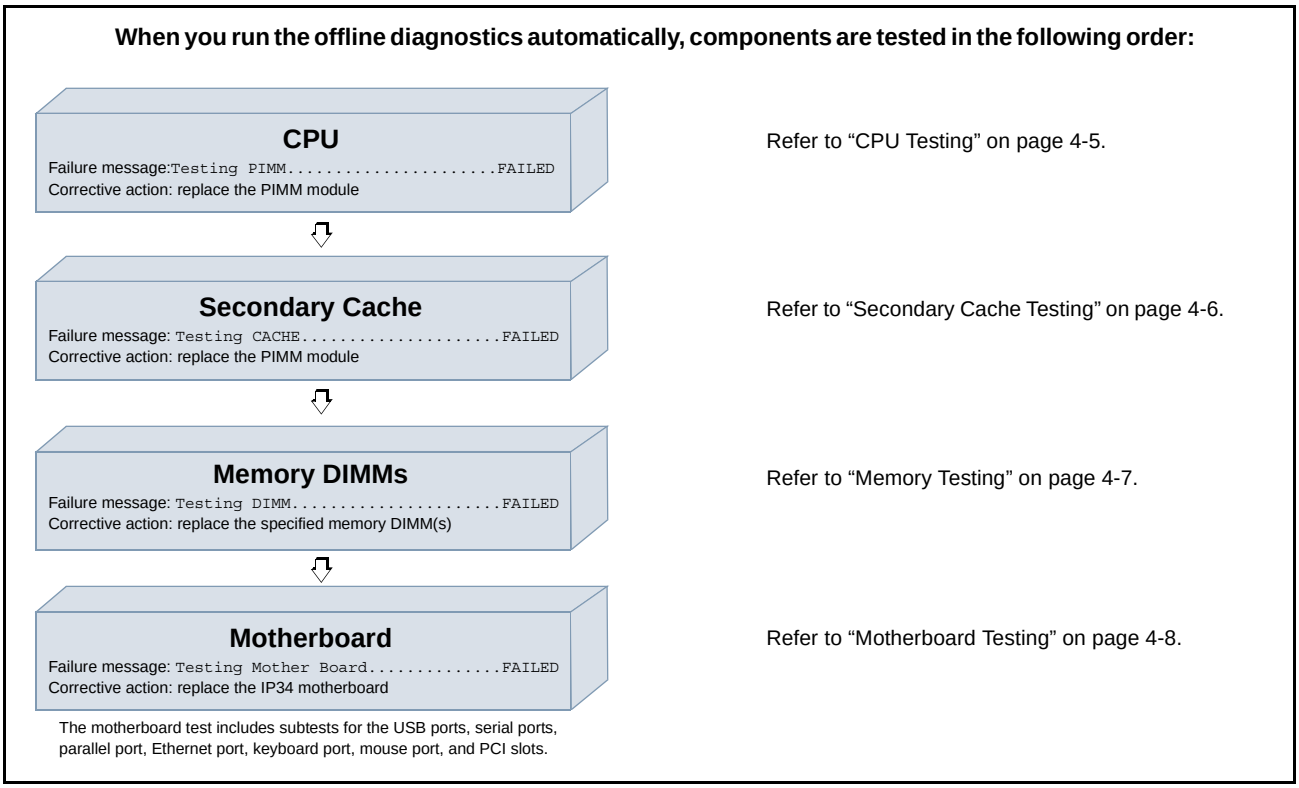

Figure 4-1 Components Tested by Offline Diagnostics

4.1 Offline Diagnostic Distribution

The customer-usable offline diagnostics (`CPU_test`, `sct`, `nmem`, `io8bt`, and `xbg`) are installed on the system disk with the operating system. The `/stand/smdk` directory structure contains the files for the offline diagnostics. You do not need to install any additional software to run the customer-usable offline diagnostics.

Note: The *Customer Diagnostics 1.0* CD, SGI part number 812-1122-001, also includes the customer-usable offline diagnostics. This CD ships with all Silicon Graphics Fuel visual workstations. You can use this CD to reinstall the customer-usable offline diagnostics, if necessary. The CD booklet includes installation procedures.

The *Internal Support Tools 2.7* CD, SGI part number 812-0640-011, includes internal versions of the customer-usable offline diagnostics (`CPU_test`, `sct`, `nmem`, `io8bt`, and `xbg`) and additional offline diagnostics (`scct` and `ndir`) that are available only for SGI personnel. You need to install the files on a system before you can run these offline diagnostics. The CD booklet includes installation procedures.

4.2 Running the Offline Diagnostics Automatically

The offline diagnostics include a "launcher" that automatically runs a sequence of tests. In most cases, you should run the offline diagnostics automatically with the launcher. Use the following procedure to run the offline diagnostics automatically:

1. Power on the system.
2. Wait until the system maintenance menu appears.

   Note: If the *AutoLoad* PROM environment variable is set to `Yes`, you must click on the `Stop for Maintenance` button to access the system maintenance menu.

3. Select the `Run Diagnostics` option.

   Note: You can also start the launcher by entering the following command at the command monitor (PROM) prompt (`>>`):

   ```
   boot -f dksc(0,1,0)/stand/smdk/smdk --a
   ```

The launcher automatically runs the offline diagnostics to test the system components in the following order:

1. The `CPU_Test` test checks the CPU.
2. The `sct` test checks the secondary cache.
3. The `nmem` test checks the memory DIMMs.
4. The `io8bt` and `xbg` tests check the motherboard (including the USB ports, serial ports, Ethernet port, parallel port, mouse port, keyboard port, Xbridge ASIC, and PCI slots).

The offline diagnostics test the simpler hardware first and then proceed to the more complex hardware.

Note: When you run the offline diagnostics from the graphics console, you can press Esc to stop the offline diagnostics and automatically reboot the system.

Table 4-2 shows the approximate time required (in minutes:seconds format) to automatically run the offline diagnostics on a workstation with a 500-MHz processor and 512 MB of memory. (Testing time varies for different hardware configurations.)

Table 4-2 Time Required to Test a System with a 500-MHz Processor and 512 MB of Memory

Testing Progress	Total Elapsed Time
The launcher boot-up sequence starts	0:00
The launcher boot-up sequence completes	0:10
PIMM testing completes	0:40
Secondary cache testing completes	1:17
Memory DIMM testing completes	5:05
Motherboard testing completes	7:30

Example output:

```
SMDK SGI Version 6.93 TEST built 12:42:40 PM Nov 14, 2001
smdk loading io discovery code...
smdk loading launcher code...
smdk>term none
Setting up diagnostics.....
Starting diagnostics.....


Testing  PIMM........   PASSED
Testing  CACHE........   PASSED
Testing  DIMM.....................................   PASSED
Testing  Mother Board.....................   PASSED

FINISHED
All diagnostics passed.
Resetting...
```

The following sections describe the tests that the launcher runs in the order that it runs them. The launcher automatically runs the tests and displays output; you do not need to enter any input.

4.2.1 CPU Testing

To perform CPU testing, the launcher uses the `CPU_Test` to verify that the CPU is operating correctly.

If the launcher completes CPU testing without detecting an error, it displays the following message:

```
Testing  PIMM..................................... PASSED
```

When CPU testing completes without detecting errors, testing continues with cache testing.

If the launcher detects a CPU error, it displays the following message:

```
Testing  PIMM..................................... FAILED
```

Then, it displays the following message and offline testing stops:

```
STOPPED.
```

After testing stops, the `smdk>` prompt appears. You can enter sMDK commands at this prompt. (Refer to “sMDK Commands” on page 29 for information about the sMDK commands.) You should drop the current test before you run any other tests.

If CPU testing fails, replace the PIMM.

4.2.2 Secondary Cache Testing

To perform secondary cache testing, the launcher uses the `sct` test to verify that the secondary data and instruction caches are operating correctly.

If the launcher completes cache testing without detecting an error, it displays the following message:

```
Testing  CACHE..................................... PASSED
```

When cache testing completes without detecting errors, testing continues with memory testing.

If the launcher detects a cache error, it displays the following message:

```
Testing  CACHE..................................... FAILED
```

Then, it displays the following message and offline testing stops:

```
STOPPED.
```

After testing stops, the `smdk>` prompt appears. You can enter sMDK commands at this prompt. (Refer to “sMDK Commands” on page 29 for information about the sMDK commands.) You should drop the current test before you run any other tests.

If cache testing fails, replace the PIMM.

4.2.3 Memory Testing

To perform memory testing, the launcher uses the `nmem` test to verify that the memory DIMMs are operating correctly.

If the launcher completes memory testing without detecting an error, it displays the following message:

```
Testing  DIMM..................................... PASSED
```

When memory testing completes without detecting errors, testing continues with motherboard testing.

If the launcher detects a memory error, it displays the following message:

```
Testing  DIMM..................................... FAILED
```

Then, it displays a message that indicates the failing memory DIMM(s) to replace.

Example 1: `Replace DIMM 2     (Uncorrectable Error(s) on pair 2/3)`

This message appears when the test detects uncorrectable errors on a DIMM pair (DIMM pair 2/3 in this example) and also detects several correctable errors on a specific DIMM in the pair (DIMM 2 in this example).

Example 2: `Replace DIMM 3     (Excessive Correctable Errors)`

This message appears when the test detects excessive correctable errors on a single DIMM.

Example 3: `Replace DIMM 0`

This message appears when the launcher cannot determine the details of the failure. This happens when `nmem` detects the error instead of the hardware. (The hardware should detect the majority of errors.)

Then, it displays the following message and offline testing stops:

```
STOPPED.
```

After testing stops, the `smdk>` prompt appears. You can enter sMDK commands at this prompt. (Refer to “sMDK Commands” on page 29 for information about the sMDK commands.) You should drop the current test before you run any other tests.

If memory testing fails, replace the specified DIMM(s).

4.2.4 Motherboard Testing

To perform motherboard testing, the launcher uses the `io8bt` and `xbg` tests to verify that the USB ports, serial ports, parallel port, Ethernet port, keyboard port, mouse port, Xbridge ASIC, and PCI slots are operating correctly.

If the launcher completes motherboard testing without detecting an error, it displays the following message:

```
Testing  Mother Board.............................. PASSED
```

When motherboard testing completes without detecting errors, the launcher displays a message indicating that all tests have passed and then reboots the system.

If the launcher detects a motherboard error, it displays the following message:

```
Testing  Mother Board.............................. FAILED
```

Then, it displays the following message and offline testing stops:

```
STOPPED.
```

After testing stops, the `smdk>` prompt appears. You can enter sMDK commands at this prompt. (Refer to “sMDK Commands” on page 29 for information about the sMDK commands.) You should drop the current test before you run any other tests.

If motherboard testing fails, replace the IP34 motherboard.

4.3 Running the Offline Diagnostics Manually

To run the offline diagnostics manually:

1. Start the offline diagnostic environment.
2. Run the desired offline diagnostic.

Note: You should run only one test at a time. Be sure to use the `drop` command to quit the current test before you run a new test.

4.3.1 Starting the Offline Diagnostic Environment

Use the following procedure to start the sMDK offline diagnostic environment:

1. Power on the system.
2. Wait until the system maintenance menu appears.

   Note: If the *AutoLoad* PROM environment variable is set to `Yes`, you must click on the `Stop for Maintenance` button to access the system maintenance menu.

3. Select the `Enter Command Monitor` option.
4. At the `>>` prompt, enter the following command to load the sMDK offline diagnostic environment:

   ```
   >> boot -f dksc(0,1,0)/stand/smdk/smdk
   ```

   sMDK displays the following information as it loads:

   ```
   136536+15728 entry: 0xa8000000001142f0
   INFO: console subchannel changed:  001a01 CPU0
   SMDK SGI Version 6.93 TEST built 12:42:40 PM Nov 14, 2001
   smdk loading io discovery code...
   smdk>
   ```

4.3.2 Running CPU_Test Manually

The CPU_Test offline diagnostic tests the CPU by generating and executing random instructions from the MIPS instruction set (except kernel instructions). Some features tested include the ALU, the floating-point unit, branches, jumps, fetches, and stores.

CPU_Test runs in passes; the phases of each pass include:

- Generation
  - Generate the random instruction.
  - Generate the random address.
  - Generate the random data.
- Execution
  - Execute the generated instruction sequence.
- Result checking
  - Compare the results in all GPRs, FPRs, and memory data areas.

Perform the following procedure to run CPU_Test manually:

1. If sMDK is not running, start sMDK. (Refer to “Starting the Offline Diagnostic Environment” on page 4-9.)
2. Run the test:

   smdk> **run CPU_Test [***test_options***]**

   Recommended: **run CPU_Test -u 1 -s 1 -e 200 -y Compare_Iteration**

   Refer to Table 4-3 for descriptions of the command-line options that are available.

   Note: The test begins printing status messages as soon as it starts.
3. View the test status:

   smdk> **ds**

   Note: Press <Enter> to return to the smdk> prompt.
4. If the test detects errors, view the ERROR display:

   smdk> **dscr 1 cpu_error 0 <***error_number***>**
5. Press <Enter> to return to the smdk> prompt.
6. Stop the test:

   smdk> **drop 1**
7. Reset the system:

   smdk> **reset**

Table 4-3 CPU_Test Command-line Options

Option	Description
`-h`	Displays help information
`-s` <*starting_pass*>	Specifies the starting pass (default: 1)
`-e` <*ending_pass*>	Specifies the ending pass (default: 100000000)
`-l`	Causes the test to loop between the starting pass and ending pass
`-u` <*value*>	Specifies how often (in passes) the test should update the user (default: 10000) If you set this parameter to a low value, the test may appear to freeze after you display the status screen or one of the displays (for example, the RUN display). If this happens, press <Enter> to return to the `smdk>` prompt. Test output resumes updating.
`-c` <*value*>	Specifies the number of CPUs to use (default: all CPUs)
`-d` <*option*>	Specifies the values to dump and display (default: no dump): `dump_all` = display addresses, registers, instructions `dump_addrs` = display code and data addresses `dump_regs` = display initial register values `dump_instrs` = display randomly generated instructions `dump_op_weight` = display op weight table
`-w`	Specifies that the test should not write the trace table entries (default: write trace table entries)
`-y` <*option*>	Selects a result comparison mode: `Test_1_CPU_Rslt_Chk` = do not perform single CPU checking `Compare_Interation` = compare results between two iterations `Compare_Saved` = compare results with a set of saved results `Capture_1_CPU_Rslt_Chk` = save results to use with `Compare_Saved` option (default: `Test_1_CPU_Rslt_Chk`)
`-z`	Specifies that no output should be displayed (used with the launcher)

Troubleshooting tips:

1. Loop on the failing pass.
2. Loop on a set of passes surrounding the failing pass.
3. Print the generated instructions, addresses, and data.
4. View the trace table.

If you want to reproduce an error, you can configure `CPU_Test` to loop on a specific pass. `CPU_Test` reproduces the same sequence of random instructions, addresses, and data that it generated the first time for that pass.

4.3.3 Running sct Manually

The `sct` offline diagnostic tests the secondary cache.

This test has four sections:

Section 0 writes walking 0/1 patterns to the secondary cache, reads the data back from the cache, and compares the results to the original data. Then, this section writes random data to the secondary cache, reads the data back, and compares the results to the original data.

Section 1 performs addressing tests.

Section 2 performs an ECC RAM test and an ECC interrupt test.

Section 3 performs a TAGs test.

Perform the following procedure to run `sct` manually:

1. If sMDK is not running, start sMDK. (Refer to "Starting the Offline Diagnostic Environment" on page 4-9.)
2. Run the test:

   `smdk> **run** sct [*test_options*]`

   Refer to Table 4-4 for descriptions of the command-line options that are available.
3. View the test status:

   `smdk> **ds**`

   Note: Press <Enter> to return to the `smdk>` prompt.
4. View the RUN display

   `smdk> **dscr 1 run 1**`

   Note: Press <Enter> to return to the `smdk>` prompt.
5. If the test detects errors, view the ERROR display:

   `smdk> **dscr 1 error 1**`
6. Press <Enter> to return to the `smdk>` prompt.
7. Stop the test:

   `smdk> **drop 1**`
8. Reset the system:

   `smdk> **reset**`

Table 4-4 sct Command-line Options

Option	Description
`-? | -h | -H`	Displays help information
`-s <`*starting_pass*`>`	Specifies the starting pass (default: 0)
`-e <`*ending_pass*`>`	Specifies the ending pass (default: 1)

Table 4-4 (continued) sct Command-line Options

Option	Description
`-l`	Turns on loop mode, which causes the test to run forever between the starting pass and the ending pass
`-S` <*section_select*>	Selects the test sections to run (hexadecimal bitmask) 0x01 : section 0 0x02 : section 1 0x03 : section 2 0x08 : section 3
`-B` <*subsection_select*>	Selects the test subsections to run (hexadecimal bitmask) 0x01 : subsection 0 0x02 : subsection 1 0x03 : subsection 2 0x08 : subsection 3
`-C`	Selects the continue-on-error mode (0 = continue on error; 1 = stop on error)
`-c` <*cpu_list*>	Selects the virtual CPUs to use
`-d` <*level*>	Specifies the level of debug messages to display
`-f` <*seed*>	Sets a fixed seed value

4.3.4 Running scct Manually

The `scct` offline diagnostic creates conflict opportunities within the cache communication protocol and stresses the coherence machine. By doing this, it also implicitly tests cache storage (both data and tags). It uses random TLB tags to exercise the TLB within the processor.

Perform the following procedure to run the cache coherency test:

1. If sMDK is not running, start sMDK. (Refer to "Starting the Offline Diagnostic Environment" on page 4-9.)
2. Run the test:

   `run scct [`*`test_options`***`]`**

   Refer to Table 4-5 for the command-line options for the cache coherency test.
3. View the test status:

   `smdk>` **`ds`**
4. Press <Enter> to return to the `smdk>` prompt.
5. Enter the following command to view the RUN display:

   `dscr <`*`index_value`***`> run 1`**

   Note: Press <Enter> to return to the `smdk>` prompt.
6. If the test detects errors, enter the following command to view page 2 of the RUN display:

   `dscr 1 run 2`
7. Press <Enter> to return to the `smdk>` prompt.
8. Enter the following command to stop the test:

   `drop 1`
9. Reset the system:

   `smdk>` **`reset`**

Table 4-5 scct Command-line Options

Option	Description
`-h | -H | -?`	Displays the help information
`-s <`*`starting_pass`*`>`	Specifies the starting pass
`-e <`*`ending_pass`*`>`	Specifies the ending pass
`-l`	Causes the test to loop between the selected starting pass and ending pass
`-S <`*`bitmask`*`>`	Selects the test section to run (bitmask)

Table 4-5 (continued) scct Command-line Options

Option	Description
`-m` <*value*>	Specifies the lowest physical byte address that can be accessed in each node (The actual range of addresses to test is determined by the configuration, resource availability, and the `-M` and `-b` options. Page 3 of the RUN display lists the memory segments that are tested.)
`-M` <*value*>	Specifies the highest physical byte address that can be accessed in each node
`-b` <*select_mask*>	Selects the physical memory banks to use in each node (a bitmask where bits 0 through 7 correspond to banks 0 through 7)
`-v` <*verbosity_level*>	Specifies the verbosity level of the test (0 through 2)

The `scct` diagnostic reports error information on the second page of the `RUN` display.

4.3.5 Running nmem Manually

The `nmem` offline diagnostic is a directed test that verifies that the memory is operating properly: It verifies the paths from the Bedrock ASIC to memory, tests memory integrity, and verifies proper SECDED operation.

This test has five sections. (Test complexity increases with each successive section.)

Section 0 performs a quick-screen test. This section writes data patterns to memory and reads data back from the memory. (It uses the March test algorithm, which repeats an address pattern in each 32-bit word of a quadword.) It compares the data that was written with the data that was read.

Section 1 performs a memory integrity test. This section verifies the integrity of each memory cell. (It uses the March test algorithm with checkerboard and random data patterns.) This section also tests memory accesses with strides across cache lines, banks (internal), and memory pages (rows).

Section 2 performs a SECDED verification test. This section verifies the integrity of each memory cell in SECDED memory. This section also verifies that the SECDED hardware corrects single-bit errors and detects double-bit errors. This section forces errors and tests interrupts.

Section 3 performs a memory random address and data test. This section verifies the data integrity of memory. It writes random data to random addresses but does not verify data. It relies on SECDED to report any memory errors.

Section 4 performs a fetch-and-op test. This section verifies proper operation of fetch and op operations by using uncached memory references in the Mspec space. It uses up to four processors to execute walk-by code, which verifies that references to the same location at the same time are handled correctly. It uses the following operation types: fetch, fetch and increment, fetch and decrement, fetch and clear, initialize, increment, decrement, AND, and OR.

Perform the following procedure to run nmem manually:

1. If sMDK is not running, start sMDK. (Refer to "Starting the Offline Diagnostic Environment" on page 4-9.)
2. Run the test:

   smdk> **run nmem [***test_options***]**

   Refer to Table 4-6 for descriptions of the command-line options that are available.
3. View the test status:

   smdk> **ds**

   Note: Press <Enter> to return to the smdk> prompt.
4. View the RUN display:

   smdk> **dscr 1 run 1**

   Note: Press <Enter> to return to the smdk> prompt.
5. If the test detects errors, view the ERROR display:

   smdk> **dscr 1 run 2**
6. Press <Enter> to return to the smdk> prompt.
7. Stop the test:

   smdk> **drop 1**
8. Reset the system:

   smdk> **reset**

Table 4-6 nmem Command-line Options

Option	Description
-n <*nodes*>	Selects the nodes to test (default: all available nodes)
-c <*cpu_list*>	Selects the CPUs to use (default: all available CPUs)
-S <*section_select*>	Selects the test sections to run (default: 0x1F, which selects all sections)
-s <*starting_pass*>	Specifies the starting pass (default: 0)
-e <*ending_pass*>	Specifies the ending pass (default: 1)
-l	Turns on loop mode, which causes the test to run forever between the starting pass and the ending pass
-C	Selects continue on error
-b <*banks*>	Selects the memory banks to use
-m <*starting_address*>	Selects the node memory starting address
-M <*ending_address*>	Selects the node memory ending address
-? | -h | -H	Displays help information

Troubleshooting: tip:

Additional information might be available in the error log. Enter the `el` command at the `smdk>` prompt to view the error log.

4.3.6 Running ndir Manually

The `ndir` offline diagnostic is a directed test that verifies that the directory memory is operating properly. The `ndir` test verifies the paths from the Bedrock (hub) chip to directory memory, tests the integrity of directory memory, and verifies proper SECDED operation in directory memory.

This test uses five test sections. (Test complexity increases as the section number increases.)

Section 0 performs a directory memory quick-screen test.

This section writes data patterns to directory memory and reads data back from directory memory. (It uses directory memory backdoor access to execute a March test algorithm with address data.) It compares the data that was written with the data that was read.

Section 1 performs a directory memory integrity test.

This section verifies the integrity of each memory cell in the directory memory. (It uses directory memory backdoor access to execute a March test algorithm with checkerboard and random data patterns.)

Section 2 performs a directory memory SECDED verification test.

This section verifies that the SECDED hardware corrects single-bit errors and detects double-bit errors in directory memory. It forces errors and tests interrupts.

Section 2 uses directory memory backdoor access to access directory memory. It checks SECDED for the directory entries only and not the region protection nor the page counter that are both in the same physical memory area. (The region protection is protected by a parity for each bit, and the page counters have no protection scheme.) This section also checks the parity error detection for the region protection data.

Section 3 performs a directory memory random address / data test.

This section randomly tests selected blocks in directory memory. For each iteration, section 3 selects a random block of contiguous directory memory, a random stride, and an initial random data pattern. It uses addresses that wrap to the starting address (plus an offset to stay within the currently selected block).

Section 4 does not run on Silicon Graphics Fuel visual workstations.

Perform the following procedure to run the directory memory test:

1. If sMDK is not running, start sMDK. (Refer to "Starting the Offline Diagnostic Environment" on page 4-9.)
2. Run the test:

   `run ndir [`*`test_options`***`]`**

   Refer to Table 4-7 for the command-line options for the directory memory test.
3. View the test status:

   `smdk>` **`ds`**

   Note: Press <Enter> to return to the `smdk>` prompt.
4. Enter the following command to view the RUN display:

   `dscr 1 run 1`

   Note: Press <Enter> to return to the `smdk>` prompt.
5. If the test detects errors, enter the following command to view page 2 of the RUN display:

   `dscr 1 run 2`
6. Press <Enter> to return to the `smdk>` prompt.
7. Enter the following command to stop the test:

   `drop 1`
8. Reset the system:

   `smdk>` **`reset`**

Table 4-7 ndir Command-line Options

Option	Description
`-n` <*nodes*>	Selects the nodes to test (default: all available nodes)
`-c` <*cpu_list*>	Selects the CPUs to use (default: all available CPUs)
`-S` <*section_select*>	Selects the test sections to run (default: 0x1F, which selects all sections)
`-s` <*starting_pass*>	Specifies the starting pass (default: 0)
`-e` <*ending_pass*>	Specifies the ending pass (default: 1)
`-l`	Turns on loop mode, which causes the test to run forever between the starting pass and the ending pass
`-C`	Selects continue-on-error mode
`-b` <*banks*>	Selects the memory banks to use
`-m` <*starting_address*>	Selects the starting directory address
`-M` <*ending_address*>	Selects the ending directory address
`-? | -h | -H`	Displays help information

The second page of the RUN display reports error information.

Example:

```
NDIR Run Display - Error Information                        Page 2 of 3
_____________________________________________________________________________
                                                  Error Number: 0002 of 0002
VCpu...:  0000,   Pid........:  0003,             Error Count.: 0001
VNode..:  0000,   Nasid......:  0000,
Section: 0001,  Sub-Section: 0000,    Condition: 0012, Pass: 0x00000001
File...: ndirFuncs.c,                 Line.....: 0147, Fail: 0x00000002
_____________________________________________________________________________

   NodeSync Error:  Timed out waiting for 2 cpus to sync up.

   1 cpus waited for 120000000 us. before timeout.

   Sync Id: 0x100000a8.

   Time-out waiting for the following cpu processes to sync up:
   VCpu: 0, Pid: 3
   VCpu: 1, Pid: 3
_____________________________________________________________________________
 [1]=next pg, [2]=next vnode, [3]=next error, [shift-num] for previous
```

Additional information might be available in the error log. Use the `el` command to view the error log.

Use the form at the following URL to decode the error log:

`http://wwwcf.americas.sgi.com/PUBLIC/sn1_diags/elog_decode/elog_decode.cgi`

4.3.7 Running io8bt Manually

The `io8bt` offline diagnostic is a directed test that verifies the I/O hardware on the IP34 motherboard (IOC3, USB ports, SCSI controller components, and PCI slots). It also tests the parallel port and the keyboard and mouse connections.

Section 0 performs a device access test. This section verifies that each selected device can be accessed. It tests the PCI configuration space, resets, reset propagation to each device, and device-to-device reads and writes.

Section 1 verifies the IOC3 hardware. This section tests all areas of the IOC3, including SSRAM, parity on SSRAM, PHY, and external loopback.

Section 2 verifies the USB hardware.

Section 3 verifies the SCSI controller hardware.

Section 4 verifies the parallel port, mouse, and keyboard.

Perform the following procedure to run `io8bt` manually:

1. If sMDK is not running, start sMDK. (Refer to "Starting the Offline Diagnostic Environment" on page 4-9.)
2. Run the test:

   `smdk>` **`run io8bt [`***`test_options`***`]`**

   Refer to Table 4-8 for descriptions of the command-line options that are available.
3. View the test status:

   `smdk>` **`ds`**

   Note: Press <Enter> to return to the `smdk>` prompt.
4. View the RUN display:

   `smdk>` **`dscr 1 run 1`**

   Note: Press <Enter> to return to the `smdk>` prompt.
5. If the test detects errors, view the ERROR display:

   `smdk>` **`dscr 1 error 1`**
6. Press <Enter> to return to the `smdk>` prompt.
7. Stop the test:

   `smdk>` **`drop 1`**
8. Reset the system:

   `smdk>` **`reset`**

Table 4-8 io8bt Command-line Options

Option	Description
`-C` [*value*]	Selects the action to perform when an error occurs 0 = continue testing 1 = stop testing 2 = fill the buffer and stop testing
`-d` [*bitmask*]	Specifies the device to use (bitmask) xxx1 = test the IOC3 hardware (dev 4) xx1x = test the USB hardware (dev 5) x1xx = test the SCSI controller hardware (dev 1)
`-e` [*ending_pass*]	Specifies the ending pass count (default: 0 [run forever])
`-h` | `-H`	Displays help information
`-l` [*value*]	Selects the looping option 0 = loop mode off 1 = loop mode on (repeats the test based on the `-s` and `-e` values) **Note:** If you set the `-e` option to 0, this option is disabled.
`-n` [*node*]	Selects a specific node to use, which isolates testing to a specific node
`-p` [*port*]	Selects a specific port to use, which isolates testing to a specific port
`-s` [*starting_pass*]	Specifies the starting pass count (default: 0)
`-S` [*bitmask*]	Selects which sections of the diagnostic to run xxxx xxx1 = section 0 (check PCI configuration registers and reset) xxxx xx1x = section 1 (check IOC3 hardware) xxxx x1xx = section 2 (check USB hardware) xxxx 1xxx = section 3 (check SCSI controller hardware) xxx1 xxxx = section 4 (check parallel port, keyboard, and mouse)
`-E` [*value*]	Selects the Ethernet test mode: 0 = perform no Ethernet testing 1 = perform only internal loopback testing (does not require loopback connector or cable changes) 2 = perform internal and external loopback testing (requires an external loopback connector on the Ethernet port) 4 = display Ethernet testing help information
`-F` [*value*]	Specifies that the test should perform SCSI-specific operations 0 = no SCSI-specific parameters 1 = dump the physical registers from pages 0, 1, and 7

Table 4-8 (continued) io8bt Command-line Options

Option	Description
`-I` [*value*]	Specifies IOC3 serial port parameters xxxx xxxx xxxx xxxx xxxx xxx1 = test serial port A xxxx xxxx xxxx xxxx xxxx xx1x = test serial port B xxxx xxxx xxxx xxxx xxxx x1xx = RS-232 or RS-422 mode 0 = RS-232 mode 1 = RS-422 mode (default = RS-232 mode) xxxx xxxx xxxx xxxx xxxx 1xxx = port A is externally looped to port B xxxx xxxx xxxx xxxx xxx1 xxxx = port A is externally looped back to itself xxxx xxxx xxxx xxxx xx1x xxxx = port B is externally looped back to itself xxxx xxxx xxxx xxxx x1xx xxxx = port A has terminal connected (set baud) xxxx xxxx xxxx xxxx 1xxx xxxx = port B has terminal connected (set baud) xxxx xxxx xxxx xxx1 xxxx xxxx = set serial port A to 9,600 baud xxxx xxxx xxxx xx1x xxxx xxxx = set serial port A to 19,200 baud xxxx xxxx xxxx x1xx xxxx xxxx = set serial port A to 38,400 baud xxxx xxxx xxxx 1xxx xxxx xxxx = set serial port A to 56,000 baud xxxx xxxx xxx1 xxxx xxxx xxxx = set serial port A to 115,000 baud xxxx xxx1 xxxx xxxx xxxx xxxx = set serial port B to 9,600 baud xxxx xx1x xxxx xxxx xxxx xxxx = set serial port B to 19,200 baud xxxx x1xx xxxx xxxx xxxx xxxx = set serial port B to 38,400 baud xxxx 1xxx xxxx xxxx xxxx xxxx = set serial port B to 56,000 baud xxx1 xxxx xxxx xxxx xxxx xxxx = set serial port B to 115,000 baud
`-N` [*value*]	Selects NVRAM testing mode 0 = limited testing (check 0x7fd1 - 0x7ff7, do not check 0x7ff0) 1 = maximum testing (preserves NVRAM contents, if possible) 2 - pattern all NVRAM (even locations = 0x25; odd locations = 0x52) 3 = read / check all NVRAM locations 4 = read / check limited areas of NVRAM 5 = skip all NVRAM testing 0x*XXX*0*YYY*f = dump NVRAM (*XXX*0 = first byte and *YYY*f = last byte) To perform NVRAM testing, you must use the `-N` option. **Caution:** Do not stop the test while it is testing the NVRAM (during test section 4). The NVRAM stores important system data.
`-R` [*bitmask*]	Sets parameters for the real-time interrupt (RTO / RTI) port xxx1 = RTO is externally looped back to RTI xx1x = RTO scope / measure mode (code loops forever) x1xx = skip RTO / RTI testing
`-U` [*value*]	Specifies that the test should perform USB-specific operations (default = 0) 0 = no USB-specific parameters 1 = attempt to detect an external USB device and then stop the test

Table 4-8 (continued) io8bt Command-line Options

Option	Description
`-K` [*value*]	Selects the keyboard and mouse tests to run: 0 = do not test the keyboard and mouse 1 = perform only register testing 2 = perform register testing, identification, and built-in self-testing (requires a connected mouse and keyboard) (default = 1)
`-L` [*value*]	Selects the parallel port tests to run: 0 = do not test the parallel port 1 = perform register testing and internal loopback testing 2 = perform register testing, internal loopback testing, and external loopback testing (requires a loopback connector on the parallel port) (default = 1)

Troubleshooting tips:

1. Determine the failing section, subsection, and condition.
2. Check all connectors and cables that are used for the hardware that is being tested by that section, subsection, and condition combination.
3. Power cycle the system to verify that the failure remains.
4. If the failure remains, replace the IP34 motherboard.

4.3.8 Running xbg Manually

The xbg offline diagnostic is a directed test that checks the functionality of the Xbridge ASIC and the PCI slots. (This test requires Alteon PCI Gigabit Ethernet cards to test the PCI slots.)

Section 0 performs a basic register test.

Section 1 performs a RAM on Array (ROA) test.

Section 2 performs an interrupt test.

Section 3 verifies PCI configuration and control.

Section 4 performs a PCI DMA basic test.

Section 5 verifies PCI page-mapped addressing.

Section 6 verifies PCI direct-mapped addressing.

Section 7 verifies PCI DMA byte-swap functionality.

Section 8 verifies PCI DMA buffers.

Section 9 performs a PCI DMA random test.

Section 10 performs a PCI DMA concurrent test.

Perform the following procedure to run xbg manually:

1. If sMDK is not running, start sMDK. (Refer to "Starting the Offline Diagnostic Environment" on page 4-9.)
2. Run the test:

   smdk> **run xbg [***test_options***]**

   Refer to Table 4-9 for descriptions of the command-line options that are available.
3. View the test status:

   smdk> **ds**

   Note: Press <Enter> to return to the smdk> prompt.
4. View the RUN display:

   smdk> **dscr 1 run 1**

   Note: Press <Enter> to return to the smdk> prompt.
5. If the test detects errors, view the ERROR display:

   smdk> **dscr 1 error 1**
6. Press <Enter> to return to the smdk> prompt.

7. Stop the test:

   ```
   smdk> drop 1
   ```

   Note: You must reset the system before you can run another test.

8. Reset the system:

   ```
   smdk> reset
   ```

Table 4-9 xbg Command-line Options

Option	Description
`-h | -H`	Displays help information
`-n <vnode>`	Specifies the virtual node to use (default: 0xffffffffffffffff [discover hardware])
`-p <port>`	Specifies the port to use: bit 8 = XIO port 8 bit 9 = XIO port 9 bit 10 = XIO port A ... bit 15 = XIO port F (default: 0x000000000000f300 [all XBOW ports])
`-d <device>`	Selects the devices to use: bit 1 = port 8, device 1 bit 2 = port 8, device 2 ... bit 8 = port 8, device 8 bit 9–16 = port 9, devices 1-8 bit 17–24 = port A, devices 1-8 bit 25–32 = port B, devices 1-8 bit 33–40 = port C, devices 1-8 bit 41–48 = port D, devices 1-8 bit 49–56 = port E, devices 1-8 bit 57–64 = port F, devices 1-8 (default: 0xffffffffffffffff [all devices])
`-C <continue_flag>`	Selects what the test should do when an error occurs (0 = continue, 1 = stop, 2 = fill the buffer and stop testing) (default: 0x0000000000000001)
`-q`	Turns on quick-look mode (default: off)
`-s <starting_pass>`	Specifies the starting pass count for random seed (default: 0x0000000000000000)
`-e <ending_pass>`	Specifies the ending pass count (0 = run forever) (default: 0x0000000000000002)
`-l <value>`	Specifies the number of times to loop (from the starting pass count to the ending pass count) (default: 0x0000000000000001)

Table 4-9 (continued) xbg Command-line Options

Option	Description
`-S` <*bitmask*>	Selects the test sections to run: bit 0 = section 0 bit 1 = section 1 ... bit 10 = section 10 (default: 0x00000000000007ff [all sections])

4.4 sMDK Commands

If you manually start the sMDK offline environment, there are several commands that you can enter at the `smdk>` prompt to control sMDK. These commands are grouped into the following categories:

- Configuration commands
- Control commands
- System commands
- Process commands

Note: For several commands, you must press <Enter> to return to the `smdk>` prompt.

4.4.1 Configuration Commands

4.4.1.1 cfg

`cfg`

Displays the hardware configuration: number of nodes, number of CPUs, NASID-to-vnode mapping, and vcpu-to-vnode mapping.

4.4.1.2 lpath

`lpath <`*load_directory_path*`>`

Specifies the location where the sMDK files are located. If you do not modify this setting, sMDK uses files from the `/stand/smdk` directory on the local system.

4.4.1.3 term

`term`

Displays the current terminal setting.

`term <`*terminal_type*`>`

Sets the terminal type. (Supported terminals are: iris-ansi and vt100; vt100 is the only value you can specify for *terminal_type*.)

4.4.1.4 version

`version`

Displays the version of sMDK that you are using and the date and time that it was built.

4.4.2 Control Commands

4.4.2.1 load

`load` <*filename*> <*options*>

Loads a test into memory and selects it as the current test. Use the load command to load a diagnostic test so you can view the memory that the test uses or set breakpoints before you run the test.

4.4.2.2 ds

`ds`

Displays the following diagnostic status information for all diagnostics that are running: index number, name, status, pass count, life count, and fail count.

The status information updates every 5 seconds until you press the <Return> key.

4.4.2.3 drop

`drop` <*diag_index*>

Kills all processes that are spawned by the diagnostic specified by <*diag_index*>.

4.4.2.4 dscr

`dscr` <*diag_index*>

Lists the displays that are available for the diagnostic test specified by <*diag_index*>.

`dscr` <*diag_index*> <*screen*> <*page*> <*optarg0*> <*optarg1*>

Displays the diagnostic information page that you specify.

The <*diag_index*> argument specifies the diagnostic, the <*screen*> argument specifies the name of the screen, and the <*page*> argument specifies the page to view.

The <*optarg0*> and <*optarg1*> arguments vary by diagnostic. (Refer to the individual diagnostic help pages for more information about these arguments.)

4.4.2.5 scrinfo

`scrinfo` <*diag_index*>

Displays information about each screen page, such as where it resides in memory and its size, for the diagnostic that you specify with <*diag_index*>. (This is a debugging feature.)

4.4.2.6 reset

`reset`

Resets the system.

4.4.3 System Commands

4.4.3.1 ping

`ping`

Requests a response from a the CPUs and reports when the CPU responds.

4.4.3.2 kl

`kl`

Displays a list of virtual CPUs whose kernel logs contain entries.

`kl <`*vcpu*`>`

Displays the kernel log for the virtual CPU that you specify.

4.4.3.3 klclr

`klclr`

Clears all kernel logs.

4.4.3.4 el

`el`

Lists all nodes that logged hardware-detected errors and the number of each type of error.

4.4.3.5 elclr

`elclr`

Clears the error logs on all nodes.

4.4.3.6 ps

`ps`

Displays the status of all processes.

4.4.3.7 tlb

`tlb` <*vcpu*> <*start_index*> <*number_of_entries*>

Displays the TLB entries for the specified virtual CPU, starting at the specified index for the specified number of entries. If you do not specify a number of entries, this command displays only one entry.

4.4.3.8 th

`th`

Displays the threshold values for all error types.

`th` <*error_type*> <*threshold*>

Sets the threshold for the specified error type.

4.4.4 Process Commands

4.4.4.1 p

`p`

Displays the current process.

`p` <*vcpu*> <*pid*>

Selects the process in which the commands operate.

4.4.4.2 run

`run`

Runs the current process. Use this command to start a process that was loaded with the `load` command or to start a process that has been frozen.

`run` <*test*> <*options*>

Loads a diagnostic from the host, selects it as the current process, and then runs the diagnostic.

4.4.4.3 bp

`bp` <*vaddr*>

Sets a breakpoint at the specified virtual address.

When test execution reaches the breakpoint, execution stops and the breakpoint is removed. Then, you can examine memory, set another breakpoint, or restart the test with the `run` command. You can set only one breakpoint for a process at a time.

4.4.4.4 rb

`rb`

Removes the breakpoint that is set in the current process.

4.4.4.5 dp

`dp` <*paddr*> [*count*]

Displays [*count*] words of physical memory starting at <*paddr*> address.

If you do not specify a [*count*] value, this command displays 1 word. If you do not specify cache ALG bits, the command combines the specified address with 0xa80000000000000 by using an OR function.

`dpa` <*paddr*> [*count*]

Displays [*count*] words (in ASCII format) of physical memory starting at address <*paddr*>. If you do not specify a [*count*] value, this command displays 1 word.

`dpb` <*paddr*> [*count*]

Displays [*count*] words (in byte format) of physical memory starting at address <*paddr*>. If you do not specify a [*count*] value, this command displays 1 word.

`dpc` <*paddr*> [*count*]

Displays [*count*] words (in code format) of physical memory starting at address <*paddr*>. If you do not specify a [*count*] value, this command displays 1 word.

`dpl` <*paddr*> [*count*]

Displays [*count*] words (in long format) of physical memory starting at address <*paddr*>. If you do not specify a [*count*] value, this command displays 1 word.

`dps` <*paddr*> [*count*]

Displays [*count*] words (in short format) of physical memory starting at address <*paddr*>. If you do not specify a [*count*] value, this command displays 1 word.

`dpw` <*paddr*> [*count*]

Displays [*count*] words (in word format) of physical memory starting at address <*paddr*>. If you do not specify a [*count*] value, this command displays 1 word.

4.4.4.6 dv

`dva` <*vaddr*> [*count*]

Displays [*count*] words (in ASCII format) of virtual memory starting at address <*vaddr*>. If you do not specify a [*count*] value, this command displays 1 word.

`dvb <vaddr> [count]`

Displays `[count]` words (in byte format) of virtual memory starting at address `<vaddr>`. If you do not specify a `[count]` value, this command displays 1 word.

`dvc <vaddr> [count]`

Displays `[count]` words (in code format) of virtual memory starting at address `<vaddr>`. If you do not specify a `[count]` value, this command displays 1 word.

`dvl <vaddr> [count]`

Displays `[count]` words (in long format) of virtual memory starting at address `<vaddr>`. If you do not specify a `[count]` value, this command displays 1 word.

`dvs <vaddr> [count]`

Displays `[count]` words (in short format) of virtual memory starting at address `<vaddr>`. If you do not specify a `[count]` value, this command displays 1 word.

`dvw <vaddr> [count]`

Displays `[count]` words (in word format) of virtual memory starting at address `<vaddr>`. If you do not specify a `[count]` value, this command displays 1 word.

4.4.4.7 mmap

`mmap`

Displays the memory map of the current process.

4.4.4.8 pmap

`pmap`

Displays the physical memory reservation table of the current process.

4.4.4.9 ef

`ef`

Displays the exception frame of the current process. (The exception frame includes the general purpose registers [GPRs], cause, status, badVaddr, and EPC.)

4.4.4.10 w

`w <vaddr>`

Sets a watchpoint for read or write operations for the current process at address `<vaddr>`.

4.4.4.11 wr

`wr` <*vaddr*>

Sets a watchpoint for read operations only for the current process at address <*vaddr*>.

4.4.4.12 ww

`ww` <*vaddr*>

Sets a watchpoint for write operations only for the current process at the virtual address that you specify with <*vaddr*>.

4.4.4.13 rw

`rw`

Removes a watchpoint for the current process.

Chapter 5

Online Diagnostics

Online diagnostics are tests that verify system hardware while the operating system is running. When you run an online diagnostic from the IRIX operating system prompt, the diagnostic runs a set of tests for a certain number of loops. Each online diagnostic has one or more *standard* tests that run by default if you do not specify a test in the command line. You may need to specifically request additional tests that you need to run.

Table 5-1 lists the online diagnostics that are available.

Table 5-1 Online Diagnostics

Name	Description	Release[a]
`bridgeloc`	PCI bridge locator and diagnostic listing tool	Customer and internal
`odydiags`	Graphics diagnostic	Internal
`olcdrom`	CD-ROM diagnostic	Customer and internal
`oldisk`	Disk diagnostic	Internal
`olenet`	Ethernet diagnostic	Customer and internal
`olmem`	Memory diagnostic	Customer and internal
`olpci`	PCI configuration viewer and diagnostic listing tool	Customer and internal
`olperi`	CPU diagnostic	Internal
`olsio`	SIO serial diagnostic	Customer and internal
`oltape`	Tape diagnostic	Customer and internal
`olusb`	USB diagnostic	Customer and internal
`olvst`	Socket-based network diagnostic	Customer and internal

Table 5-1 (continued) Online Diagnostics

Name	Description	Release[a]
pandora	System stress test	Customer and internal
runalldiags	Script to automatically run a sequence of tests	Customer and internal

a. The “Release” column indicates the software release that contains each online diagnostic. A “Customer” entry indicates that the diagnostic is available on a *Customer Diagnostics* CD. An “Internal” entry indicates that the diagnostic is available on an *Internal Support Tools* CD. Refer to “Online Diagnostic Distribution” on page 5-4 for more information.

The online diagnostic tests are divided into the following categories:

- CPU test (olperi)
- Memory test (olmem)
- I/O tests (bridgeloc, olpci, olenet, olsio, and olusb)
- Graphics test (odydiags)
- Storage device tests (olcdrom, oldisk and oltape)
- Network device test (olvst)
- System stress test (pandora)

Figure 5-1 shows the hardware components that the online diagnostics test when you run them automatically with the runalldiags script.

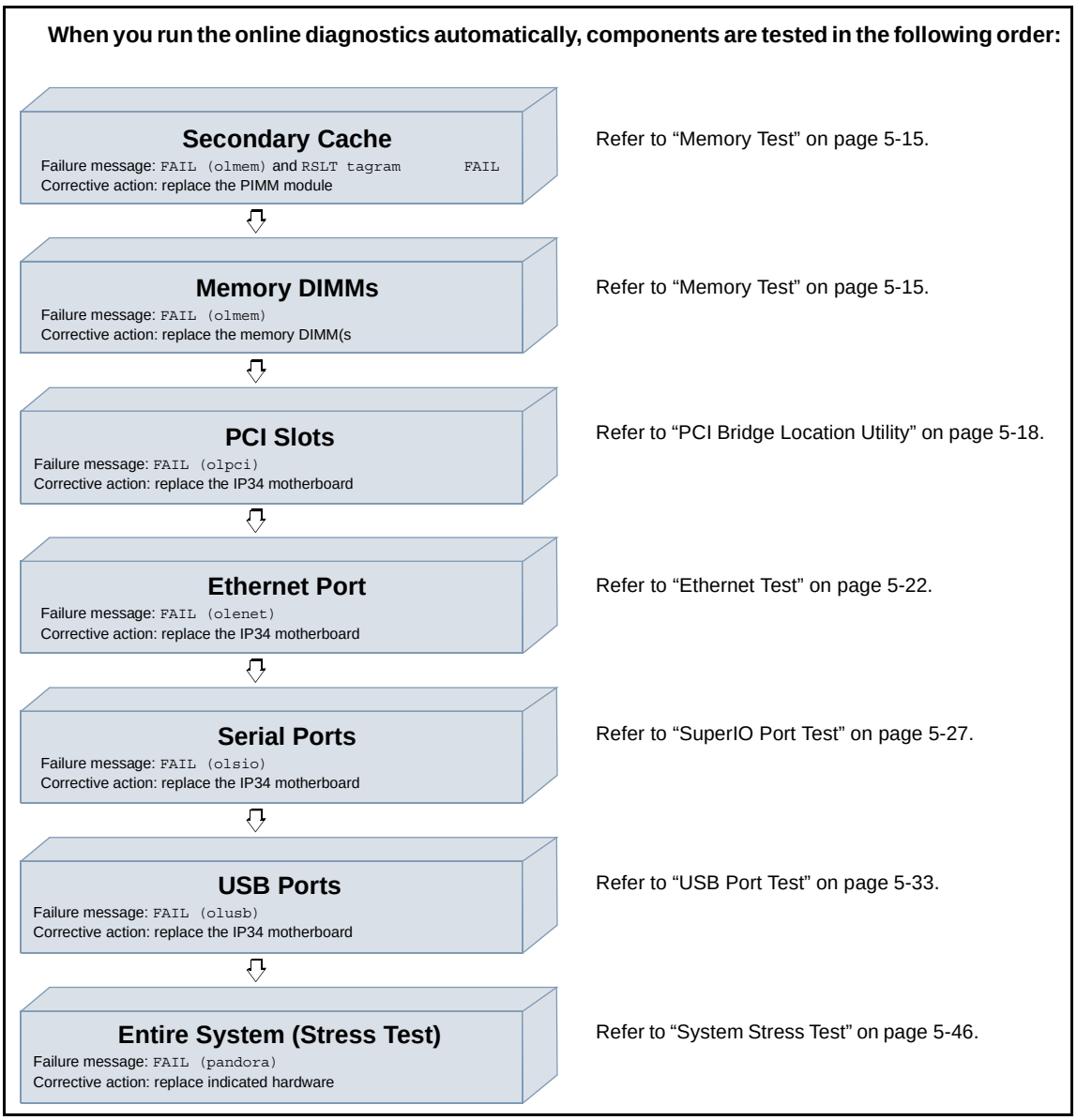

Figure 5-1 Components Tested by the Online Diagnostics

5.1 Online Diagnostic Distribution

The *Customer Diagnostics 1.0* CD, SGI part number 812-1122-001, includes the online diagnostics that are available for customer use (`bridgeloc`, `olcdrom`, `olenet`, `olmem`, `olpci`, `olsio`, `oltape`, `olusb`, `olvst`, `pandora`, and `runalldiags`). This CD ships with all Silicon Graphics Fuel visual workstations. You need to install files from the CD on a system before you can run the online diagnostics. The CD booklet includes installation procedures.

The *Internal Support Tools 2.7* CD, SGI part number 812-0640-011, includes internal versions of the customer diagnostics (`bridgeloc`, `olcdrom`, `olenet`, `olmem`, `olpci`, `olsio`, `oltape`, `olusb`, `olvst`, `pandora`, and `runalldiags`) and additional online diagnostics that are available only for SGI personnel (`olperi`, `oldisk`, and `odydiags`). You need to install files from the CD on a system before you can run the online diagnostics. The CD booklet includes installation procedures.

5.2 Running the Online Diagnostics Automatically

The `runalldiags` script automatically runs a sequence of online diagnostics. It runs in three modes:

- Basic mode verifies memory and performs 30 minutes of stress testing. (If you want to perform regularly scheduled testing, use basic mode.)
- Normal mode performs the same tests as basic mode and also performs I/O testing. (The I/O testing may disrupt the serial port and USB devices.)
- Extensive mode performs more disruptive I/O testing. (Ethernet is unavailable, and USB operations are disrupted.) It also performs more intensive CPU, memory, and stress testing. Use this mode only if you suspect there is a problem with the system.

Note: The `runalldiags` script also runs the `torpedo` test. You should not manually run the `torpedo` test on Silicon Graphics Fuel visual workstations because the test requires multiple CPUs to detect failing hardware. The `torpedo` test is not documented in this manual because Silicon Graphics Fuel visual workstations do not have multiple CPUs.

Procedures for running the individual tests are described in the individual test descriptions later in this chapter.

5.2.1 Basic Mode

In basic mode, the `runalldiags` script:

- Runs `olperi` with the following command line (if the internal diagnostics are loaded):

  `/usr/diags/bin/olperi`

- Runs `olmem` with the following command line:

  `/usr/diags/bin/olmem`

  Note: Be sure to check `/var/adm/SYSLOG` for memory errors logged by the operating system.

- Runs `pandora` for 30 minutes with the following command line:

  `/usr/diags/bin/pandora -runtime 30`

5.2.2 Normal Mode

In normal mode, the `runalldiags` script:

- Runs `olperi` with the following command line (if the internal diagnostics are loaded):

  `/usr/diags/bin/olperi`

- Runs `olmem` with the following command line:

  `/usr/diags/bin/olmem`

  Note: Be sure to check `/var/adm/SYSLOG` for memory errors logged by the operating system.

- Runs `oldisk` on each mounted local disk with the following command line (if the internal diagnostics are loaded):

  `/usr/diags/bin/oldisk`

  Note: The `runalldiags` script runs `oldisk` on the disks in parallel. It starts testing all disks at the same time.

- Runs `olpci` with the following command line:

  `/usr/diags/bin/olpci -v <`*vertex*`> -nonstd`

- Runs `olenet` with the following command line:

  `/usr/diags/bin/olenet -v <`*vertex*`>`

- Runs `olsio` with the following command line:

  `/usr/diags/bin/olsio -v <`*vertex*`>`

- Runs `olusb` with the following command line:

  `/usr/diags/bin/olusb -v <`*vertex*`>`

- Runs `pandora` for 30 minutes with the following command line:

  `/usr/diags/bin/pandora -runtime 30`

5.2.3 Extensive Mode

In extensive mode, the `runalldiags` script:

- Runs `olperi` for 10 minutes with the following command line (if the internal diagnostics are loaded):

  ```
  /usr/diags/bin/olperi -runtime 10
  ```

- Runs `olmem` with the following command line:

  ```
  /usr/diags/bin/olmem REPEAT=2
  ```

  Note: Be sure to check `/var/adm/SYSLOG` for memory errors logged by the operating system.

- Runs `oldisk` on each mounted local disk with the following command line (if the internal diagnostics are loaded):

  ```
  /usr/diags/bin/oldisk -filename <file>
  ```

  Note: The `runalldiags` script runs `oldisk` on the disks in parallel. It starts testing all disks at the same time.

- Runs `olpci` for 2 minutes with the following command line:

  ```
  /usr/diags/bin/olpci -nonstd -v <vertex> -runtime 2
  ```

- Runs `olenet` with the following command line:

  ```
  /usr/diags/bin/olenet -v <vertex> -loop-int-phy
  ```

- Runs `olsio` for 10 minutes with the following command line:

  ```
  /usr/diags/bin/olsio -v <vertex> -runtime 10
  ```

- Runs `olusb` for 5 minutes with the following command line:

  ```
  /usr/diags/bin/olusb -v <vertex> -usb-inv-probe -runtime 5
  ```

- Runs `pandora` for 60 minutes with the following command line:

  ```
  /usr/diags/bin/pandora -runtime 60
  ```

- Runs `olvst` with the following command line if you enter the `-host` option:

  ```
  /usr/diags/bin/olvst -a ping -H <host>
  ```

5.2.4 Running the runalldiags script

Perform the following procedure to run the `runalldiags` script:

1. Enter the following command to change to the directory that contains the diagnostics:
   `cd /usr/diags/bin`
2. Enter the following command to start the script:
   `./runalldiags` [*options*]

Note: When you run `runalldiags` in `-normal` or `-extensive` modes, you should run it from the console. The Ethernet testing that `runalldiags` performs in `-normal` and `-extensive` modes disrupts any `telnet` sessions on the system.

Refer to Table 5-2 for descriptions of the command-line options.

Table 5-2 runalldiags Command-line Options

Option	Description
`-h | -help`	Displays help information
`-basic`	Runs the script in basic mode
`-normal`	Runs the script in normal mode (default)
`-extensive`	Runs the script in extensive mode
`-host <`*host*`>`	Specifies a system to target for network tests
`-d <`*directory*`>`	Specifies the directory that contains the online diagnostics

If a diagnostic fails, the script saves the output from the diagnostic in a file in the `/tmp` directory (for example, `/tmp/diagTestOutput.1.olenet`). Output from the script indicates the actual name of the file. When a diagnostic fails, the script continues to run the remaining diagnostics.

5.2.5 Example

The following example shows output from running `runalldiags` in basic mode:

```
shad# ./runalldiags -basic

Running online diagnostics at Basic level

Time: Mon Oct  1 10:55:53 CDT 2001
System Information: IRIX64 shad 6.5-wolfi-root-SN1O 6.5.10m 07171440 IP35
Plan on running: olmem pandora

olmem - Online Memory Diagnostic    (Check /var/adm/SYSLOG for error
message)
/usr/diags/bin/olmem
PASS(olmem)
pandora - System Stress Test
/usr/diags/bin/pandora -runtime 30
PASS(pandora)

Finished running at Mon Oct  1 11:35:38 CDT 2001
Ran: 2 Failed: 0
```

5.3 Common Command-line Options

Table 5-3 lists command-line options that are common to all online diagnostics. Use these options to modify test behavior when you run the diagnostics manually.

Note: The individual test descriptions later in this chapter describe the command-line options that are specific to each test.

Table 5-3 Common Online Diagnostic Command-line Options

Option	Description
--<*test_name*>	Prevents the specified test from running.
-all	Runs all standard tests.
-c|-cont|-continue	Ignores errors and continues testing.
-code	Displays CODE messages. This is the default.
-color	Highlights PASS messages in green, FAIL messages in red, and unresolved error messages in yellow. This is the default.
-config <*filename*>	Loads the specified configuration file.
-diag	Displays DIAG messages. This is the default.
-forever	Runs the diagnostic indefinitely.
-h|-help	Runs with an interactive help menu and causes all other command-line options to be ignored. No tests will be run.
-hrtb	Displays HRTB messages. This is the default.
-hwdebug <*level*>	Specifies the verbosity of hardware debugging information. Valid values are 0 through 5. The default is 0.
-info	Displays INFO messages. This is the default.
-interact	Runs the test interactively.
-loop	Displays LOOP messages. This is the default.
-meta	Displays META messages. This is the default.
-nocode	Does not display CODE messages.
-nocolor	Does not highlight PASS, FAIL, and unresolved error messages in different colors.
-nodiag	Does not display DIAG messages.
-noESP	Disables logging of diagnostic events to Embedded Support Partner.
-nohrtb	Does not display HRTB messages. This is the default.
-noinfo	Does not display INFO messages.
-noloop	Does not display LOOP messages.
-nometa	Does not display META messages.

Table 5-3 (continued) Common Online Diagnostic Command-line Options

Option	Description
-nonrequired	Does not automatically select tests that are required by other tests being run.
-nonstd	Runs all nonstandard tests.
-norev	Does not display REV messages.
-norslt	Does not display RSLT messages.
-notest	Does not display TEST messages.
-notime	Does not display TIME messages. This is the default.
-notrace	Does not display TRCE messages.
-operator	Provides minimal output to the screen. Limits messages to CMDL, META, LOOP, REV, RSLT, and ****ERROR messages.
-rev	Displays REV messages. This is the default.
-rslt	Displays RSLT messages. This is the default.
-runtime <*time*>	Runs the test for the specified time (in minutes).
-test	Displays TEST messages. This is the default.
-time	Displays TIME messages.
-trace	Displays TRCE messages. This is the default.
<*test_name*>=<*number*>	Runs the specified test for the specified number of times.
HWDEBUG=<*level*>	Same as -hwdebug.
INDENT_STEP=<*step*>	Indents the text by the specified number of spaces.
LOG=<*filename*>	Copies diagnostic output to the specified file.
MAXERR=<*number*>	Exits the diagnostic after the specified number of tests has failed. The default is 1.
MAX_ERRORS=<*number*>	Exits the diagnostic after the specified number of errors has occurred. The default is 20.
META=<*number*>	Prints out META information when the specified number of loops is completed.
REPEAT=<*loops*>	Runs the list of tests the specified number of times. The default is 1.
TRACE=<*filename*>	Logs TRCE messages to the specified file.
WIDTH=<*number*>	Sets the width of the diagnostic messages to the specified number; does not include the output tag. The default is 59.

5.4 Common Output

All online diagnostics begin each line of output with common output tags. These output tags make it easier to interpret the information that the test displays. Refer to Table 5-4 for a listing of all the output tags and their descriptions.

Online diagnostics display similar pass and fail output. Messages that indicate that a test has passed successfully are highlighted in green; messages that indicate that a test has failed are highlighted in red; and messages that indicate that a test did not complete or was unresolved are highlighted in yellow.

Table 5-4 Common Online Diagnostic Output Tags

Tag	Description
ABRT	Indicates an error that caused the diagnostic to unexpectedly exit.
CMDL	Displays the command line that is used to start the diagnostic.
CODE	Calls out a specific board or chip.
DIAG	Displays information about the cause of a failure.
HDBG	Displays information that is useful for debugging hardware.
HRTB	Indicates that the diagnostic is still running during time-consuming operations.
INFO	Displays general information about the hardware or diagnostic operations.
LOOP	Signals the end of a loop.
META	Displays a summary of the diagnostic. Highlighted green for pass, red for fail, and yellow for unresolved.
NOTE	Displays important diagnostic information.
REV	Displays the revision level of the diagnostic.
RSLT	Displays the result of the most recent test. Highlighted green for pass, red for fail, and yellow for unresolved.
TEST	Indicates the start of a test.
TIME	Displays the current time.
TOUT	Appears when the diagnostic exits because it passes the maximum run time.
TRCE	Indicates (traces) the code that is used when a test fails; this should not be used often by field engineers.
****ERROR	Signals that a hardware error was detected.

5.5 Test Concurrency

You can run most online diagnostics concurrently with user jobs; however, some online diagnostics are too stressful on a system to run concurrently. Refer to Table 5-5 for more information.

Table 5-5 Concurrency of Online Diagnostics with User Jobs

Test	Description	Concurrency
`bridegloc`	PCI bridge locator and diagnostic listing tool	Can run with user jobs.
`odydiags`	Graphics diagnostic	Do not run with user jobs.
`olcdrom`	CD-ROM diagnostic	Can run with user jobs that do not use the CD-ROM drive.
`oldisk`	Disk diagnostic	Can run with user jobs.
`olenet`	Ethernet diagnostic	Can run with user jobs that do not use the Ethernet port.
`olmem`	Memory diagnostic	Do not run with user jobs.
`olpci`	PCI configuration viewer and diagnostic listing tool	Can run with user jobs.
`olperi`	Random instruction diagnostic	Do not run with user jobs.
`olsio`	SIO serial diagnostic	Can run with user jobs that do not use the SIO ports under test.
`oltape`	Tape diagnostic	Can run with user jobs.
`olusb`	USB diagnostic	Can run with user jobs.
`olvst`	Socket-based network diagnostic	Can run with user jobs.
`pandora`	System stress test	Do not run with user jobs. **Note:** This test uses nearly 100% of the system resources; any user job running is resource starved, which causes extensive swapping.

5.6 CPU Instruction Test

The processor element random instruction test (`olperi`) is a directed test that checks the processor chip user mode floating-point, integer, branch, and memory load / store instructions. During each iteration, it produces random machine code sequences and stores them to random addresses in memory. It then executes the code and verifies the results by comparing the final state of the general-purpose registers (GPRs), floating-point registers (FPRs), and memory to simulated results.

5.6.1 Prerequisites for Running olperi

The `olperi` test has the following prerequisites:

- You must have root privilege.
- You must stop all user programs and other diagnostics that are running.

5.6.2 Running olperi

Perform the following procedure to run the `olperi` test:

1. Enter the following command to change to the directory that contains the diagnostics:
   `cd /usr/diags/bin`
2. Enter the following command to start the test:
   `./olperi` [*options*]

Refer to Table 5-6 for descriptions of the command-line options.

Table 5-6 olperi Command-line Options

Option	Description
`-f`	Includes the floating-point instruction set in the test vector. The default includes all instruction sets.
`-a`	Includes the integer instruction set in the test vector. The default includes all instruction sets.
`-b`	Includes the branch instruction set in the test vector. The default includes all instruction sets.
`-m`	Includes the memory load / store instruction set in the test vector. The default includes all instruction sets.
`-s` <*starting seed index*>	Generates the test vector with the indicated seed index.
`-i` <*number of instructions*>	Specifies the number of machine instructions in each test vector. Valid values are between `16` and `1024`.

Table 5-6 (continued) olperi Command-line Options

Option	Description
`-q` <*quick mode*>	Compares a checksum that represents the result from all passes instead of comparing results with all of the CPUs every iteration. The default is not quick mode. **Note:** A decrease in execution time is evident only when the `olperi` test runs on a large number of CPUs or for long periods of time.
`-il` <*number of loops*>	Selects the number of internal loops. This option is important when using the `-d` option. Setting the internal loops to 1 minimizes the dump print sessions to just one. The default is `5000`.
<`-s` *starting seed index*> `-d`	Dumps the data for all the registers and generated instructions.This option requires the `-s` option. SGI recommends that you reduce the number of instructions, if possible, by using the `-i` option, and then set the `-il` option to 1 before you use this option. SGI also recommends that you redirect dump data to a file with the `>` operator.

Example 1:

```
./olperi -s 121
```

`olperi` runs one loop of test vectors, starting at test vector (seed index) 121 (`-s 121`).

Example 2:

```
./olperi REPEAT=100
```

`olperi` runs 100 repetitions (`REPEAT=100`).

Example 3:

```
./olperi -forever -f
```

`olperi` runs test code sequences that contain only floating-point instructions (`-f`). The test runs indefinitely (`-forever`).

Example 4:

```
./olperi -il 1 -s 2371 -i 32 -d > olperi.dump
```

`olperi` runs only one internal loop (`-il 1`). `olperi` starts at test vector (seed index) 2371 (`-s 2371`) and uses 32 instructions (`-i 32`). `olperi` performs a vector dump (`-d`) of registers, memory, and instructions. This command line redirects the vector dump to a file named `olperi.dump` (`> olperi.dump`).

5.6.3 Output from olperi

The following sample shows output from a passing `olperi` test:

```
CMDL            ./olperi
TEST olperi     olperi Test                        Test(1/1), Loop(1/1)
RSLT olperi     PASS       5000 seeds tested: Mon Jul 10 11:43:10 2000
LOOP            Completed Loop 1 of 1, duration: 55.668 sec       PASS
META            ITERATION=1   PASSES         NON-PASSES
META            olperi        1              0
META            TOTAL         1              0
```

When `olperi` detects a miscompare, it outputs the initial, final, and simulated final states of the GPRs, FPRs, and memory. It also outputs a trace of the test code simulation. The test also prints `Miscompare` in the right margin of the output to indicate where the error occurred as well as the physical number of the processor with the miscompare.

If `olperi` fails, it displays the following message (highlighted red):

```
RSLT olperi     FAIL
```

5.6.4 Troubleshooting Tips for olperi

To further isolate a problem, use the following procedure:

1. Rerun `olperi` for one internal test loop. Start with the failing seed index, reduce the number of instructions to test, and eliminate multiple printings of the dump (`olperi -s <`*starting seed index*`> -i <`*number of instructions*`> -il 1`).
2. Continue to reduce the number of instructions until `olperi` passes.
3. Rerun `olperi` with the `-d` option and the `-i` option to get a dump of the instructions, register values, and memory values. Set the `-i` option to the highest number of instructions that passed `olperi`.
4. Rerun `olperi` with the `-d` option and with the `-i` option set to the lowest number of instructions that failed `olperi`. Increase instructions by one to include the failing instruction as the last generated instruction.
5. Compare the dumps to find the instruction that caused the failure. It should be the third from the last instruction in the list.

   Note: You may also find it useful to compare the register values and memory values.

5.7 Memory Test

The memory test (`olmem`) is a directed test that verifies the memory and cache components. It performs the following functions:

- Tests the high-order bits
- Detects all stuck-at faults, all coupling faults, and some pattern-sensitive faults
- Exercises all cache TAGRAM bits

The `olmem` test can test most of free memory or a specified block of memory. If it detects an error, `olmem` tests the failing page of memory to further isolate the failure.

Note: The memory test cannot test all of the memory in a system because some memory is used by the kernel.

5.7.1 Prerequisites for Running olmem

The `olmem` test has the following prerequisites:

- You must have root privilege.
- You must stop all user programs that have CPU affinity to obtain accurate results.

5.7.2 Running olmem Manually

Perform the following procedure to run the `olmem` test:

1. Enter the following command to change to the directory that contains the diagnostics:
   `cd /usr/diags/bin`
2. Enter the following command to start the test:
   `./olmem` [*options*]

Refer to Table 5-7 for descriptions of the command-line options.

Table 5-7 olmem Command-line Options

Option	Description
`-h|-help`	Runs the test with an interactive help menu. Causes other options to be ignored; no tests are run.
`-addr-pattern`	Runs the address pattern test to test memory.
`-memaddr`	Runs the moving inversion memory test.
`-tagram`	Runs a quick test that exercises the cache.
`MEM=<`*size*`>`	Specifies the amount of memory (in MB) to test. The default is 80% of free memory.

Example 1:.

```
./olmem
```

`olmem` tests most of free memory.

Example 2:

```
./olmem MEM=1024
```

`olmem` tests 1,024 MB (1 GB) of memory (`MEM=1024`).

5.7.3 Output from olmem

The following sample shows output from a passing `olmem` test:

```
REV                 olmem version 1.3 built on Nov 12 2001 at 10:02:18
INFO                Start time Mon Nov 26 04:26:11 2001
INFO                Running on IRIX64 6.5.15m 10290626 IP35 (trout)
INFO
INFO                NOTE: Most memory errors are seen by the kernel instead of
INFO                olmem. Check the console and /var/adm/SYSLOG for memory
INFO                errors after olmem is done.
INFO
CMDL                ./olmem
TEST tagram         Exercise cache tagram bits             Test(1/5), Loop(1/1)
INFO                Trying to test 239 MB out of 411 MB free with 1 cpu's
HRTB                Starting threads to lock memory into place. DONE!
HRTB                Running memory test. DONE!
RSLT tagram         PASS
TEST l1cache        L1 Cache test                          Test(2/5), Loop(1/1)
HRTB                Running memory test. DONE!
RSLT l1cache        PASS
TEST l2cache        L2 Cache test                          Test(3/5), Loop(1/1)
HRTB                Running memory test. DONE!
RSLT l2cache        PASS
TEST addr-pattern   Address pattern test (with inversion)  Test(4/5), Loop(1/1)
HRTB                Running memory test. DONE!
RSLT addr-pattern   PASS
TEST memaddr        Moving inversion memory test           Test(5/5), Loop(1/1)
HRTB                Running memory test..... DONE!
RSLT memaddr        PASS
LOOP                Completed Loop 1   of 1, duration: 148.690 sec   PASS
META                ITERATION=1       PASSES         NON-PASSES
META                tagram            1              0
META                l1cache           1              0
META                l2cache           1              0
META                addr-pattern      1              0
META                memaddr           1              0
META                TOTAL             5              0
```

5.7.4 Troubleshooting Tips for olmem

- The operating system logs errors in `/var/adm/SYSLOG` while `olmem` runs. Check `/var/adm/SYSLOG` for memory errors every time that `olmem` finishes testing memory.
- If `olmem` exits with the following message, check the specified path for memory errors:

```
ABRT    BUS ERROR: This may be due to a double bit error. Check
ABRT    /var/adm/SYSLOG
```

- The operating system logs single-bit memory errors only when the `systune` parameter `sbe_log_errors` is enabled. Enter the following command to enable this parameter:

```
systune -r sbe_log_errors 1
```

- The console reports single-bit memory errors only when the `systune` parameter `sbe_report_cons` is enabled. Enter the following command to enable this parameter:

```
systune -r sbe_report_cons 1
```

5.8 I/O Tests

The I/O tests are directed tests that verify the I/O components.

5.8.1 PCI Bridge Location Utility

The PCI bridge location utility (`bridgeloc`) locates and displays all PCI bridge vertices in the hardware graph. The output from this utility is used as an argument (or as a component of an argument) to other PCI I/O online tests (for example, `olpci` and `olenet`).

5.8.1.1 Prerequisites for Running bridgeloc

The `bridgeloc` utility has the following prerequisites:

- You must have root privilege.

5.8.1.2 Running bridgeloc Manually

Perform the following procedure to run `bridgeloc`:

1. Enter the following command to change to the directory that contains the diagnostics:
   `cd /usr/diags/bin`
2. Enter the following command to start the utility:
   `./bridgeloc`

5.8.1.3 Output from bridgeloc

The following sample shows output from the `bridgeloc` utility:

```
Mon Nov 26 04:30:53 PST 2001
IRIX64 trout 6.5-wolfi-root-SN1O 6.5.15m 10290626 IP35
REV                Online PCI Bridge Locator and Diag Listing Utility 1.4

INFO               PCI bridges were found at the following locations:

BRDG               /hw/module/001c01/Ibrick/xtalk/14/pci
BRDG               /hw/module/001c01/Ibrick/xtalk/15/pci

INFO               Devices located at the following /hw locations may be
INFO               tested using the indicated online diagnostic:
INFO               (The format of each line is a device code followed by a
INFO               partial command line for the applicable online diagnostic.
INFO               Additional command line options may be required to
INFO               properly test the device.  Please refer to the man page
INFO               for the particular diagnostic for details.  Device
INFO               codes are BRDG for PCI Bridges, ENET for ethernet
INFO               interfaces, SIO for serial ports, USB for the USB and
INFO               RTI for Real-Time Interrupts.)

BRDG               olpci -v /hw/module/001c01/Ibrick/xtalk/14/pci
BRDG               olpci -v /hw/module/001c01/Ibrick/xtalk/15/pci
ENET               olenet -v /hw/module/001c01/Ibrick/xtalk/15/pci/4
 SIO               olsio -v /hw/module/001c01/Ibrick/xtalk/15/pci/4
 RTI               olrti -v /hw/module/001c01/Ibrick/xtalk/15/pci/4
 USB               olusb -v /hw/module/001c01/Ibrick/xtalk/15/pci/5
```

5.8.2 PCI Bridge Dump Utility

The PCI bridge dump utility (`olpci`) lists the online diagnostics that are applicable to each vertex of the given PCI bridge. (The other online diagnostics use `olpci` to locate the hardware graph vertex of the BaseIO devices and other PCI devices.)

5.8.2.1 Prerequisites for Running olpci

The `olpci` utility has the following prerequisites:

- You must have root privilege.
- You must use the `bridgeloc` utility output to determine the PCI bridge vertex:

```
Mon Oct  1 10:55:53 CDT 2001
IRIX64 shad 6.5-wolfi-root-SN1O 6.5.10m 07171440 IP35
REV          Online PCI Bridge Locator and Diag Listing Utility 1.4
INFO         PCI bridges were found at the following locations:
BRDG         /hw/module/001c01/Ibrick/xtalk/14/pci
BRDG         /hw/module/001c01/Ibrick/xtalk/15/pci
...
BRDG         olpci -v /hw/module/001c01/Ibrick/xtalk/14/pci
BRDG         olpci -v /hw/module/001c01/Ibrick/xtalk/15/pci
...
```

5.8.2.2 Running olpci Manually

Perform the following procedure to run `olpci`:

1. Enter the following command to change to the directory that contains the diagnostics:
   `cd /usr/diags/bin`
2. Enter the following command to start the utility:
   `./olpci <-v|-vertex` *pci-bridge-vertex*`>` [*options*]

Refer to Table 5-8 for descriptions of the command-line options.

Table 5-8 olpci Command-line Options

Option	Description
`-pciCfgSpace`	Displays the configuration registers and attached devices.
`--pciCfgSpace`	Does not display the configuration registers and attached devices.
`-pciDiagList`	Lists online diagnostics for each hwgraph vertex. This is the default.
`--pciDiagList`	Does not list the online diagnostics for each hwgraph vertex.
`-v|-vertex` <*pci-bridge-vertex*>	Specifies the hardware graph vertex. This option is required.

5.8.2.3 Output from olpci

The following sample shows output from the `olpci` utility:

```
REV                 Online PCI Bridge Config Space (OLPCI) version 1.1 built on
REV                 Nov 12 2001 at 10:02:19
INFO                Start time Mon Nov 26 04:34:42 2001
INFO                Running on IRIX64 6.5.15m 10290626 IP35 (trout)
CMDL                ./olpci -v /hw/module/001c01/Ibrick/xtalk/14/pci
TEST pciDiagList    Online Diag List for PCI devices  Test(1/1), Loop(1/1)
INFO                Online PCI Configuration Check (OLPCI)
INFO                Online Diagnostic Listing for PCI devices attached to:
INFO                    /hw/module/001c01/Ibrick/xtalk/14/pci/controller
INFO                (Format is a partial command line for the applicable
INFO                online diagnostic. Additional options may be required.
INFO                Please see the man page for the specific diagnostic
INFO                for further details.

RSLT pciDiagList    PASS         Diagnostic Listing completed successfully.
LOOP                Completed Loop 1   of 1, duration: 0.087 sec   PASS
META                ITERATION=1       PASSES         NON-PASSES
META                pciDiagList       1              0
META                TOTAL             1              0
```

5.8.3 Ethernet Test

The Ethernet test (`olenet`) is a directed test that checks the Ethernet hardware. In all cases, `olenet` performs a basic test of the Ethernet controller; if a loopback test is not selected, it also displays information about the current status, including the MAC address, link speed, and link status.

The `olenet` test can perform internal or external loopback tests. The `olenet` test first performs external loopback tests at 10 Mbps. If the 10-Mbps test passes, the loopback test repeats at 100 Mbps. If either test fails, `olenet` attempts an internal loopback test on the PHY (physical) chip. If that test fails, `olenet` repeats the test on the IOC3 chip (Ethernet controller).

5.8.3.1 Prerequisites for Running olenet

The `olenet` test has the following prerequisites:

- You must have root privilege.
- You must stop all user programs that use the Ethernet port to receive accurate results.
- You must use the `bridgeloc` utility output to determine the Ethernet controller vertex:

```
Mon Oct  1 10:55:53 CDT 2001
IRIX64 shad 6.5-wolfi-root-SN1O 6.5.10m 07171440 IP35
REV          Online PCI Bridge Locator and Diag Listing Utility 1.4
INFO         PCI bridges were found at the following locations:
BRDG         /hw/module/001c01/Ibrick/xtalk/14/pci
BRDG         /hw/module/001c01/Ibrick/xtalk/15/pci
...
ENET         olenet -v /hw/module/001c01/Ibrick/xtalk/15/pci/4
...
```

- To run the external loopback tests, you must connect an external loopback connector (RJ45 connector that has pin 1 connected to pin 3 and pin 2 connected to pin 6).

5.8.3.2 Running olenet Manually

Note: You should run `olenet` from the console. The Ethernet testing that `olenet` performs disrupts any `telnet` sessions on the system.

Perform the following procedure to run `olenet`:

1. Enter the following command to change to the directory that contains the diagnostics:
   `cd /usr/diags/bin`
2. Enter the following command to start the test:
   `./olenet <-v|-vertex` *pci-bridge-vertex* **`| -interface`** *interface***`>`** [*options*]

Refer to Table 5-9 for descriptions of the command-line options.

Table 5-9 olenet Command-line Options

Option	Description
`-enet-info`	Displays information about the current state of the Ethernet controller. This is the default.
`-enet-probe`	Performs a simple register test of the Ethernet controller.
`-loop-ext-10`	Performs an external loopback test at 10 Mbps.
`-loop-ext-100`	Performs an external loopback test at 100 Mbps.
`-loop-int-ioc3`	Performs an internal loopback test on the IOC3 chip.
`-loop-int-phy`	Performs an internal loopback test on the PHY chip.
`PACKETS=<`*packets*`>`	Sends the specified number of packets in a loopback test. The default is `1000`.
`-v|-vertex <`*pci-bridge-vertex*`>`	Specifies the Ethernet controller location in the hardware graph. This option is required unless the `-interface` option is used.
`-interface <`*interface*`>`	Uses the specified interface for the loopback test. This option is required unless the `-vertex` option is used.

Example:

```
./olenet -loop-int-phy -interface ef0 PACKETS=2000
```

`olenet` runs an internal loopback test (`-loop-int-phy`) on ef0 (`-interface ef0`). `olenet` sends 2,000 packets instead of the default 1,000 packets (`PACKETS=2000`).

5.8.3.3 Output from olenet

The following sample shows output from a passing `olenet` test:

```
REV                 olenet version 1.3 built on Nov 12 2001 at 10:02:18
INFO                Start time Mon Nov 26 04:35:31 2001
INFO                Running on IRIX64 6.5.15m 10290626 IP35 (trout)
CMDL                olenet -v /hw/module/001c01/Ibrick/xtalk/15/pci/4
TEST enet-probe     Simple register test of IOC3           Test(1/2),
Loop(1/1)
INFO
INFO                Testing ef0 at
/hw/module/001c01/Ibrick/xtalk/15/pci/4
INFO
RSLT enet-probe     PASS
TEST enet-info      Display Ethernet Status                Test(2/2),
Loop(1/1)
INFO                MAC Address: 08:00:69:0b:c3:6b
INFO                Speed: 100 Mb/s     Full Duplex: Enabled
INFO                Link Status: Good Autonegotiation: Enabled
INFO
INFO                Device ready to transmit
INFO
RSLT enet-info      PASS
LOOP                Completed Loop 1   of 1, duration: 0.111 sec   PASS
META                ITERATION=1       PASSES         NON-PASSES
META                enet-probe        1              0
META                enet-info         1              0
META                TOTAL             2              0
```

If `olenet` detects that the Link Status is bad, it does not identify this as an error; however, you should investigate it. The following sample shows output from an `olenet` test that detected a bad Link Status:

```
REV               Online Ethernet Diagnostics 1.1
CMDL              ./olenet -vertex
/hw/module/001c01/Ibrick/xtalk/15/pci/4
TEST enet-probe   Simple register test of IOC3       Test(1/2), Loop(1/1)
RSLT enet-probe   PASS
TEST enet-info    Display Ethernet Status            Test(2/2), Loop(1/1)
INFO              MAC Address: 08:00:69:11:bd:31
INFO              Speed: 100 Mb/s     Full Duplex: Disabled
INFO              Link Status: Bad Autonegotiation: Enabled
INFO
INFO
RSLT enet-info    PASS
LOOP              Completed Loop 1   of 1, duration: 0.023 sec   PASS
META              ITERATION=1     PASSES         NON-PASSES
META              enet-probe      1              0
META              enet-info       1              0
META              TOTAL           2              0
```

If one of the following messages appears on the console during loopback tests, consider them part of normal output and do not investigate them:

```
WARNING: efo: link fail - check ethernet cable
NOTICE: efo: link ok
```

The following sample shows output from an `olenet` test that detected an error on the PHY chip:

```
Tue Jul 11 14:19:30 CDT 2000
IRIX64 ioif-sn1-b 6.5-blosure-bamboo.072099-SN1 07280743 IP35
REV               Online Ethernet Diagnostics 1.1
CMDL              ./olenet -interface ef4 -loop-int-phy -forever
CMDL              PACKETS=10000
TEST enet-probe   Simple register test of IOC3       Test(1/2), Loop(1/0)
RSLT enet-probe   PASS
TEST loop-int-phy Internal Ethernet Loopback (PHY)   Test(2/2), Loop(1/0)
TRCE              Starting Test.
**** ERROR 000004   Error receiving packet 17, seq num 7203, (Data miscompare)
INFO              Invalid sequence number
**** ERROR 000004   Error receiving packet 19, seq num 7203, (Data miscompare)
INFO              Invalid sequence number
**** ERROR 000004   Error receiving packet 93, seq num 7209, (Data miscompare)
INFO              Invalid sequence number
**** ERROR 000004   Error receiving packet 97, seq num 7209, (Data miscompare)
INFO              Invalid sequence number
**** ERROR 000004   Error receiving packet 113, seq num 7215, (Data miscompare)
INFO              Invalid sequence number
**** ERROR 000004   Error receiving packet 116, seq num 7215, (Data miscompare)
INFO              Invalid sequence number
INFO              Interface drops: 0   Socket Drops: 2968
INFO              Sent: 1251   Received: 1115  Good: 1007
INFO              Internal PHY loopback failed. Going to IOC3 loopback.
TRCE              Starting Test.
INFO              Interface drops: 0   Socket Drops: 3034
INFO              Sent: 10000  Received: 10000         Good: 10000
```

```
INFO              Showing 20 of 108 errors. Set MAX_ERRORS=108 to see all
INFO              errors
RSLT loop-int-phy FAIL
INFO              Maximum error count (1) reached
META              ITERATION=1       PASSES         NON-PASSES
META              enet-probe        1              0
META              loop-int-phy      0              1
META              TOTAL             1              1
```

5.8.4 SuperIO Port Test

The SuperIO (SIO) port (`olsio`) test runs internal loopback tests on the serial ports using the PIO and DMA transfers. By default, `olsio` runs internal loopback tests on both serial ports. `olsio` may be configured via command-line options to run specific tests on a specific port.

5.8.4.1 Prerequisites for Running olsio

The `olsio` test has the following prerequisites:

- You must have root privilege.
- To run the external loopback tests on a single port (the `-SPExt` option), ensure that you use a loopback plug on the port that you want to test.
- To run the external loopback tests between two ports (the `-DPExt` option), ensure that you use a loopback cable between the ports that you want to test.
- You must use the `bridgeloc` utility output to determine the vertex of the SIO port:

```
Mon Oct  1 10:55:53 CDT 2001
IRIX64 shad 6.5-wolfi-root-SN1O 6.5.10m 07171440 IP35
REV           Online PCI Bridge Locator and Diag Listing Utility 1.4
INFO          PCI bridges were found at the following locations:
BRDG          /hw/module/001c01/Ibrick/xtalk/14/pci
BRDG          /hw/module/001c01/Ibrick/xtalk/15/pci
...
SIO          olsio -v /hw/module/001c01/Ibrick/xtalk/15/pci/4
...
```

5.8.4.2 Running olsio Manually

Perform the following procedure to run the `olsio` test:

1. Enter the following command to change to the directory that contains the diagnostics:
   `cd /usr/diags/bin`
2. Enter the following command to start the test:
   `./olsio <-v|-vertex` *ioc3-vertex*`>` [*options*]

Refer to Table 5-10 for descriptions of the command-line options.

Table 5-10 olsio Command-line Options

Option	Description
`-v|-vertex` <*ioc3-vertex*>	Tests the specified hardware graph vertex of the IOC3 ASIC that controls the desired SIO port. This option is required.
`-I|-Internal`	Runs the internal loopback tests. This is the default. **Note:** If you want to run both internal and external tests, ensure that you use the `-Internal` option with the external test option that you want to run.

Table 5-10 (continued) olsio Command-line Options

Option	Description
`-SPExt|-SinglePortExternal`	Runs the external loopback tests on a single port. **Note:** Ensure that you use a loopback plug on the port that you want to test.
`-DPExt|-DualPortExternal`	Runs the external loopback tests between ports A and B of the same SIO card. **Note:** Ensure that you use a loopback cable between the ports that you want to test.
`-p|-passes` <*number-of-passes*>	Runs the test for the specified number of passes. The default is `10`.
`-rs232`	Selects only RS-232 mode. The default is `-rs232` and `-rs422`.
`-rs422`	Selects only RS-422 mode. The default is `-rs232` and `-rs422`.
`-A`	Runs the selected tests on Port A. The default is `-A` and `-B` (if Port B exists).
`-B`	Runs the selected tests on Port B. The default is `-A` and `-B` (if Port B exists).
`-PIO`	Runs by using only PIO transfers. The default is `-PIO` and `-DMA`.
`-DMA`	Runs by using only DMA transfers. The default is `-PIO` and `-DMA`.
`<-DPExt|-DualPortExternal -rs232>` `-f|-flowcontrol`	Turns on flow control for the PIO handshaking test. Use this option with the `-DualPortExternal` options and the `-rs232` option.
`-b|-baud` <*baud-rate*>	Sets the desired baud rate. The default is 460 Kbps.

5.8.4.3 Output from olsio

If `olsio` passes, it displays the following message (highlighted green):

```
RSLT sioTest      PASS
```

If `olsio` fails, it displays the following message (highlighted red):

```
RSLT sioTest      FAIL
```

The following sample shows passing output from the `olsio` test:

```
REV                Online SIO Port Test (OLSIO) version 3.1 built on Nov
12
REV                2001 at 10:02:19
INFO               Start time Mon Nov 26 04:37:11 2001
INFO               Running on IRIX64 6.5.15m 10290626 IP35 (trout)
CMDL               ./olsio -v /hw/module/001c01/Ibrick/xtalk/15/pci/4
TEST sioTest       SIO Port Diagnostic              Test(1/1),
Loop(1/1)
INFO               Testing SIO port(s) located at
INFO               /hw/module/001c01/Ibrick/xtalk/15/pci/4
```

```
INFO                Port A, ttyd3, will be tested.
INFO                Port B, ttyd4, will be tested.

INFO                Testing Internal Loopback in PIO mode
TRCE                Initializing SIO UARTA...
TRCE                Initializing SIO UARTB...
INFO                Baud rate: requested = 115000, actual = 114583
INFO                Baud rate: requested = 115000, actual = 114583
TRCE                Testing Port A, ttyd3, (PIO mode), pass = 1
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Testing Port B, ttyd4, (PIO mode), pass = 1
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Initializing SIO UARTA...
TRCE                Initializing SIO UARTB...
INFO                Baud rate: requested = 115000, actual = 114583
INFO                Baud rate: requested = 115000, actual = 114583
TRCE                Testing Port A, ttyd3, (PIO mode), pass = 2
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Testing Port B, ttyd4, (PIO mode), pass = 2
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Initializing SIO UARTA...
TRCE                Initializing SIO UARTB...
INFO                Baud rate: requested = 115000, actual = 114583
INFO                Baud rate: requested = 115000, actual = 114583
TRCE                Testing Port A, ttyd3, (PIO mode), pass = 3
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Testing Port B, ttyd4, (PIO mode), pass = 3
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Initializing SIO UARTA...
TRCE                Initializing SIO UARTB...
INFO                Baud rate: requested = 115000, actual = 114583
INFO                Baud rate: requested = 115000, actual = 114583
TRCE                Testing Port A, ttyd3, (PIO mode), pass = 4
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Testing Port B, ttyd4, (PIO mode), pass = 4
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Initializing SIO UARTA...
TRCE                Initializing SIO UARTB...
INFO                Baud rate: requested = 115000, actual = 114583
INFO                Baud rate: requested = 115000, actual = 114583
TRCE                Testing Port A, ttyd3, (PIO mode), pass = 5
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Testing Port B, ttyd4, (PIO mode), pass = 5
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Initializing SIO UARTA...
TRCE                Initializing SIO UARTB...
INFO                Baud rate: requested = 115000, actual = 114583
INFO                Baud rate: requested = 115000, actual = 114583
TRCE                Testing Port A, ttyd3, (PIO mode), pass = 6
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Testing Port B, ttyd4, (PIO mode), pass = 6
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Initializing SIO UARTA...
TRCE                Initializing SIO UARTB...
INFO                Baud rate: requested = 115000, actual = 114583
INFO                Baud rate: requested = 115000, actual = 114583
TRCE                Testing Port A, ttyd3, (PIO mode), pass = 7
```

```
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Testing Port B, ttyd4, (PIO mode), pass = 7
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Initializing SIO UARTA...
TRCE                Initializing SIO UARTB...
INFO                Baud rate: requested = 115000, actual = 114583
INFO                Baud rate: requested = 115000, actual = 114583
TRCE                Testing Port A, ttyd3, (PIO mode), pass = 8
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Testing Port B, ttyd4, (PIO mode), pass = 8
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Initializing SIO UARTA...
TRCE                Initializing SIO UARTB...
INFO                Baud rate: requested = 115000, actual = 114583
INFO                Baud rate: requested = 115000, actual = 114583
TRCE                Testing Port A, ttyd3, (PIO mode), pass = 9
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Testing Port B, ttyd4, (PIO mode), pass = 9
TRCE                Internal Loopback:wrote FIFO, now reading
TRCE                Initializing SIO UARTA...
TRCE                Initializing SIO UARTB...
INFO                Baud rate: requested = 115000, actual = 114583
INFO                Baud rate: requested = 115000, actual = 114583
TRCE                Testing Port A, ttyd3, (PIO mode), pass = 10
TRCE                Internal Loopback:wrote FIFO, now reading
DIAG       000000   Port A, ttyd3, PASSED Loopback (PIO mode)
TRCE                Testing Port B, ttyd4, (PIO mode), pass = 10
TRCE                Internal Loopback:wrote FIFO, now reading
DIAG       000000   Port B, ttyd4, PASSED Loopback (PIO mode)
DIAG       000000   PASSED Internal Loopback (PIO mode)

INFO                Testing Internal Loopback in DMA mode
TRCE                Initializing SIO UARTA...
TRCE                Initializing SIO UARTB...
INFO                Baud rate: requested = 115000, actual = 114583
INFO                Baud rate: requested = 115000, actual = 114583
HRTB                UARTA DMA in progress (pass = 1).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTA DMA in progress (pass = 2).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTA DMA in progress (pass = 3).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTA DMA in progress (pass = 4).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTA DMA in progress (pass = 5).
```

```
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTA DMA in progress (pass = 6).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTA DMA in progress (pass = 7).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTA DMA in progress (pass = 8).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTA DMA in progress (pass = 9).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTA DMA in progress (pass = 10).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
DIAG       000000   Port A, ttyd3, PASSED Loopback (DMA mode)
HRTB                UARTB DMA in progress (pass = 1).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTB DMA in progress (pass = 2).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTB DMA in progress (pass = 3).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTB DMA in progress (pass = 4).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTB DMA in progress (pass = 5).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTB DMA in progress (pass = 6).
TRCE                DMA transmission complete!
```

```
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTB DMA in progress (pass = 7).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTB DMA in progress (pass = 8).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTB DMA in progress (pass = 9).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
HRTB                UARTB DMA in progress (pass = 10).
TRCE                DMA transmission complete!
HRTB                Continuing
TRCE                DMA reception complete!
HRTB                Continuing DONE!
DIAG       000000   Port B, ttyd4, PASSED Loopback (DMA mode)
DIAG       000000   PASSED Internal Loopback (DMA mode)

RSLT sioTest        PASS         sioTest PASSED.
LOOP                Completed Loop 1   of 1, duration: 118.512 sec   PASS
META                ITERATION=1       PASSES         NON-PASSES
META                sioTest           1              0
META                TOTAL             1              0
```

5.8.5 USB Port Test

The USB port (`olusb`) test verifies a USB host controller and optionally checks the devices that are attached to that controller. The `olusb` test has the following subtests:

- *Host controller check* verifies the basic functionality of the host controller by performing a series of read / write tests on the registers of the controller.
- *Inventory probe* checks each device that is connected to the host controller and displays information about each device and the current topology of the bus. Certain problems with the USB device are detected as well, such as devices that are not responding or failed ports that are not providing power on a USB hub.

By default, `olusb` runs the host controller check test.

5.8.5.1 Prerequisites for Running olusb

The `olusb` test has the following prerequisites:

- You must have root privilege.
- You must use the `bridgeloc` utility output to determine the vertex of the USB controller:

```
Mon Oct  1 10:55:53 CDT 2001
IRIX64 shad 6.5-wolfi-root-SN1O 6.5.10m 07171440 IP35
REV           Online PCI Bridge Locator and Diag Listing Utility 1.4
INFO          PCI bridges were found at the following locations:
BRDG          /hw/module/001c01/Ibrick/xtalk/14/pci
BRDG          /hw/module/001c01/Ibrick/xtalk/15/pci
...
USB          olusb -v /hw/module/001c01/Ibrick/xtalk/15/pci/6
...
```

5.8.5.2 Running olusb Manually

Perform the following procedure to run the `olusb` test:

1. Enter the following command to change to the directory that contains the diagnostics:
   `cd /usr/diags/bin`
2. Enter the following command to start the test:
   `./olusb <-v|-vertex` *vertex*`>` [*options*]

Refer to Table 5-11 for descriptions of the command-line options.

Table 5-11 olusb Command-line Options

Option	Description
`-h|-help`	Runs with an interactive help menu. Causes all other command-line options to be ignored. No tests will be run.
`-usb-hc-check`	Runs the host controller check test. This is the default. **Note:** The USB bus is not available to the system while this test is running, but it is available once the test is finished.

Table 5-11 (continued) olusb Command-line Options

Option	Description
`-usb-inv-probe`	Runs the inventory probe test. The default is `-usb-hc-check`. **Note:** The USB bus is not available to the system while this test is running and it may not be available when the test is complete. (This would affect any keyboards or mice that are attached via USB.)
`<-usb-inv-probe> SAVE=<`*filename*`>`	Saves the contents and state of the bus to the specified file. This file and the `COMPARE` option can be used during a later run for comparison. Use this option with the `-usb-inv-probe` option.
`<-usb-inv-probe> COMPARE=<`*filename*`>`	Compares the current state and topology of the bus with the state saved in the specified file. Any differences will be flagged as an error. Use this option with the `-usb-inv-probe` option.
`-v|-vertex <`*vertex*`>`	Specifies the hardware graph vertex of the USB controller. This option is required. **Note:** The `bridgeloc` utility can be used to find the *vertex*.

5.8.5.3 Output from olusb

If `olusb` passes, it displays one or both of the following messages (highlighted green):

```
RSLT usb-hc-check      PASS
RSLT usb-inv-probe     PASS
```

If `olusb` fails, it displays one or both of the following messages (highlighted red):

```
RSLT usb-hc-check     FAIL
RSLT usb-inv-probe     FAIL
```

The following sample shows passing output from the `olusb` test:

```
REV                 olusb version 1.1.2 built on Nov 12 2001 at 10:02:14
INFO                Start time Mon Nov 26 04:41:36 2001
INFO                Running on IRIX64 6.5.15m 10290626 IP35 (trout)
CMDL                ./olusb -v /hw/module/001c01/Ibrick/xtalk/15/pci/5
TEST usb-hc-check   USB Host Controller Register Checkout  Test(1/1),
Loop(1/1)
INFO                Device conforms to OpenHCI version 17.0
RSLT usb-hc-check   PASS
LOOP                Completed Loop 1   of 1, duration: 0.589 sec   PASS
META                ITERATION=1       PASSES         NON-PASSES
META                usb-hc-check      1              0
META                TOTAL             1              0
```

5.8.5.4 Troubleshooting Tips for olusb

- You may experience problems with devices that are attached via the USB after running `olusb` with the `-usb-inv-probe` option. Unplug and reconnect these devices to correct this problem.
- USB error messages on the console from the operating system are normal while `olusb` runs. Ignore these messages and refer to the `olusb` output for the correct information.

5.9 Storage Tests

5.9.1 CD-ROM Test

The CD-ROM (`olcdrom`) test verifies that a system can access a CD in the CD-ROM drive. If you create a diagnostic data CD and insert it in the CD-ROM drive before you run the test, the test also verifies that CD read operations return the correct data.

5.9.1.1 Prerequisites for Running olcdrom

The `olcdrom` test has the following prerequisites:

- You must have root privilege.
- To verify that CD read operations return the correct data, you must create a diagnostic data CD. (Refer to the `-gendata` description in Table 5-12.)

5.9.1.2 Running olcdrom

Perform the following procedure to run the `olcdrom` test:

1. Enter the following command to change to the directory that contains the diagnostics:
   `cd /usr/diags/bin`
2. Insert a CD into the CD-ROM drive.

   Note: To test CD read operations, insert a CD that contains diagnostic data (created with the `-gendata` option).
3. Enter the following command to start the test:
   `./olcdrom <-device` *`device`***`>`** [*`options`*]

Refer to Table 5-12 for descriptions of the command-line options.

Table 5-12 olcdrom Command-line Options

Option	Description
`-h|-help`	Runs with an interactive help menu. All other command-line options are ignored and no tests are run.
`-device <`*`device`*`>`	Specifies the CD-ROM device to test. If the device contains a CD with diagnostic data (created with the `-gendata` option), the test reads the data from the CD and compares it to expected data. If the device contains a normal CD, the test simply verifies that it can access the CD-ROM.
`-rpass <`*`passcount`*`>`	Specifies the number of passes to perform during the random data pattern section of the test.
`-apass <`*`passcount`*`>`	Specifies the number of passes to perform during the random access section of the test.

Table 5-12 (continued) olcdrom Command-line Options

Option	Description
`-patterns` <*patterns*>	Specifies the data patterns to use. *patterns* is a bitmask of selected patterns to run: 0x001: All 1's pattern 0x002: All 0's pattern 0x004: Odd-bit pattern 0x008: Even-bit pattern 0x010: Single-bit pattern (16 patterns total) 0x020: Walking 1 and 0 bit patterns with 8-bit boundaries (16 patterns total) 0x040: Walking 1 and 0 bit patterns with 16-bit boundaries (32 patterns total) 0x080: Walking 1 and 0 bit patterns with 32-bit boundaries (64 patterns total) 0x100: Walking 1 and 0 bit patterns with 64-bit boundaries (128 patterns total) 0x200: Random data patterns 0x400: Random accesses The default setting is 0x60f, which selects the ones, zeros, odd bits, even bits, and random data patterns with random access.
`-gendata`	Creates a CD image file (`olcdrom.1`) that contains data for the test to use. If you insert a CD that contains this data into the CD-ROM drive before you run the `olcdrom` test, the test reads the data from the CD and compares it to expected values. Use a third-party CD writer to create a CD with the image file. **Note:** You can use the `-device` parameter with the `-gendata` parameter to specify an alternate name for the CD image file.

Example 1:

```
./olcdrom -device /dev/scsi/sc0d410
```

`olcdrom` tests the CD-ROM drive that is located at /dev/scsi/sc0d410.

Example 2:

```
./olcdrom -gendata
```

`olcdrom` creates a CD image file (`olcdrom.1`) that contains diagnostic data. Use a third-party CD writer to create a CD with the image file.

5.9.1.3 Output from olcdrom

If `olcdrom` passes, it displays the following message (highlighted green):

```
RSLT olcdrom      PASS
```

If `olcdrom` fails, it displays the following message (highlighted red):

```
RSLT olcdrom      FAIL
```

The following example shows error output from `olcdrom`:

```
klsys7 6# ./olcdrom -device /dev/scsi/sc5d1l0
REV                 olcdrom version 1.0 built on Jun  5 2002 at 07:25:22
INFO                Start time Mon Jun 10 12:15:05 2002
INFO                Running on IRIX64 6.5.17m 05230726 IP35 (klsys7)
INFO                Compiled on Jun  4 2002 at 19:07:11
CMDL                ./olcdrom -device /dev/scsi/sc5d1l0
TEST olcdrom        CDROM confidence test
Test(1/1), Loop(1/1)
INFO                CD-ROM      TEAC          CD-532E-B         1.0A
INFO                device block size:        2048 bytes
INFO                device block count:     332725 blocks
INFO                SGI diagnostic data located at block 29, offset 0
INFO                testing zeros...
INFO                testing ones...
INFO                testing evens...
INFO                testing odds...
INFO                testing random data, pass 0/4
**** ERROR 000000   miscompare:
INFO                00222220 EXP: 20fc645721dc3429
INFO                         ACT: 20fc005721dc3429
INFO                         DIF: 0000640000000000
INFO                testing random data, pass 1/4
**** ERROR 000000   miscompare:
INFO                00222218 EXP: 20fc645721dc3429
INFO                         ACT: 20fc005721dc3429
INFO                         DIF: 0000640000000000
INFO                testing random data, pass 2/4
**** ERROR 000000   miscompare:
INFO                00222210 EXP: 20fc645721dc3429
INFO                         ACT: 20fc005721dc3429
INFO                         DIF: 0000640000000000
INFO                testing random data, pass 3/4
**** ERROR 000000   miscompare:
INFO                00222208 EXP: 20fc645721dc3429
INFO                         ACT: 20fc005721dc3429
INFO                         DIF: 0000640000000000
INFO                testing random access, pass 1/2...
INFO                testing random access, pass 2/2...
RSLT olcdrom        FAIL         one or more tests failed as
indicated
INFO                Maximum error count (1) reached
```

5.9.1.4 Troubleshooting Tips for olcdrom

- If `olcdrom` indicates that the CD-ROM device does not appear ready after loading a CD, verify that the CD is properly seated in the CD tray.

5.9.2 Disk Test

The disk (`oldisk`) test checks a system disk by writing data patterns to the disk, reading them back, and comparing the data. By default, `oldisk` accesses the drive asynchronously.

Note: The `oldisk` test does not work over the Network File System (NFS).

5.9.2.1 Prerequisites for Running oldisk

The `oldisk` test has the following prerequisite:

- You must have root privilege.

5.9.2.2 Running oldisk

Perform the following procedure to run the `oldisk` test:

1. Enter the following command to change to the directory that contains the diagnostics:
   `cd /usr/diags/bin`
2. Enter the following command to start the test:
   `./oldisk <-filename` *file*`>` [*options*]

Refer to Table 5-13 for descriptions of the command-line options.

Table 5-13 oldisk Command-line Options

Option	Description
`-filename <`*file*`>`	Uses the specified file for testing.
`-h|-help`	Runs with an interactive help menu. All other command-line options are ignored and no tests are run.
`-filesize <`*size*`>`	Uses the specified amount of disk space (in megabytes) for testing. The default is 128.
`-async-direct`	Accesses the drive asynchronously, bypassing the internal buffer of the operating system. This is the default.
`-async-buffered`	Accesses the drive asynchronously, using the internal buffer of the operating system.
`-sync-direct`	Accesses the drive synchronously, bypassing the internal buffer of the operating system.
`-sync-buffered`	Accesses the drive synchronously, using the internal buffer of the operating system.
`-patterns <`*patterns*`>`	Uses the specified data patterns. Valid values are zeros, ones, checker, quadword, halfword, counting, and random; separate lists with commas. The default is all.
`-paranoid`	Double-checks every data buffer as it is filled. **Note:** This option prevents a memory error from showing up as a disk error, but it slightly increases execution time.

Example 1:

```
./oldisk -filename /usr/disktestfile
```

`oldisk` tests the filesystem that is mounted under `/usr` with a file called `disktestfile` (`-filename /usr/disktestfile`). `oldisk` uses the default file size of 128 MB.

Example 2:

```
./oldisk -filename /usr/disktestfile -filesize 64 -patterns
ones,random
```

`oldisk` tests the filesystem that is mounted under `/usr` with a file called `disktestfile` (`-filename /usr/disktestfile`). `oldisk` uses a file size of 64 MB (`-filesize 64`). `oldisk` uses a data pattern that contains all ones and a data pattern that contains random data (`-patterns ones,random`).

5.9.2.3 Output from oldisk

If `oldisk` passes, it displays the following message (highlighted green):

```
RSLT async_direct      PASS
```

If `oldisk` fails, it displays the following message (highlighted red):

```
RSLT async_direct      FAIL
```

5.9.2.4 Troubleshooting Tips for oldisk

If you suspect that a memory error could show up as a disk error, use the `-paranoid` option.

5.9.3 Tape Test

The tape (`oltape`) test exercises any online tape devices that have been configured into the operating system. When you use devices, the block-special node in `/dev/rmt` is accessed directly.

5.9.3.1 Prerequisites for Running oltape

The `oltape` test has the following prerequisites:

- You must have root privilege.
- To run the `eot` or `tapemark` test, you must configure down the tape device that you want to test.

5.9.3.2 Running oltape Manually

Perform the following procedure to run the `oltape` test:

1. Enter the following command to change to the directory that contains the diagnostics:
   `cd /usr/diags/bin`
2. Enter the following command to start the test:
   `./oltape <-d|-device` *Tape Device*`>` [*options*]

Refer to Table 5-14 for descriptions of the command-line options.

Table 5-14 oltape Command-line Options

Option	Description
`-d|-device` <*Tape Device*>	Performs the I/O operations on the specified tape device. This option is required.
`-rw`	Writes a number of blocks, rewinds the tape, reads back the same number of blocks, and compares them to expected data values. The default is `rw`, `thrash`, `tapemark`, `block`, and `eot`.
`-read`	Reads a number of blocks and compares them to the expected data values. The default is `rw`, `thrash`, `tapemark`, `block`, and `eot`. **Note:** If you are using random block sizes (`BLOCKMIN` and `BLOCKMAX` are not equal), the SEED value must be set to the process ID value that was returned by the `-write` option.
`-write`	Writes a number of blocks. The default is `rw`, `thrash`, `tapemark`, `block`, and `eot`. **Note:** If you are using random block sizes (`BLOCKMIN` and `BLOCKMAX` are not equal), the SEED value must be set to the process ID value that was returned by the `-write` option.
`-thrash`	Writes a single block, rewinds the tape, reads back the block, and compares the block to the expected data values. This sequence is repeated for each pass. The default is `rw`, `thrash`, `tapemark`, `block`, and `eot`.

Table 5-14 (continued) oltape Command-line Options

Option	Description
`-eot`	Writes data until the end of the tape is reached, rewinds the tape, reads back all the blocks, and compares the blocks to the expected data values. The default is `rw`, `thrash`, `tapemark`, `block`, and `eot`. **Note:** This test is available only when operating on a tape device that has been configured down.
`-block`	Uses block positioning operations to skip around the tape while writing, reading, and comparing data values. The default is `rw`, `thrash`, `tapemark`, `block`, and `eot`.
`-tapemark`	Skips around the tape to write, read, and compare data values. The default is `rw`, `thrash`, `tapemark`, `block`, and `eot`. **Note:** This test is available only when operating on a tape device that has been configured down.
`<-rw|-write|-read>` `PASSMAX=<`*n*`>|` `PASSMIN=<`*n*`>|` `PASSSTEP=<`*n*`>`	Writes / reads the specified number of blocks. Use this option with the *-rw*, *-read*, and *-write* options. The default is 1 pass. `PASSMAX` specifies the ending pass number. `PASSMIN` specifies the starting pass number. `PASSSTEP` increases the pass count by the specified amount until `PASSMAX` is reached. **Note:** The pass option specifications must be the same when reading and writing tapes. **Note:** *PASSMIN* and *PASSSTEP* are most useful when specified with the *+blockispass* option.
`BLOCKMAX=<`*n*`>|` `BLOCKMIN=<`*n*`>|` `BLOCKSTEP=<`*n*`>`	Uses the specified block size for I / O operations. The default is a block size of 4,096. `BLOCKMAX` specifies the maximum block size. `BLOCKMIN` sets the block size to a value between `BLOCKMIN` and `BLOCKMAX` for each pass. `BLOCKSTEP` increases the block size by the specified amount for each pass.
`ECHOFILE <`*echo-file-name*`>`	Writes the first data buffer to the specified file. No actual I / O to the device is performed.
`DIFFFILE <`*diff-file-name*`>`	Writes differences in the case of data compare errors to the specified file.
`PATTERN=<`*pattern*`>`	Uses the specified data pattern. The following values are valid *pattern* values: `bits` sets each word with a random sequence of consecutive 1 bits. `slide0` clears bit 0 and sets all other bits of the first word; subsequent words are circularly left-shifted by 1 bit. `slide1` sets bit 0 and clears all other bits of the first word; subsequent words are circularly left-shifted by 1 bit. `random` sets each word to a random value. `all` uses all patterns, one per pass. This is the default. Setting the *pattern* to a numeric constant sets each word based on the number that you specify.

Table 5-14 (continued) oltape Command-line Options

Option	Description
TIMEOUT=<*n*>	Uses the specified time-out value when asynchronous I/O is performed. The default is 30 seconds.
SEED=<*n*>	Uses the specified seed value to calculate block sizes for reads. **Note:** If you are using random block sizes (BLOCKMIN and BLOCKMAX are not equal), the seed value must be set to the process ID value that was returned by the -write option.
<PASSMAX=*n* | PASSMIN=*n* | PASSSTEP=*n*> +blockispass	Controls block sizes for functions *-rw*, *-read*, *-write*. Use this option with the *PASSMAX*, *PASSMIN*, and *PASSSTEP* options.
<PASSMAX=*n* | PASSMIN=*n* | PASSSTEP=*n*> +blocklimits	Uses the minimum and maximum block size returned from the device. Use this option with the *PASSMAX*, *PASSMIN*, and *PASSSTEP* options.
+fast	Does not perform data compares.
-h	Displays the command-line synopsis.

Example 1:

./oltape -d <*tape-device*>

oltape performs the default read/write, thrash, tapemark, block, and end-of-tape tests on the specified <*tape-device*> (-d <*tape-device*>). You must configure the tape device down to run the tapemark and end-of-tape tests.

Example 2:

./oltape --tapemark --eot -d <*tape-device*>

oltape performs only the read/write, thrash, and block tests on the specified <*tape-device*>. (The --tapemark option specifies that oltape should not run the tapemark test; the --eot option specifies that oltape should not run the end-of-tape test.)

5.9.3.3 Output from oltape

oltape displays META tags that indicate the progress of the test. The TOTAL line summarizes the number of passing and non-passing subtests.

5.10 Network Test

The TCP socket test (`olvst`) is a socket-based exerciser that uses TCP sockets to send and receive data across a network. The `olvst` test runs in two modes:

- *Ping mode* uses ICMP to send data packets to a remote system, which then returns the packets to the local system. The local system verifies that the returned data matches the data that was sent to the remote system.
- *Read/write mode* reads and writes data patterns between two systems. It compares the data that is sent from the remote system with the data that is received by the local system.

If `olvst` fails, the failing hardware is the network interface that it tested. The failing FRU is the board that contains the failing network interface.

5.10.1 Prerequisites for Running olvst

The `olvst` test has the following prerequisites:

- You must have root privilege.

5.10.2 Running olvst Manually

Perform the following procedure to run `olvst`:

1. Enter the following command to change to the directory that contains the diagnostics:
   `cd /usr/diags/bin`
2. Enter the following command to start the test:
   `./olvst` [*options*]

Refer to Table 5-15 for descriptions of the command-line options.

Table 5-15 olvst Command-line Options

Option	Description
`-a|-action <`*action*`>`	Performs the specified action. The following values are valid *action* values: `ping` elicits ECHO_RESPONSE packets from the host. The block size is limited to 16,376 bytes with this option. `rw` reads and writes data on the same machine. This option requires the `-options verbose` option. `read` reads data from the cooperating process. This option requires the `-options verbose` option. `write` writes data to the specified host. Ensure that you start a `read` action before you start a `write` action.

Table 5-15 (continued) olvst Command-line Options

Option	Description
`-b|-blocksize` <*blocksize-max* `[:`*blocksize-min* `[:`*blocksize-step*`]] | pass>`	Uses the specified block size for I/O operations. *blocksize-max* specifies the maximum block size. This value must be greater than 0. The default is 4,096. *blocksize-min* sets the block size to a value between *blocksize-min* and *blocksize-max* for each pass. *blocksize-step* increases the block size by the specified amount for each pass until *blocksize-max* is reached. `pass` sets the block size to the current pass count value. **Note:** The block size must be the same on both the receiving and sending hosts.
`-e|-echofile <`*echo-file-name*`>`	Writes the first data buffer to the specified file and then exits; no I/O is performed.
`-H|-Host` <*host-name*`> [:`*port-number*`]`	Specifies the host on which the cooperating process is located. The default is the machine on which the test is running. The *port-number* specifies the TCP/IP port to use.
`-o|-options <`*options*`>`	Turns on the list of options separated by colons. There is no default. The following values are valid *option* values: `fast` does not compare data. `verbose` displays the progress of the test. Use this option with the `rw` or `read` actions.
`-p|-passes` <*pass-end*`> [:`*pass-start* `[` `:`*pass-step*`] ]`	Performs the specified number of passes. *pass-end* specifies the ending pass number. *pass-start* specifies the starting pass number. *pass-step* increases the pass count by the specified amount until *pass-end* is reached. **Note:** The number of passes must be the same on both the receiving and sending hosts.
`-P|-pattern <`*pattern*`>`	Uses the specified data pattern. The following values are valid *pattern* values: `bits` sets each byte to have a sequence of consecutive 1 bits. `slide0` clears bit 0 and sets all other bits of the first byte; subsequent words are circularly left-shifted by 1 bit. `slide1` sets bit 0 and clears all other bits of the first byte; subsequent words are circularly left-shifted by 1 bit. `random` sets each byte to a random value. `all` uses the `bits`, `slide0`, `slide1`, and `random` patterns, one per pass. This is the default. Setting *pattern* to a numeric constant sets each word to the specified number. **Note:** The pattern must be the same on both the receiving and sending hosts.
`TIMEOUT=<`*n*`>`	Uses the specified time-out value when asynchronous I/O is performed.

Example 1:

```
./olvst -a ping -H host1.sgi.com -p 10000
```

`olvst` sends 10,000 (`-p 10000`) ping operations (`-a ping`) to host1.sgi.com (`-H host1.sgi.com`).

Example 2:

```
./olvst -a read -p 1000 -b pass -H host1.sgi.com
```

`olvst` runs a read process (`-a read`) that reads from host1.sgi.com (`-H host1.sgi.com`) by using blocksizes that are equal in size to the pass count (`-b pass`) for 1,000 passes (`-p 1000`). The read and write processes should be run on different hosts (for example, host1.sgi.com and host2.sgi.com) to prevent the TCP layer from avoiding hardware testing.

Note: There must be an `olvst` write process on the remote system. A sample write process to run on host1.sgi.com for this example is: `olvst -a write -p 1000 -b pass -H host2.sgi.com`

5.10.3 Output from olvst

The following sample shows output from a passing `olvst` test:

```
Mon Jul 10 11:54:14 CDT 2000
IRIX64 klsys7 6.5 07070640 IP35
REV               Online Socket-Based Network Test 1.0 (Compiled Jul 7 2000
REV               00:34:58)
CMDL              ./olvst
TEST olvst        Socket-Based Network Test          Test(1/1), Loop(1/1)
DIAG       000000 Executing Read/Write Test
DIAG       000000 The total number of reads = 1
DIAG       000000 The total number of bytes read  = 4096
DIAG       000000 Average number of bytes per read = 4096
DIAG       000000 Time taken to do all reads = 0.000188s
DIAG       000000 speed = 174304316.39bps = 174304.316Kbps =
DIAG       000000 174.3043Mbps
RSLT olvst        PASS          olvst reading process PASSED in rw mode.
LOOP              Completed Loop 1   of 1, duration: 0.010 sec       PASS
META              ITERATION=1   PASSES         NON-PASSES
META              olvst         1              0
META              TOTAL         1              0
RSLT olvst        PASS           olvst writing process PASSED in rw mode.
LOOP              Completed Loop 1   of 1, duration: 0.013 sec       PASS
META              ITERATION=1   PASSES          NON-PASSES
META              olvst         1              0
META              TOTAL         1              0
```

If `olvst` fails, it displays the following message (highlighted red):

```
RSLT olvst        FAIL
```

5.11 System Stress Test

The pandora test is a programmable system-level stress test that simulates heavy user loads on a system by testing several hardware components simultaneously. This test stresses the entire system, including:

- Processor (integer and floating-point unit)
- Memory
- I/O devices (SCSI devices, etc.)
- Network devices (Ethernet, etc.)
- Router interconnect hardware
- Graphics (multipipe system) hardware

5.11.1 Prerequisites for Running pandora

The pandora test has the following prerequisites:

- You must have root privilege.
- You must stop all user programs or other diagnostics to receive accurate results.

5.11.2 Running pandora Manually

Perform the following procedure to run the pandora test:

1. Enter the following command to change to the directory that contains the diagnostics:
   cd /usr/diags/bin
2. Enter the following command to start the test:
   ./pandora [*options*]

Refer to Table 5-16 for descriptions of the command-line options.

Table 5-16 pandora Command-line Options

Option	Description
-d	Enters debug mode in which you can modify the sysinfo file; does not generate a new sysinfo file. **Note:** Ensure that all of the directories defined in the sysinfo file exist, otherwise an error will occur.
-dpca	Dumps the precomputed array.
-h|-help	Prints the help menu.
-g <-mount>	Stops after generating the sysinfo file.
-mount	Mounts optional drives.
<-runtime 0> -tloops <*tloops*>	Executes the specified number of test loops. This option requires that the -runtime option be set to 0.

Table 5-16 (continued) pandora Command-line Options

Option	Description
`-runtime <`*time* `-mount>`	Runs the test for the specified time (in minutes). This option requires the `-mount` option.
`-v`	Prints the `pandora` version and exits.

Example 1:

```
./pandora -runtime 30 -mount
```

`pandora` runs for 30 minutes and mounts additional drives.

Example 2:

```
./pandora PERCENT=50
```

`pandora` tests 50 percent of the available memory.

Example 2:

```
./pandora -runtime 0 -tloops 100
```

`pandora` runs for 100 test loops.

5.11.3 Output from pandora

The following sample shows output from a failing `pandora` test:

```
CMDL             pandora
TEST pandora     System level stress test.          Test(1/1), Loop(1/1)
INFO             Pandora start time: Wed Jul 12 10:53:34 2000
REV              Pandora version (3.0) compiled on Jul 12 2000 10:52:24
INFO             A new "sysinfo" file is created.
INFO             Pandora run time: 30 minutes
INFO             PID0x883 DIO_VECTOR PASSED
INFO             PID0x87d DIO_RANDOM PASSED
INFO             PID0x887 MEM_RD_WR PASSED
INFO             PID0x882 MEM_RD_ONLY PASSED
INFO             PID0x880 MEM_RD_WR PASSED
INFO             PID0x885 MEM_RD_WR PASSED
INFO             PID0x878 GFX_PIXEL PASSED
INFO             PID0x889 FPU_RAND_GEN PASSED
INFO             PID0x86e FPU_RAND_GEN PASSED
INFO             PID0x88a FPU_MATRIX_INV PASSED
INFO             PID0x875 FPU_MATRIX_INV PASSED
INFO             PID0x86f Process activity report:
INFO             PID0x86f MEM PROCESS TYPE
INFO             PID0x882 Test loops 247 Process loops 4 MEM_RD_ONLY
INFO             PID0x880 Test loops 51 Process loops 4 MEM_RD_WR
INFO             PID0x885 Test loops 70 Process loops 4 MEM_RD_WR
INFO             PID0x887 Test loops 65 Process loops 4 MEM_RD_WR
INFO             PID0x86f IO PROCESS TYPE
INFO             PID0x883 Test loops 37 Process loops 4 DIO_VECTOR
```

```
INFO              PID0x87d Test loops 40 Process loops 4 DIO_RANDOM
INFO              PID0x86f FPU PROCESS TYPE
INFO              PID0x889 Test loops 19 Process loops 4 FPU_RAND_GEN
INFO              PID0x88a Test loops 36 Process loops 4 FPU_MATRIX_INV
INFO              PID0x86e Test loops 50 Process loops 4 FPU_RAND_GEN
INFO              PID0x875 Test loops 41 Process loops 4 FPU_MATRIX_INV
INFO              PID0x86f GFX PROCESS TYPE
INFO              PID0x878 Test loops 37 Process loops 4 GFX_PIXEL
INFO              PID0x86f NTWK PROCESS TYPE
RSLT pandora      PASS         PID0x86f Pandora PASSED
INFO              PID0x86f Pandora has completed
INFO              Pandora end time: Wed Jul 12 11:23:55 2000
```

If a data miscompare occurs, `pandora` prints out the unit number and the controller number that were involved along with the expected and actual data. The following sample output is from a `pandora` test that detected a memory failure:

```
NOTE         This diagnostic can NOT be run concurrent with any user jobs
CMDL              ./pandora
TEST pandora      System level stress test.          Test(1/1), Loop(1/1)
REV               Pandora version 3.2 built on Mar 27 2001 at 11:35:37
INFO              Start time Wed Mar 28 02:06:16 2001
INFO              Running on IRIX64 6.5.12f 03270658 IP35 (paver)
INFO              Generating a new "sysinfo" file.
INFO              Pandora run time: 30 minutes
DIAG      000001 PID0xa03 memOOTFAddrSeqIncDgenVect FAILED
DIAG      000001 PID0xa03 was running on Cpu 1
DIAG      000001 PID0xa03 was using data size 3 ull
DIAG      000001 PID0xa03 Aaddr block size 20 Ull read 5
DIAG      000001 PID0xa03 Start addr 0x1357b7c0 End addr 0x16a0a640
DIAG      000001 PID0xa03 Addr 0x1357c300 Expt 0xffffffff Recv 0x1fffffff
DIAG      000002 PID0xa03 Syndrome 0x100000000
DIAG      000000 PID0xa03 Page 0x1357c000 is in CACHED and NOT_PINED mode
DIAG      000002 PID0xa03 mop FAILED
DIAG      000002 PID0xa03 Tloops 50 Ploop 0
DIAG      000002 PID0xa03 Time to failure Hours(0) Minutes(0) Seconds(50)
DIAG      000002 PID0xa03 memExerciseFunction FAILED
INFO              PID0xa03 ERROR MEM_RD_WR FAILED
INFO              PID0x968 DIO_VECTOR PASSED
INFO              PID0xa0c MEM_RD_WR PASSED
INFO              PID0xa10 FPU_MATRIX_INV PASSED
INFO              PID0xa06 MEM_RD_WR PASSED
INFO              PID0xa11 FPU_MATRIX_INV PASSED
INFO              PID0x966 DIO_RANDOM PASSED
INFO              PID0xa09 MEM_RD_ONLY PASSED
INFO              PID0xa0f GFX_PIXEL PASSED
INFO              PID0x965 FPU_RAND_GEN PASSED
INFO              PID0xa0d FPU_RAND_GEN PASSED
INFO              PID0xa05 Process activity report:
INFO              PID0xa05 MEM PROCESS TYPE
INFO              PID0xa09 Test loops 250 Process loops 4 MEM_RD_ONLY
INFO              PID0xa03 Test loops 51 Process loops 4 MEM_RD_WR
INFO              PID0xa0c Test loops 70 Process loops 4 MEM_RD_WR
INFO              PID0xa06 Test loops 69 Process loops 4 MEM_RD_WR
INFO              PID0xa05 IO PROCESS TYPE
INFO              PID0x968 Test loops 39 Process loops 4 DIO_VECTOR
INFO              PID0x966 Test loops 42 Process loops 4 DIO_RANDOM
```

```
INFO              PID0xa05 FPU PROCESS TYPE
INFO              PID0x965 Test loops 20 Process loops 4 FPU_RAND_GEN
INFO              PID0xa10 Test loops 38 Process loops 4 FPU_MATRIX_INV
INFO              PID0xa0d Test loops 50 Process loops 4 FPU_RAND_GEN
INFO              PID0xa11 Test loops 45 Process loops 4 FPU_MATRIX_INV
INFO              PID0xa05 GFX PROCESS TYPE
INFO              PID0xa0f Test loops 35 Process loops 4 GFX_PIXEL
INFO              PID0xa05 NTWK PROCESS TYPE
RSLT pandora      FAIL          PID0xa05 Pandora FAILED
INFO              PID0xa05 Pandora has completed
INFO              Pandora end time: Wed Jul 12 13:35:42 2000
INFO              Maximum error count (1) reached
```

5.11.4 Troubleshooting Tips for pandora

- If you encounter problems while running `pandora`, try the following solutions:
  - If the system hangs, perform a non-maskable interrupt (NMI) and analyze the dump.
  - If the system crashes, check the SYSLOG for a FRU analysis.
  - If the problem is with the memory and directory DIMMs, check the SYSLOG for error-correction code (ECC) errors while running `pandora`.
  - If the problem is with the routers and links, use `linkstat` while running `pandora` or check the SYSLOG for excessive check-bit or sequencing errors.
- If `pandora` fails, the error might be the result of one of the following conditions:
  - Data miscompares, mount failures, or timeouts could be I/O-related errors.
  - Data miscompares could also be floating-point unit errors or graphics-related errors.
  - Miscompare, lost, duplicate, or damaged packets could be network-related errors.
- The operating system logs single-bit memory errors only when the `systune` parameter `sbe_log_errors` is enabled. Enter the following command to enable this parameter:

  `systune -r sbe_log_errors 1`
- The console reports single-bit memory errors only when the `systune` parameter `sbe_report_cons` is enabled. Enter the following command to enable this parameter:

  `systune -r sbe_report_cons 1`

5.12 Graphics Test

The `odydiags` test is a directed test that verifies the VPro graphics hardware. Refer to the *VPro Odyssey Graphics Diagnostic Reference*, publication number 108-0314-00x, for more information about `odydiags`.

Appendix A

POD Mode

POD mode is an interactive mode that uses the command-line interpreter that is in the PROM. You can use POD mode on a crashed system to:

- Manually run several of the power-on diagnostics
- Set and clear PROM environment variables and PROM logs
- Enable / disable the CPU
- Flash firmware
- Run the FRU analyzer

A.1 Entering POD Mode

The system enters POD mode when any of the following conditions occur:

- A power-on diagnostic fails.
- An unexpected interrupt occurs while the PROM code is executing.
- The boot sequence detects that the boot stop point virtual debug switches (switches 4 and 5) are set to stop the boot process (refer to Table A-1).

Table A-1 Boot Stop Point Debug Switch Settings

HEX	5	4	Boot Stop Point
00 00	Off	Off	Do not stop boot
00 10	On	Off	Local
00 08	Off	On	Global
00 18	On	On	Memoryless

- You issue a nonmaskable interrupt (NMI) at the PROM level.
- You issue two NMIs at the kernel level.

  The first NMI takes the system to the PROM level; the second NMI takes the system to the POD prompt.
- You type the **pod** command at the command monitor prompt (for example, >>**pod**).

A.2 Dirty Exclusive, Uncached, and Cached Modes

POD mode can run in three modes: dirty exclusive (Dex) mode, uncached (Unc) mode, or cached (Cac) mode.

A.2.1 Dirty Exclusive Mode

In dirty exclusive (Dex) mode, the power-on diagnostics require the fewest system resources to run. Dex mode does not require memory; it accesses the program code directly from the PROM and uses the data cache in the processors as the memory for its stack. Dex mode does not use the secondary cache. Nonmaskable interrupts and uncaught exceptions typically return you to the Dex mode prompt.

Dex mode is very slow because PROM instruction fetches are slow. Avoid performing long memory tests or flashing remote PROMs from Dex mode.

When you load or store data in memory or run memory tests in Dex mode, ensure that you do not access cached memory addresses.

A.2.2 Uncached Mode

In uncached (Unc) mode, the power-on diagnostics place the program code, data, and stack into main memory. This mode is similar to Cac mode, but Unc mode runs slower; however, program execution is quicker in Unc mode than in Dex mode.

A.2.3 Cached Mode

In cached (Cac) mode, the power-on diagnostics place the program code, data, and stack into the secondary cache. Cached mode is available only after memory has been probed and configured. POD execution is much quicker in Cac mode than in Dex mode or Unc mode because it runs from the onboard secondary cache.

A.3 POD Mode Prompt

The POD mode prompt has the following format:

- The node number and CPU that you are communicating with
- The NASID (NUMA Address Space ID)
- The port that you are communicating with
- The current POD mode

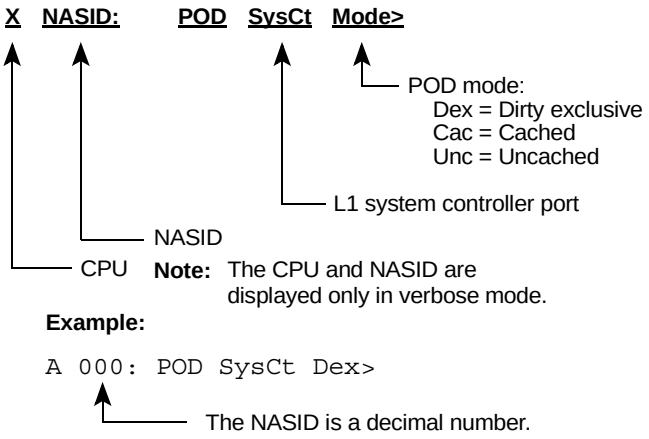

Figure A-1 POD Prompt

A.4 POD Mode Commands

Table A-2 lists the POD mode commands that you can enter at the `POD>` prompt.

Table A-2 POD Mode Commands

Command	Description
`?` [*cmdname*]	Display help information.
`adline` <*line*>	Dump the specified data cache line.
`adtag` <*line*>	Dump the specified data cache tag.
`ailine` <*line*>	Dump the specified instruction cache line.
`aitag` <*line*>	Dump the specified instruction cache tag.
`altregs` <*num*>	Use the alternate registers.
`asline` <*line*>	Dump the specified secondary cache line.
`astag` <*line*>	Dump the specified secondary cache tag.
`btrace [epc sp]`	Stack backtrace.

Table A-2 (continued) POD Mode Commands

Command	Description
`call <addr> [a0 [a1 [a2 [a3]]]]`	Call a subroutine.
`clear`	Clear errors.
`clearlog [n:<node>]`	Clear the log only. Does not clear variables and aliases.
`cfg [n:<node>]`	Display the processor's config register.
`crb [n:<node>]`	Dump I/O interface (II) coherent resource buffers (CRBs).
`crbx [n:<node>]`	Dump the CRBs in a 137-column-wide format.
`dbg [virt_val]`	Display or change the virtual debug switch settings.
`delay <microsec>`	Delay command execution.
`dgbrdg [mn | mh | mm] [n<nasid>] [b<bus>]`	Run the bridge test.
`dgconf [mn | mh | mm] [n<nasid>] [b<bus>]`	Run the PCICONFIG test.
`dgpci [mn | mh | mm] [n<nasid>] [b<bus>] [p<pcinum>]`	Run the PCI bus test.
`dgpckm [mn | mh | mm]`	Run the keyboard and mouse test
`dgsdma [mn | mh | mm | mx] [n<nasid>] [b<bus>] [p<pcinum>]`	Run the serial DMA test.
`dgspio [mn | mh | mm] [n<nasid>] [b<bus>] [p<pcinum>]`	Run the serial programmed I/O test.
`dgxbow [mn | mh | mm] [n<nasid>]`	Run the XBOW test.
`dips`	Read and display the virtual debug switch settings.
`dirinit <start> <len>`	Initialize a range of memory to the "Unowned" state.
`dirstate [state [base [len]]]`	Scan a range of addresses for the directory state that you specify.
`dirtest <start> <len>`	Perform directory test and initialization.
`dis <addr> [count]`	Disassemble the data stored in the specified memory locations.
`disable n:<node> [slice/banks]`	Disable the specified memory. (Do not disable the CPU.)
`dline <line>`	Dump the specified data cache line.
`dmc [n:<node>]`	Dump the memory configuration.
`dtag <line>`	Dump the specified data cache tag.

Table A-2 (continued) POD Mode Commands

Command	Description
`dumpspool [n:<node> <slice>]`	Dump the PI error spool.
`ecc [on | off]`	Enable or disable ECC mode.
`echo "<string>"`	Display a string on the console.
`edump_bri [n:<node>]`	Dump bridge error information to memory.
`enable n:<node> [slice/banks]`	Enable the specified CPU or memory.
`error`	Display errors.
`error_dump`	Dump the error information.
`exec [segname [flag]]`	Load and execute a segment.
`flush`	Flush and invalidate caches.
`for (<cmd>;<expr>;<cmd>) <cmd>`	Repeat the specified command(s) for the number of iterations determined by the specified expression.
`fru [1 | 2]`	Run the FRU analyzer program.
`go dex | unc | cac`	Set memory mode.
`help [cmdname]`	Display help information.
`if (<expr>) <cmd>`	Execute the specified command(s) depending on the value of a conditional.
`iline <line>`	Dump the specified instruction cache line.
`im [byte]`	Set the processor interrupt mask.
`initalllogs`	Initialize all PROM logs for all modules in the system.
`initlog [n:<node>]`	Initialize the PROM log.
`inval [i] [d] [s]`	Invalidate cache(s).
`itag <line>`	Dump the specified instruction cache tag.
`jump <addr> [a0 [a1]]`	Invalidate cache and continue execution at a specific address.
`kdebug [stackaddr]`	Debug the kernel.
`kern_sym`	Use the kernel symbol table.
`la <addr> [count]`	Load byte data and display it as ASCII characters.
`lb <addr> [count]`	Load a byte.
`ld <addr> [count]`	Load a doubleword.
`lh <addr> [count]`	Load a halfword.
`log [n:<node>] [skip [count]]`	Display entries from the PROM log for the specified node.
`loop <cmd>`	Repeat the specified command(s) forever.
`lw <addr> [count]`	Load a word.

Table A-2 (continued) POD Mode Commands

Command	Description
`maxerr <count>`	Test the error limit.
`mbist <addr>`	Run the memory self-test.
`memcmp <dst> <src> <len>`	Compare data (by bytes) in one memory location to data in another memory location.
`memcpy <dst> <src> <len>`	Copy memory (by bytes) from one memory location to another memory location.
`meminit <start> <len>`	Clear memory.
`memset <dst> <byte> <len>`	Fill memory with a byte pattern.
`memsum <src> <len>`	Add data (by bytes).
`memtest <start> <len>`	Perform memory sanity test.
`modnic`	Display the NIC value of the current module.
`module [num]`	Display or change a module number.
`nic [n:<node>]`	Display the hub NIC.
`nm <addr>`	Look up a PROM address.
`nmi n:<node> [a]`	Send a nonmaskable interrupt (NMI) to a node.
`pa <addr> [bitno]`	Print an address.
`partition [num]`	Get or set a hardware partition number.
`pb <expr>`	Print the value of an expression in binary.
`pcfg [n:<node>]`	Dump the pcfg structure.
`pd <expr>`	Print the value of an expression in decimal.
`pf [regno]`	Print the contents of a COP1 floating-point register.
`po <expr>`	Print the value of an expression in octal.
`pr [gprname [val | ~]]`	Print the contents of a register.
`printenv [n:<node>] [var]`	Print the value of the specified environment variable(s).
`px <expr>`	Print the value of an expression in hexadecimal.
`qual [on | off]`	Enable or disable quality mode.
`reconf`	Initialize the memory / directory section (MD) of the hub.
`repeat <count> <cmd>`	Repeat count.
`reset`	Reset the system.
`santest <addr>`	Perform the memory sanity test.
`sb <addr> [val [count]]`	Store a byte.
`scandir <addr> [len]`	Scan directory states.

Table A-2 (continued) POD Mode Commands

Command	Description
`sd <addr> [val [count]]`	Store a doubleword.
`sdv <addr> <count>`	Store a doubleword and verify it.
`segs [flag]`	List the segments.
`setenv [n:<node>] <var> ["<string>"]`	Set an environment variable.
`sf <regno> <val>`	Store the contents of a COP1 floating-point register.
`sh <addr> [val [count]]`	Store a halfword.
`sleep <sec>`	Sleep for a specified number of seconds.
`sline <line>`	Dump the specified secondary cache line.
`sr <grpname> <val>`	Store the contents of a register.
`stag <line>`	Dump the specified secondary cache tag.
`sstag <line> <taglo> <taghi> [way]`	Store a cache tag.
`sscache <line> <taglo> <taghi>`	Store a secondary cache doubleword.
`sw <addr> [val [count]]`	Store a word.
`talk [n:<node> a | b | c | d]`	Use the net UART.
`tdisable n:<node> [slice]`	Temporarily disable the specified CPU.
`time <cmd>`	Perform benchmark timing.
`tlbc [index]`	Clear the translation lookaside buffer (TLB).
`tlbr [index]`	Read the TLB.
`unsetenv [n:<node>] <var>`	Remove an environment variable.
`verbose [0 | 1]`	Specify whether or not the power-on diagnostics run in verbose mode.
`version`	Display the PROM version.
`while (<expr>) <cmd>`	Repeat the specified command(s) while the specified expression evaluates to true.
`why`	Display the current exception and NMI status.

Table A-3 lists POD mode commands that are disabled on Silicon Graphics Fuel visual workstations.

Table A-3 Disabled POD Mode Commands

Command	Description
`bist le | ae | lr | ar [n:<node>]`	Run the hub self-test.
`chklink`	Run the local link connection test.
`cpu [[n:<node>] a | b | c | d]`	Switch to the specified CPU.
`disc`	Discover the NUMAlink interconnect.
`fflash <node> [...]`	Prompt for processor-specific values to flash.
`flash <node> [...]`	Program the remote PROM.
`hubsde`	Run the hub send data error test.
`node [[vec] id]`	Get or set a node.
`rbist le | ae | lr | ar [n:<node>]`	Run the router self-test.
`rnic [vec]`	Read the router NIC.
`route [vec <node>]`	Set up a route.
`rstat <vec>`	Dump and clear the router status.
`rtab [vec]`	Dump the route table.
`rtrsde`	Run the router send data error test.
`slave`	Go into slave mode.
`softreset n:<node>`	Perform a soft reset on a node.
`vr <vec> <vaddr>`	Perform a vector read operation.
`vw <vec> <vaddr> <val>`	Perform a vector write operation.
`vx <vec> <vaddr> <val>`	Perform a vector exchange operation.

Reader Comment Form

Title: Silicon Graphics Fuel™ Visual Workstation Diagnostic Reference Manual

Number: 108-0350-002

Your feedback on this publication will help us provide better documentation in the future. Please take a moment to answer the few questions below.

For what purpose did you primarily use this document?

_____Troubleshooting
_____Tutorial or introduction
_____Reference information
_____Classroom use
_____Other - please explain

_____

Using a scale from 1 (poor) to 10 (excellent), please rate this document on the following criteria and explain your ratings:

_____Accuracy _____
_____Organization _____
_____Readability _____
_____Physical qualities (binding, printing, page layout) _____
_____Amount of diagrams and photos _____
_____Quality of diagrams and photos _____

Completeness (Check one and explain your answer)

_____Too much information _____ Too little information _____Correct amount

_____

_____

_____

You may write additional comments in the space below. Mail your comments to the address below, fax them to us at +1 715 726 4353, or e-mail them to us at *spt@sgi.com.* When possible, please give specific page and paragraph references. We will respond to your comments in writing within 48 hours.

_____

_____

_____

NAME _____
JOB TITLE _____
E-MAIL ADDRESS _____
SITE/LOCATION _____
TELEPHONE _____
DATE _____

Attn: Service Publications and Training
890 Industrial Boulevard
P.O. Box 4000
Chippewa Falls, WI 54729-0078
USA