Online Diagnostics
(SGI™ Origin™ 200 System; Silicon Graphics® Onyx2® System; SGI™ 2000 Series; and SGI™ 3000 Family)

Document Number 108-0286-005

CONTRIBUTORS
Written by Stacey Delcore and Darrin Goss
Edited by Allison Gosbin
Production by Rhonda Kunsman
Engineering contributions by Gregoire Banderet, Gary Davidson, Jason Godfrey, Jeff Keopp, Roberto Romano, and Lisa Steinmetz

Record of Revision

Version	Description
001	August 2000 Original printing.
002	October 2000 Updated for IRIX 6.5.10.
003	January 2001 Updated for IRIX 6.5.11.
004	April 2001 Updated for IRIX 6.5.12. Includes SGI Origin 200 system, Silicon Graphics Onyx2 system, and SGI 2000 series tests.
005	July 2001 Updated for IRIX 6.5.13. Adds information about the `runalldiags` script and adds example command lines for several of the diagnostics.

Contents

Tables

Chapter 1

Introduction

This document describes online diagnostics for the following systems: SGI Origin 200; Silicon Graphics Onyx2; SGI 2000 series; and SGI 3000 family. This chapter explains how to install the diagnostics; lists the test categories, availability of each test, common command-line options, and common test output tags; and provides concurrency information on each test.

1.1 About 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.

1.2 Installing Online Diagnostics

The *Internal Support Tools 2.5* CD, part number 812-0640-009, contains all of the files that you need to install the online diagnostics. The online diagnostics are located in the `field_diags_irix` image file.

Refer to the *Silicon Graphics Internal Support Tools 2.5 CD Installation Instructions*, part number 007-3516-009, for detailed information about installing the online diagnostics on SGI Origin 200, SGI 2000 series, Silicon Graphics Onyx2, or SGI 3000 family systems.

1.3 Test Categories

The online diagnostic tests are divided into the following categories:

- CPU tests (`olperi` and `torpedo`)
- Memory test (`olmem`)
- Router test (`olrtr` and `linktest`)
- I/O tests (`bridgeloc`, `olpci`, `olenet`, `olsio`, and `olusb`)
- Storage device tests (`oldisk` and `oltape`)
- Network device tests (`olgsn`, `olvst`, and `olvht`)
- Real-time interrupt test (`olrti`)
- System stress tests (`pandora` and `grizzly`)

1.4 Availability of Online Diagnostics

Table 1-1 lists the availability of each online diagnostic by processor type for the SGI Origin 200, SGI 2000 series, Silicon Graphics Onyx2, and SGI 3000 family systems.

Table 1-1 Online Diagnostic Test Availability (by IP Type)

	IP Type			
Diagnostic Test	**IP27 (2000 Series & Onyx2)**	**IP29 (Origin 200)**	**IP31 (2000 Series & Onyx2)**	**IP35 (3000 Family)**
`bridgeloc`				x
`grizzly`	x		x	x
`linktest`	x	x	x	
`oldisk`	x	x	x	x
`olenet`				x
`olgsn`				x
`olmem`	x			x
`olpci`				x
`olperi`	x		x	x
`olrti`	x	x	x	x
`olrtr`				x
`olsio`				x
`oltape`	x	x	x	x
`olusb`				x

Table 1-1 (continued) Online Diagnostic Test Availability (by IP Type)

Diagnostic Test	IP Type			
	IP27 (2000 Series & Onyx2)	IP29 (Origin 200)	IP31 (2000 Series & Onyx2)	IP35 (3000 Family)
olvht	x	x	x	x
olvst	x	x	x	x
pandora	x	x	x	x
torpedo	x	x	x	x

1.5 Common Command-line Options

Table 1-2 lists command-line options that are common to all online diagnostics except `linktest`. Use these options to modify test behavior.

Note: The individual test descriptions in Chapter 2 describe the command-line options that are specific to each test.

Table 1-2 Common Online Diagnostic Command-line Options

Option[a]	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.

Table 1-2 (continued) Common Online Diagnostic Command-line Options

Option[a]	Description
`-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.
`-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 have failed. The default is 1.
`MAX_ERRORS=<`*number*`>`	Exits the diagnostic after the specified number of errors have occurred. The default is 20.

Table 1-2 (continued) Common Online Diagnostic Command-line Options

Option[a]	Description
`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.

a. These options are not available with the router link test (`linktest`).

1.6 Common Output

All online diagnostics, except `linktest`, 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 1-3 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 1-3 Common Online Diagnostic Output Tags

Tag[a]	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.

Table 1-3 (continued) Common Online Diagnostic Output Tags

Tag[a]	Description
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.

a. These tags are not available with the router link test (`linktest`).

1.7 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. You should not run user jobs when you run the stress tests because the stress tests use extensive resources. Refer to Table 1-4 for more information.

Table 1-4 Concurrency of Online Diagnostics with User Jobs

Test	Description	Concurrency
bridegloc	PCI bridge locator and diagnostic listing tool	Can run with user jobs.
grizzly	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.
linktest	Router link diagnostic	Do not run with user jobs.
oldisk	Disk diagnostic	Can run with user jobs.
olenet	IO7-based Ethernet diagnostic	May run with user jobs that do not use the IO7-based Ethernet port.
olgsn	GSN diagnostic	Can run with user jobs.
olmem	Memory diagnostic	Avoid running with user jobs that have CPU affinity. **Note:** Performance degrades relative to the amount of memory under test.
olpci	PCI configuration viewer and diagnostic listing tool	Can run with user jobs.
olperi	Random instruction diagnostic	Can run with user jobs if you limit the number of CPUs to test. **Note:** The default is to run on all processors.

Table 1-4 (continued) Concurrency of Online Diagnostics with User Jobs

Test	Description	Concurrency
`olrti`	Real-time interrupt diagnostic	May run with user jobs that do not use the RTI ports under test.
`olrtr`	Router diagnostic	Do not run with user jobs. **Note:** False failures could occur if run with other jobs.
`olsio`	SIO serial diagnostic	May 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.
`olvht`	HIPPI diagnostic	May run with user jobs that do not use the HIPPI interface.
`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.
`torpedo`	Floating-point diagnostic	Do not run with other user jobs. **Note:** Consumes 70% of the CPU activity and performs extensive swapping that degrades performance.

1.8 runalldiags Script

The `runalldiags` script provides an easy way to run a predefined set of online diagnostic tests. It runs in three modes:

- Basic mode verifies CPUs and 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. (I-brick 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 System Verification Program (SVP) also provides a robust method to run online diagnostics. SVP includes more option settings than the `runalldiags` script. If you need more control than `runalldiags` provides, use SVP.

1.8.1 Basic Mode

In basic mode, the `runalldiags` script:

- Runs `olperi` with the following command line:

  `/usr/diags/bin/olperi`

- Runs `torpedo` for 10 minutes with the following command line:

  `/usr/diags/bin/torpedo -runtime 10`

- 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`

1.8.2 Normal Mode

In normal mode, the `runalldiags` script:

- Runs `olperi` with the following command line:

  `/usr/diags/bin/olperi`

- Runs `torpedo` for 10 minutes with the following command line:

  `/usr/diags/bin/torpedo -runtime 10`

- 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:

  `/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` on each I brick and P brick with the following command line:

  `/usr/diags/bin/olpci -v <vertex> -nonstd`

- Runs `olenet` on each I brick:

  `/usr/diags/bin/olenet -v <vertex>`

- Runs `olsio` on each I brick with the following command line:

  `/usr/diags/bin/olsio -v <vertex>`

  Note: The `runalldiags` script runs `olsio` on all I bricks in parallel. It starts testing all I bricks at the same time.

- Runs `olrti` with the following command line:

  `/usr/diags/bin/olrti -v <vertex> -internal`

- Runs `olusb` on each I brick 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`

1.8.3 Extensive Mode

In extensive mode, the `runalldiags` script:

- Runs `olperi` for 10 minutes with the following command line:

  `/usr/diags/bin/olperi -runtime 10`

- Runs `torpedo` with the following command line:

  `/usr/diags/bin/torpedo`

- 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:

  `/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 on each I brick and P brick with the following command line:

  `/usr/diags/bin/olpci -nonstd -v <vertex> -runtime 2`

- Runs `olenet` on each I brick with the following command line:

  `/usr/diags/bin/olenet -v <vertex> -loop-int-phy`

- Runs `olsio` for 10 minutes on each I brick with the following command line:

  `/usr/diags/bin/olsio -v <vertex> -runtime 10`

  Note: The `runalldiags` script runs `olsio` on the I bricks in parallel. It starts testing all I bricks at the same time.

- Runs `olrti` for 5 minutes with the following command line:

  `/usr/diags/bin/olrti -v <vertex> -internal -nohrtb -runtime 5`

- Runs `olusb` for 5 minutes on each I brick 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*`>`

1.8.4 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*]

Refer to Table 1-5 for descriptions of the command-line options.

Table 1-5 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 running the remaining diagnostics.

1.8.5 Example

The following example shows output from running runalldiags in basic mode:

```
roman# runalldiags -basic

Running online diagnostics at Basic level
Time: Tue Jun 19 11:04:21 CDT 2001
System Information: IRIX64 roman 6.5-ALPHA-1287375420 6.5.13f 06010821
IP35
Plan on running: olperi torpedo olmem pandora

olperi - CPU Diagnostic
/usr/diags/bin/olperi
PASS(olperi)
torpedo - CPU Floating Point Unit Diagnostic
/usr/diags/bin/torpedo -runtime 10
PASS(torpedo)
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 Tue Jun 19 11:52:36 CDT 2001
Ran: 4  Failed: 0
```

Chapter 2

Test Descriptions

This chapter describes the online diagnostics that are available for SGI Origin 200, Silicon Graphics Onyx2, SGI 2000 series, and SGI 3000 family systems. It includes descriptions of the tests, how to run them, test output, and troubleshooting tips.

2.1 CPU Tests

The CPU tests include directed tests and stress tests that verify the CPU hardware in a C brick.

2.1.1 Processor Element Random 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.

2.1.1.1 Prerequisites for Running olperi

The `olperi` test has the following prerequisites:

- Your system must have an IP27, IP31, or IP35 processor.
- You must have root privilege.
- You must stop all user programs and other diagnostics on the CPUs that you want to test. SGI recommends that you limit the number of CPUs that will be tested by using the `-r` option.

2.1.1.2 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 2-1 for descriptions of the command-line options.

Table 2-1 olperi Command-line Options

Option	Description
`-r` <*processor list*>	Runs the test on the specified processors. The default is all processors. Separate lists with commas and ranges with the word `to`.
`-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`.
`-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 `-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 through one loop of test vectors, starting at test vector (seed index) 121 (`-s 121`). `olperi` tests all CPUs because the `-r` option is not specified.

Example 2:

```
./olperi -r 0 REPEAT=100
```

`olperi` tests CPU 0 (`-r 0`) for 100 repetitions (`REPEAT=100`).

Example 3:

```
./olperi -forever -r 0to7,21to24,30 -f
```

`olperi` tests CPUs 0 to 7, 21 to 24, and 30 (`-r 0to7, 21to24, 30`). The test code sequences contain only floating-point instructions (`-f`). The test runs indefinitely (`-forever`).

Example 4:

```
./olperi -r 4 -il 1 -s 2371 -i 32 -d > olperi.dump
```

`olperi` tests CPU 4 (`-r 4`) with 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`).

2.1.1.3 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
```

2.1.1.4 Troubleshooting Tips for olperi

To further isolate a problem, use the following procedure:

1. Rerun `olperi` for one internal test loop on the failing CPU. Start with the failing seed index, reduce the number of instructions to test, and eliminate multiple printings of the dump (`olperi -r <`*processor list*`> -s <`*starting seed index*`> -i <`*number of instructions*`> -il 1`).

   Note: Use the `hinv -v` command to locate the failing CPU.

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.

2.1.2 CPU Instruction Test

The CPU instruction test (`torpedo`) is a floating-point unit (FPU) stress test. It uses several standard floating-point algorithms to test the FPU in each CPU. It compares all of the results and reports any miscompares. The `torpedo` test verifies basic functionality before it performs complex operations.

The `torpedo` test includes the following test sections:

- *MPFacilities*: verifies that each processor gets the same result based on a predetermined floating-point constant
- *FpuBasics*: checks the add, subtract, multiply, divide, and square root operations
- *ParallelOps*: uses loops that maximize the execution unit utilization
- *MathFunctions*: checks trigonometric operations
- *RootFinding*: solves equations by using the Newton-Raphson and bisection algorithms
- *MatrixOps*: solves linear algebraic equations

The `torpedo` test runs each section once and then randomly executes the test sections. As it completes each test section, it compares the results from all CPUs. If this test fails, the failing FRU is the C brick that contains the failing CPU.

2.1.2.1 Prerequisites for Running torpedo

The `torpedo` test has the following prerequisites:

- Your system must have a IP27, IP29, IP31, or IP35 processor.
- You must have root privilege.
- You must stop all user programs or other diagnostics to receive accurate results.

2.1.2.2 Running torpedo

Perform the following procedure to run the `torpedo` 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:
   `./torpedo` [*options*]

Refer to Table 2-2 for descriptions of the command-line options.

Table 2-2 torpedo Command-line Options

Option	Description
`-onfailure <`*action*`>`	Defines how `torpedo` should react to failures. The following are valid *action* values: `abort` exits immediately without doing any clean-up. `cont` continues testing. This is the default. `loop` repeats the miscompare loop forever. `repeat` repeats the last loop. `stop` stops testing and performs clean-up.
`-nologseeds`	Does not dump the random seeds to a file. The default dumps random seeds to a file.
`<-runtime 0> -rloops <`*n*`>`	Executes the specified number of test loops. The value `0` runs each test section once and does not perform random testing. The default is `1000`.
`-ncpus <`*n*`>`	Tests the specified number of CPUs. The default is the number of usable CPUs in the system.
`-iseed <`*n*`>`	Specifies the initial seed value. The default is `1`.
`-showallmis`	Prints all miscompares in a test loop.
`-tconfig <`*file*`>`	Loads the specified configuration file.

Example 1:

```
./torpedo -ncpus 15
```

`torpedo` runs on 15 CPUs (`-ncpus 15`).

Example 2:

```
./torpedo -dumpcfg mycfg
```

`torpedo` dumps the test configuration in file `mycfg` (`-dumpcfg mycfg`) and then exits.

Example 3:

```
./torpedo -tconfig mycfg
```

`torpedo` loads the configuration parameters stored in file `mycfg` (`-tconfig mycfg`) and then runs.

Example 4:

```
./torpedo -runtime 60
```

`torpedo` runs for 60 minutes instead of the default 30 minutes (`-runtime 60`).

Example 5:

```
./torpedo -forever
```

`torpedo` runs indefinitely (`-forever`) and uses the default 30 minutes to complete each loop.

2.1.2.3 Output from torpedo

The following sample shows output from a passing `torpedo` test:

```
NOTE              This diagnostic should be run by itself for best test
NOTE              coverage.
CMDL              ./torpedo
TEST torpedo      FPU stress test                   Test(1/1), Loop(1/1)
REV               Version 2.2 compiled Mar 27 2001 at 12:13:14
INFO              Start time : Tue Mar 27 13:13:29 2001
INFO              Executing selected tests once.
INFO              Executing selected tests randomly.
RSLT torpedo      PASS         All Tests Passed
INFO              End time   : Tue Mar 27 13:43:29 2001
```

When the `torpedo` test detects an error and more than two processors are being tested, the test identifies the CPU that has different results than the other CPUs. The following sample shows output from a failing `torpedo` test:

```
NOTE              This diagnostic should be run by itself for best test
NOTE              coverage.
CMDL              ./torpedo
TEST torpedo      FPU stress test                   Test(1/1), Loop(1/1)
REV               Version 2.1 compiled on Mar 27 2001 at 12:13:14
INFO              Start time : Tue Mar 27 14:06:32 2001
INFO              Executing selected tests once.
**** ERROR 000001 Miscompare found between CPU 1 and CPU 0
**** ERROR 000001 Thread 1 on CPU 1 : results[0] = 0xbc6c779a3f73c3f9
**** ERROR 000001 Thread 0 on CPU 0 : results[0] = 0000000000000000
**** ERROR 000001 Miscompare found between CPU 2 and CPU 0
**** ERROR 000001 Thread 2 on CPU 2 : results[0] = 0xbc6c779a3f73c3f9
**** ERROR 000001 Thread 0 on CPU 0 : results[0] = 0000000000000000
**** ERROR 000001 MPFacilities - Not all CPUs obtained the same results.
DIAG       000001 CPU 0's results did not match those of any other CPUs.
DIAG       000001 The following 2 CPUs obtained identical results
DIAG       000001 1       2
DIAG       000001 Processor 0 is probably bad.
DIAG       000001 Location: /hw/module/001c07/node/cpubus/0/a
INFO              Executing selected tests randomly.
RSLT torpedo      FAIL         Error Occurred in 1 test of 3960
INFO              End time   : Tue Mar 27 14:36:32 2001
INFO              Maximum error count (1) reachedp
```

2.2 Memory Test

The memory test (`olmem`) is a directed test that verifies the memory and cache components in a C brick. 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 from all of the available CPUs 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.

2.2.1 Prerequisites for Running olmem

The `olmem` test has the following prerequisites:

- Your system must have an IP27 or IP35 processor.
- You must have root privilege.
- You must stop all user programs that have CPU affinity to receive accurate results.

2.2.2 Running olmem

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 2-3 for descriptions of the command-line options.

Table 2-3 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).

2.2.3 Output from olmem

The following sample shows output from a passing olmem test:

```
Mon Jul 10 11:44:02 CDT 2000
IRIX64 klsys7 6.5 6.5.9m 07070640 IP35
REV               Online Memory Diagnostics 1.0
INFO
INFO              NOTE: Most memory errors are seen by the kernel instead
INFO              of olmem. Check the console and /var/adm/SYSLOG for
INFO              memory errors after olmem is done.
INFO
CMDL              ./olmem
TEST tagram       Exercise cache tagram bits         Test(1/3), Loop(1/1)
INFO              Trying to test 1180893184 bytes (1126 MB) with 12 cpu's
HRTB              Starting threads to lock memory into place.............
HRTB              DONE!
HRTB              Running memory test. DONE!
RSLT tagram       PASS
TEST addr-pattern  Address pattern test (with inversion) Test(2/3),Loop(1/1)
HRTB              Running memory test. DONE!
RSLT addr-pattern PASS
TEST memaddr      Moving inversion memory test       Test(3/3), Loop(1/1)
HRTB              Running memory test....... DONE!
RSLT memaddr      PASS
LOOP              Completed Loop 1   of 1, duration: 114.882 sec     PASS
META              ITERATION=1   PASSES         NON-PASSES
META              tagram        1              0
META              addr-pattern  1              0
META              memaddr       1              0
META              TOTAL         3              0
```

The following sample shows output from a failing olmem test:

```
Thu May  4 12:42:24 PDT 2000
IRIX64 ioif-o2k 6.5-blosure-irix6.5.6m.101999-SN0 6.5.6m 10200846 IP27
REV               Online Memory Diagnostics 0.3
CMDL              ./olmem MEM=128 HWDEBUG=1
TEST tagram       Exercise cache tagram bits         Test(1/1), Loop(1/1)
INFO              Trying to test 134217728 bytes (128 MB) with 8 cpu's
HRTB               Starting threads to lock memory into place........ DONE!
INFO              Running memory test
**** ERROR 000001 Error at 0164828080 (/hw/module/1/slot/n1/node DIMM Bank 3)
HDBG       000001 Virtual Address: 12038080  Physical Address: 0164828080
```

```
HDBG       000001 from cpu /hw/module/1/slot/n2/node/cpu/a
HDBG       000001 Expected a5a5, got a5a0
INFO              Memory errors detected. Rescanning bad pages.
**** ERROR  000001  Error at 0164828080 (/hw/module/1/slot/n1/node DIMM Bank 3)
HDBG       000001 Virtual Address: 12038080  Physical Address: 0164828080
HDBG       000001 from cpu /hw/module/1/slot/n2/node/cpu/a
HDBG       000001 Expected a5a5, got a5a0
INFO              Examining collected errors
DIAG        000000 Suspected failure for 0164828080 (/hw/module/1/slot/n1/node
DIAG       000000 DIMM Bank 3): Cache (/hw/module/1/slot/n2/node/cpu/a)
HDBG       000001 (Error only happened on one CPU)
INFO
RSLT tagram       FAIL
INFO              Maximum error count (1) reached
META              ITERATION=1       PASSES         NON-PASSES
META              tagram            0              1
META              TOTAL             0              1
```

2.2.4 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`

2.3 Router Tests

The router tests are directed tests that verify router components and connections.

2.3.1 Router Test

The router test (`olrtr`) is a directed test that verifies the router components in R bricks. You also can configure `olrtr` to test specific components of the inter-node communications hardware. The logic that is tested includes network data paths, network addressing, and network-level cache coherency. It runs in three modes:

- *Quick mode (-q)* tests all links by sending data between all nodes. It uses data comparison to verify coherency.
- *Random mode (-r)* transfers and compares random data that is distributed randomly between the selected nodes.
- *Path mode (-P)* performs read and write operations across user-specified paths. It relies on hardware error detection.

2.3.1.1 Prerequisites for Running olrtr

The `olrtr` test has the following prerequisites:

- Your system must have an IP35 processor.
- You must have root privilege.
- You must stop all user jobs to receive accurate results.
- You must run the `topology` utility.

2.3.1.2 Running olrtr

Perform the following procedure to run `olrtr`:

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:
   `./olrtr` [*options*]

Refer to Table 2-4 for descriptions of the command-line options.

Table 2-4 olrtr Command-line Options

Option	Description
`-P` <*file*>	Tests a set of network paths in the specified file. Refer to Section 2.3.1.3, “Path List File,” for information on this file.
`-q` <*file*>	Tests the nodes in the specified file. Refer to Section 2.3.1.4, “Node List File,” for information on this file.

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

Option	Description
`-Q`	Tests all of the nodes in the system. **Note:** Ensure that other users are not using the system when you use this option.
`-r <`*file*`>`	Randomly tests the nodes in the specified file. Refer to Section 2.3.1.4 for information on the node list file.
`-R`	Randomly tests the nodes in the system. This is the default. **Note:** Ensure that other users are not using the system when you use this option.
`-G <`*component name*`>`	Lists the paths that are needed to test all of the ports of a router or MetaRouter. The string format of the router or MetaRouter is `/hw/module/006r16/router`. Refer to Section 2.3.1.3 for more information on the format of the list.

Example 1:

```
./olrtr -P pathList REPEAT=2 -continue
```

`olrtr` tests the paths between the nodes that are specified in the file `pathList` (`-P pathList`). `olrtr` runs for 2 loops (`REPEAT=2`) and continues when errors are detected (`-continue`).

Example 2:

```
./olrtr -q nodeList REPEAT=10 MAX_ERRORS=6
```

`olrtr` runs a quick test on the nodes that are specified in the file `nodeList` (`-q nodeList`). `olrtr` runs for 10 loops (`REPEAT=10`) and continues on error until 6 errors are reported (`MAX_ERRORS=6`).

Example 3:

```
./olrtr -G /hw/module/006r16/router > pathList
```

`olrtr` generates all combinations of paths that are required to test all ports on the router at /hw/module/006r16/router (`-G /hw/module/006r16/router`). This command line redirects the output to the file `pathList` (`> pathList`).

Note: You can use the file `pathList` as the input file for the `-P` option. You must edit the file `pathList` to remove the nonpath related statements and the paths that you do not want to test.

Example 4:

```
./olrtr -R -runtime 60 MAXERR=60
```

`olrtr` runs a random confidence test that stresses all links in the system (`-R`). `olrtr` runs for 60 minutes (`-runtime 60`). The longer run time is necessary because each iteration of the random confidence test checks only a random subset of the system. `olrtr` continues on error for up to 60 errors (`MAXERR=60`).

Note: The `-R` option is not necessary in this example because the random confidence test runs by default.

2.3.1.3 Path List File

The path list file is used to specify the paths between the nodes that are used in testing. An entry in this file consists of two node descriptions and represents a single path to be tested. Multiple entries can be used to test combinations of paths simultaneously; a node can be repeated in different entries of a file. The order of the nodes in an entry is not important in choosing a path. The following sample shows a path list file entry:

```
/hw/module/006c07/node /hw/module/006c10/node
```

2.3.1.4 Node List File

The node list file is used to specify the nodes that are used in testing. A single node is entered on each line of the file. The following sample shows a node list file with three node entries:

```
/hw/module/006c07/node
/hw/module/006c18/node
/hw/module/006c10/node
```

2.3.1.5 Output from olrtr

The following sample shows output from a passing `olrtr` test:

```
INFO: IMPORTANT:  Anytime HARDWARE changes are made to the system,
INFO:             YOU MUST REMOVE /tmp/olrtr.top to force olrtr to read
INFO:             the new topology or olrtr will not execute correctly.
REV               Online Router Diagnostic 0.0
CMDL              ./olrtr
TEST olrtr        Router Test                        Test(1/1), Loop(1/1)
RSLT olrtr        PASS          RandomTest : Mon Jul 10 11:52:38 2000
LOOP              Completed Loop 1   of 1, duration: 25.924 sec      PASS
META              ITERATION=1   PASSES         NON-PASSES
META              olrtr         1              0
META              TOTAL         1              0
```

If `olrtr` fails, it displays the following message (highlighted red):

```
RSLT olrtr        FAIL
```

If the hardware detects an unrecoverable fault, `olrtr` aborts and reports the error to the system console. If `olrtr` detects an error, it reports the physical source and destination nodes and continues running or exits (based on the setting of the `-continue` command-line option).

2.3.1.6 Troubleshooting Tips for olrtr

If you encounter problems while running `olrtr`, try the following solutions:

- Use the `linkstat(1)` command to view link faults.
- If it appears that `olrtr` has stalled, enter `ps -af | grep olrtr` to determine whether all of the processes are getting CPU time. If they are not getting CPU time, you may want to kill the job and wait until the targeted nodes are available.
- If hardware changes have been made since you last ran `olrtr`, remove `olrtr.top` from the `tmp` directory.
- If the `topology` utility was not run prior to starting `olrtr` and the following message is displayed, start the test again:

```
olrtr needed to create /tmp/olrtr.top
olrtr can now be restarted normally
```

If `olrtr` fails, the error might be caused by one of the following conditions:

- When using this test with `linkstat -ac`, any activity on the system can cause inaccurate test results.
- The `-P` option invokes a path test that does not perform data compares; it relies on hardware error detection and reporting. Because the `-r`, `-q`, `-R`, and `-Q` options perform data compares, you might see old or corrupted data that the error-detection hardware missed.

2.3.2 Router Link Test

The router link test (`linktest`) is a directed test that verifies the router links. You also can configure `linktest` to test specific components of the inter-node communications hardware. The logic that is tested includes network data paths, network addressing, and network-level cache coherency.

2.3.2.1 Prerequisites for Running linktest

The `linktest` diagnostic has the following prerequisites:

- Your system must have an IP27, IP29, or IP31 processor.
- You must have root privilege.
- You must run the `topology` utility.

2.3.2.2 Running linktest

Perform the following procedure to run `linktest`:

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:
   `./linktest` [*options*]

Refer to Table 2-5 for descriptions of the command-line options.

Note: The `linktest` diagnostic does not use the command-line options common to the online diagnostics available with SGI 3000 family systems.

Table 2-5 linktest Command-line Options

Option	Description
`-P` <*file*>	Tests the set of network paths in the specified file. The default runs the test on all network paths. For information on this file, refer to Section 2.3.2.3, “Path List File”.
`-q` <*file*>	Runs *quick* tests on the nodes in the specified file. The default is `-R`. For information on this file, refer to Section 2.3.2.4, “Node List File”.
`-Q`	Tests all of the nodes in the system. The default is `-R`. **Note:** Ensure that other users are not using the system when you use this option.
`-r` <*file*>	Runs a random confidence test on the nodes in the specified file. The default is `-R`. Refer to Section 2.3.1.4 for information on this file.
`-R`	Runs a random confidence test on all of the nodes in the system. This is the default. **Note:** Ensure that other users are not using the system when you use this option.
`-G` <*component name*>	Generates a list of the paths needed to test all of the ports of a router or MetaRouter. For information on the format of this list, refer to Section 2.3.2.3, “Path List File”.

Table 2-5 (continued) linktest Command-line Options

Option	Description
`-p` <*last pass*>	Runs the test for the specified number of passes. The default runs the test until it is manually stopped.
`-u` <*hh*:*mm*:*ss*>	Specifies how often the pass count is written to `stdout`. Use this to determine the progress of the test.

Example 1:

```
./linktest -P pathList -p 2
```

`linktest` tests the paths between the nodes that are specified in the file `pathList` (`-P pathlist`). `linktest` runs for 2 passes (`-p 2`).

Example 2:

```
./linktest -q nodeList -p 1
```

`linktest` runs a quick link test on the nodes that are specified in the file `nodeList` (`-q nodeList`). `linktest` runs for 1 pass (`-p 1`).

Example 3:

```
./linktest -G /hw/module/98/slot/r2/metarouter > pathList
```

`linktest` generates all combinations of paths that are needed to test all ports on the metarouter at /hw/module/98/slot/r2/metarouter (`-G /hw/module/98/slot/r2/metarouter`). This command line redirects the output to the file `pathList` (`> pathList`).

Note: You can use the file `pathList` as the input file for the `-P` option. You must edit the file `pathList` to remove the nonpath related statements and the paths that you do not want to test.

Example 4:

```
./linktest -R -p 100 -u 10:00
```

`linktest` runs a random confidence test that stresses all links in the system (`-R`). The test runs for 100 passes (`-p 100`). The large number of passes is necessary because each iteration of the random confidence test checks only a random subset of the system. `linktest` displays the current pass count every 10 minutes to update the user about test progress (`-u 10:00`).

2.3.2.3 Path List File

The path list file is used to specify the paths between the nodes that are used in testing. An entry in this file consists of two node descriptions and represents a single path to be tested. Multiple entries can be used to test combinations of paths simultaneously; a node can be repeated in different entries of a file. The order of the nodes in an entry is not important in choosing a path. The following sample shows a path list file entry:

```
/hw/module/8/slot/n4/node /hw/module/3/slot/n4/node
```

2.3.2.4 Node List File

The node list file is used to specify the nodes that are used in testing. A single node is entered on each line of the file. The following sample shows a node list file with three node entries:

```
/hw/module/8/slot/n4/node
/hw/module/3/slot/n4/node
/hw/module/3/slot/n3/node
```

Note: To get the descriptions of the nodes, routers, or MetaRouters, use the descriptions in the `/tmp/linktest.top` file.

2.3.2.5 Output from linktest

The following sample shows output from the `linktest` diagnostic:

Note: The `linktest` diagnostic does not use the output tags common to the online diagnostics available with SGI 3000 family systems.

```
strlab04 12# /usr/diags/bin/linktest -Q -p1

linktest start-up messages:    Thu Apr  5 14:22:00 2001

linktest terminating at pass 1  Thu Apr  5 14:22:03 2001

Lagging linktest child pid 152090 exits.  Thu Apr  5 14:22:03 2001
Lagging linktest child pid 151201 exits.  Thu Apr  5 14:22:03 2001
Lagging linktest child pid 152087 exits.  Thu Apr  5 14:22:03 2001
Lagging linktest child pid 151217 exits.  Thu Apr  5 14:22:03 2001
```

If the hardware detects an unrecoverable fault, `linktest` aborts and reports the error to the system console. If `linktest` detects an error, it reports the physical source and destination nodes and exits.

2.3.2.6 Troubleshooting Tips for linktest

- If you encounter problems while running `linktest`, try the following solutions:
  - Use the `linkstat(1)` utility to view link faults.
  - If it appears that `linktest` has stalled, enter `ps -af | grep linktest` to determine whether all of the processes are getting CPU time. If they are not getting CPU time, you may want to kill the job and wait until the targeted nodes are available.
  - If the `topology` utility was not run prior to starting `linktest` and the following message is displays, start the test again:

    ```
    linktest needed to create /tmp/linktest.top.
    linktest can now be restarted normally
    ```

- If `linktest` fails, the error might be caused by one of the following conditions:
  - When you use this test with `linkstat -ac`, any activity on the system can cause inaccurate test results.
  - The `-P` option invokes a path test that does not perform data compares; it relies on hardware error detection and reporting. Because the `-r`, `-q`, `-R`, and `-Q` options perform data compares, you might see old or corrupted data that the error-detection hardware missed.
  - Systems with disabled CPUs are not supported by this test.

2.4 I/O Tests

The I/O tests are directed tests that verify the I/O components in the I and P bricks.

2.4.1 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`).

2.4.1.1 Prerequisites for Running bridgeloc

The `bridgeloc` utility has the following prerequisites:

- Your system must have an IP35 processor.
- You must have root privilege.

2.4.1.2 Running bridgeloc

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`

2.4.1.3 Output from bridgeloc

The following sample shows output from the `bridgeloc` utility:

```
Mon Jul 10 11:47:04 CDT 2000
IRIX64 klsys7 6.5 6.5.9m 07070640 IP35
REV               Online PCI Bridge Locator and Diag Listing Utility 1.0
INFO              PCI bridges were found at the following locations:
INFO              /hw/module/007c10/Ibrick/xtalk/14/pci
INFO              /hw/module/007c10/Ibrick/xtalk/15/pci
INFO              /hw/module/007c13/Pbrick/xtalk/13/pci
INFO              /hw/module/007c13/Pbrick/xtalk/14/pci
INFO              /hw/module/007c13/Pbrick/xtalk/8/pci
INFO              /hw/module/007c16/Pbrick/xtalk/12/pci
INFO              /hw/module/007c16/Pbrick/xtalk/15/pci
INFO              /hw/module/007c16/Pbrick/xtalk/9/pci
INFO              These hwgraph vertices support the following online diags:
INFO              (Format is a command line stub for applicable online
INFO             diag.  Additional options may be required.  Refer to the
INFO              man page for the specific diag for details)
INFO              olpci -v /hw/module/007c10/Ibrick/xtalk/14/pci
INFO              olpci -v /hw/module/007c10/Ibrick/xtalk/15/pci
INFO              olpci -v /hw/module/007c13/Pbrick/xtalk/13/pci
INFO              olpci -v /hw/module/007c13/Pbrick/xtalk/14/pci
INFO              olpci -v /hw/module/007c13/Pbrick/xtalk/8/pci
INFO              olpci -v /hw/module/007c16/Pbrick/xtalk/12/pci
INFO              olpci -v /hw/module/007c16/Pbrick/xtalk/15/pci
INFO              olpci -v /hw/module/007c16/Pbrick/xtalk/9/pci
ENET              olenet -v /hw/module/007c10/Ibrick/xtalk/15/pci/4
 SIO              olsio -v /hw/module/007c10/Ibrick/xtalk/15/pci/4
 USB              olusb -v /hw/module/007c10/Ibrick/xtalk/15/pci/5
```

2.4.2 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 IO7 baseIO devices and other PCI devices.)

2.4.2.1 Prerequisites for Running olpci

The `olpci` utility has the following prerequisites:

- Your system must have an IP35 processor.
- You must have root privilege.
- You must use the `bridgeloc` utility output to determine the PCI bridge vertex:

```
Mon Jul 10 11:47:04 CDT 2000
IRIX64 klsys7 6.5 6.5.9m 07070640 IP35
REV         Online PCI Bridge Locator and Diag Listing Utility 1.0
INFO        PCI bridges were found at the following locations:
INFO        /hw/module/007c10/Ibrick/xtalk/14/pci
INFO        /hw/module/007c10/Ibrick/xtalk/15/pci
...
INFO        olpci -v /hw/module/007c10/Ibrick/xtalk/14/pci
INFO        olpci -v /hw/module/007c10/Ibrick/xtalk/15/pci
INFO        olpci -v /hw/module/007c13/Pbrick/xtalk/13/pci
INFO        olpci -v /hw/module/007c13/Pbrick/xtalk/14/pci
INFO        olpci -v /hw/module/007c13/Pbrick/xtalk/8/pci
INFO        olpci -v /hw/module/007c16/Pbrick/xtalk/12/pci
INFO        olpci -v /hw/module/007c16/Pbrick/xtalk/15/pci
INFO        olpci -v /hw/module/007c16/Pbrick/xtalk/9/pci
...
```

2.4.2.2 Running olpci

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 2-6 for descriptions of the command-line options.

Table 2-6 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.

Table 2-6 (continued) olpci Command-line Options

Option	Description
`-v|-vertex` <*pci-bridge-vertex*>	Specifies the hardware graph vertex. This option is required.

Example 1:

```
./olpci -v /hw/module/001c10/Ibrick/xtalk/14/pci
```

`olpci` lists the online diagnostics that are applicable to the /hw/module/001c10/Ibrick/xtalk/14/pci vertex and lists all vertices that are attached to the vertex (`-v /hw/module/001c10/Ibrick/xtalk/14/pci`).

Example 2:

```
./olpci -v /hw/module/001c10/Ibrick/xtalk/14/pci -pciCfgSpace -runtime 5
```

`olpci` lists the online diagnostics that are applicable to the /hw/module/001c10/Ibrick/xtalk/14/pci vertex and lists all vertices that are attached to the vertex (`-v /hw/module/001c10/Ibrick/xtalk/14/pci`). `olpci` also dumps and decodes all PCI configuration space registers (`-pciCfgSpace`). `olpci` runs for 5 minutes (`-runtime 5`).

2.4.2.3 Output from olpci

The following sample shows output from the `olpci` utility:

```
Mon Jul 10 11:50:31 CDT 2000
IRIX64 klsys7 6.5 07070640 IP35
REV              Online PCI Bridge Config Space 1.0 (Compiled Jul  6 2000
REV               20:48:25)
CMDL              ./olpci -v /hw/module/007c10/Ibrick/xtalk/14/pci
TEST pciDiagList  Online Diag List for attached de   Test(1/1), Loop(1/1)
INFO              Online PCI Configuration Check (OLPCI)
INFO              Online Diagnostic Listing for PCI devices attached to:
INFO              /hw/module/007c10/Ibrick/xtalk/14/pci/controller
INFO              (Format is a command line stub for applicable online
INFO              diag. Additional options may be required. Refer to the
INFO              the man page for the specific diag for details.)
INFO
INFO
RSLT pciDiagList  PASS         Diagnostic Listing completed successfully.
LOOP              Completed Loop 1   of 1, duration: 0.006 sec       PASS
META              ITERATION=1   PASSES         NON-PASSES
META              pciDiagList   1              0
META              TOTAL         1              0
```

2.4.3 Ethernet Test

The Ethernet test (olenet) is a directed test that checks the I-brick-based Ethernet. In all cases, olenet performs a basic test of the Ethernet controller and displays information about the current status of the I-brick-based Ethernet, including the MAC address, link speed, and link status.

The olenet test can also 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).

2.4.3.1 Prerequisites for Running olenet

The olenet test has the following prerequisites:

- Your system must have an IP35 processor.
- You must have root privilege.
- You must stop all user programs that use the I-brick-based Ethernet port to receive accurate results.
- You must use the bridgeloc utility output to determine the Ethernet controller vertex:

```
Mon Jul 10 11:47:04 CDT 2000
IRIX64 klsys7 6.5 6.5.9m 07070640 IP35
REV         Online PCI Bridge Locator and Diag Listing Utility 1.0
INFO        PCI bridges were found at the following locations:
INFO        /hw/module/007c10/Ibrick/xtalk/14/pci
INFO        /hw/module/007c10/Ibrick/xtalk/15/pci
...
ENETolenet -v /hw/module/007c10/Ibrick/xtalk/15/pci/4
...
```

- To run the external loopback tests, you must connect an external loopback connector (RJ-45 connector that has pin 1 connected to pin 3 and pin 2 connected to pin 6).

2.4.3.2 Running olenet

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 2-7 for descriptions of the command-line options.

Table 2-7 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` `[PACKETS=`*packets*`]`	Performs an external loopback test at 10 Mbps.
`-loop-ext-100` `[PACKETS=`*packets*`]`	Performs an external loopback test at 100 Mbps.
`-loop-int-ioc3` `[PACKETS=`*packets*`]`	Performs an internal loopback test on the IOC3 chip.
`-loop-int-phy` `[PACKETS=`*packets*`]`	Performs an internal loopback test on the PHY chip.
`-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.
`PACKETS=<`*packets*`>`	Sends the specified number of packets in a loopback test. The default is `1000`.

Example 1:

```
./olenet -vertex /hw/module/001c10/Ibrick/xtalk/15/pci/4
```

`olenet` runs a basic test and reports information on the current state of the I-brick-based Ethernet for the I brick that is connected to the /hw/module/001c10/Ibrick/xtalk/15/pci/4 vertex (`-vertex /hw/module/001c10/Ibrick/xtalk/15/pci/4`).

Example 2:

```
./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`).

2.4.3.3 Output from olenet

The following sample shows output from a passing `olenet` test:

```
Mon Jul 10 11:48:27 CDT 2000
IRIX64 klsys7 6.5 07070640 IP35
REV              Online Ethernet Diagnostics 1.1
CMDL             ./olenet -v /hw/module/007c10/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: 08:00:69:11:34:61
INFO             Speed: 100 Mb/s     Full Duplex: Disabled
INFO             Link Status: Good Autonegotiation: Enabled
INFO
INFO             Device ready to transmit
INFO
RSLT enet-info   PASS
LOOP             Completed Loop 1   of 1, duration: 0.005 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/000C1/Cbrick/xtalk/15/pci/
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
```

2.4.4 SuperIO Port Test

The SuperIO (SIO) port (`olsio`) test runs internal loopback tests on port A and B (if port B exists) of an SIO port using the PIO and DMA transfers. By default, `olsio` runs internal loopback tests on all serial ports that are located at the specified IOC3 ASIC vertex. `olsio` may be configured via command-line options to run specific tests on a specific port.

2.4.4.1 Prerequisites for Running olsio

The `olsio` test has the following prerequisites:

- Your system must have an IP35 processor.
- 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 Jul 10 11:47:04 CDT 2000
IRIX64 klsys7 6.5 6.5.9m 07070640 IP35
REV         Online PCI Bridge Locator and Diag Listing Utility 1.0
INFO        PCI bridges were found at the following locations:
INFO        /hw/module/007c10/Ibrick/xtalk/14/pci
INFO        /hw/module/007c10/Ibrick/xtalk/15/pci
...
SIO         olsio -v /hw/module/007c10/Ibrick/xtalk/15/pci/4
...
```

2.4.4.2 Running olsio

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 2-8 for descriptions of the command-line options.

Table 2-8 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 2-8 (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.

Example 1:

```
./olsio -v /hw/module/001c10/Ibrick/xtalk/15/pci/4
```

`olsio` tests all configured SIO ports that are located at /hw/module/001c10/Ibrick/xtalk/15/pci/4 (`-v /hw/module/001c10/Ibrick/xtalk/15/pci/4`) by using internal loopback. `olsio` performs PIO and DMA transfers for the default 10 passes.

Example 2:

```
./olsio -v /hw/module/001c10/Ibrick/xtalk/15/pci/4 -p 1000 -DMA -rs232
-SPExt -A
```

`olsio` tests port A (`-A`) of the SIO port that is located at /hw/module/001c10/Ibrick/xtalk/15/pci/4 (`-v /hw/module/001c10/Ibrick/xtalk/15/pci/4`) by using single-port external loopback (`-SPExt`) with DMA transfers (`-DMA`) in RS-232 mode (`-rs232`). `olsio` performs 1,000 passes (`-p 1000`).

2.4.4.3 Output from olsio

If `olsio` passes, it displays the following message (highlighted green):

```
RSLT olsio      PASS
```

If `olsio` fails, it displays the following message (highlighted red):

```
RSLT olsio      FAIL
```

2.4.4.4 Troubleshooting Tips for olsio

When an external loopback test fails in RS-422 mode (the `-rs422` option) while performing DMA transfers, it does not indicate faulty hardware.

2.4.5 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.

2.4.5.1 Prerequisites for Running olusb

The `olusb` test has the following prerequisites:

- Your system must have an IP35 processor.
- You must have root privilege.
- You must use the `bridgeloc` utility output to determine the vertex of the USB controller:

```
Mon Jul 10 11:47:04 CDT 2000
IRIX64 klsys7 6.5 6.5.9m 07070640 IP35
REV         Online PCI Bridge Locator and Diag Listing Utility 1.0
INFO        PCI bridges were found at the following locations:
INFO        /hw/module/007c10/Ibrick/xtalk/14/pci
INFO        /hw/module/007c10/Ibrick/xtalk/15/pci
...
USB         olusb -v /hw/module/007c10/Ibrick/xtalk/15/pci/5
```

2.4.5.2 Running olusb

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 2-9 for descriptions of the command-line options.

Table 2-9 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 is available once the test is finished.
`-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 at 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*.

Example 1:

```
./olusb -vertex /hw/module/001c10/Ibrick/xtalk/15/pci/5
```

`olusb` tests the USB controller chip in the I brick that is located at /hw/module/001c10/Ibrick/xtalk/15/pci/5 (`-vertex /hw/module/001c10/Ibrick/xtalk/15/pci/5`). `olusb` does not attempt to communicate with devices that are attached to the USB bus.

Example 2:

```
./olusb -vertex /hw/module/001c10/Ibrick/xtalk/15/pci/5 -usbinv-probe
```

`olusb` tests the USB controller chip in the I brick that is located at /hw/module/001c10/Ibrick/xtalk/15/pci/5 (`-vertex /hw/module/001c10/Ibrick/xtalk/15/pci/5`). `olusb` attempts to communicate with and display information about attached USB devices (`-usbinv-probe`).

2.4.5.3 Output from olusb

If `olusb` passes, it displays the following message (highlighted green):

```
RSLT olusb      PASS
```

If `olusb` fails, it displays the following message (highlighted red):

```
RSLT olusb      FAIL
```

2.4.5.4 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.

2.5 Storage Tests

2.5.1 Disk Test

The disk (`oldisk`) test checks a system disk by writing and reading data patterns to the disk and comparing the data. By default, `oldisk` accesses the drive asynchronously.

Note: The `oldisk` test does not work over the Network File System (NFS).

2.5.1.1 Prerequisites for Running oldisk

The `oldisk` test has the following prerequisites:

- Your system must have an IP27, IP29, IP31, or IP35 processor.
- You must have root privilege.

2.5.1.2 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 2-10 for descriptions of the command-line options.

Table 2-10 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.

Table 2-10 (continued) oldisk Command-line Options

Option	Description
`-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 file system 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 file system 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`).

2.5.1.3 Output from oldisk

If `oldisk` passes, it displays the following message (highlighted green):

```
RSLT oldisk       PASS
```

If `oldisk` fails, it displays the following message (highlighted red):

```
RSLT oldisk       FAIL
```

2.5.1.4 Troubleshooting Tips for oldisk

If you suspect that a memory error could show up as a disk error, use the `-paranoid` option.

2.5.2 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.

2.5.2.1 Prerequisites for Running oltape

The oltape test has the following prerequisites:

- Your system must have an IP27, IP29, IP31, or IP35 processor.
- You must have root privilege.
- To run the eot or tapemark tests, you must configure down the tape device that you want to test.

2.5.2.2 Running oltape

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 2-11 for descriptions of the command-line options.

Table 2-11 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 2-11 (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 only available if 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 only available if 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 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 2-11 (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.)

2.5.2.3 Output from oltape

If oltape passes, it displays the following message (highlighted green):

```
RSLT oltape         PASS
```

If the oltape test fails, it displays the following message (highlighted red):

```
RSLT oltape         FAIL
```

2.6 Network Tests

The network tests are directed tests that exercise various types of network connections.

2.6.1 Gigabyte System Network Test

The Gigabyte System Network (GSN) test (`olgsn`) verifies the functionality of the GSN hardware. The `olgsn` test has the following states based on the connection state of the GSN board.

- *Unconnected* opens the GSN device, reads the device statistics, verifies the flags, and closes the device. This test does not actually transfer data to the local SuMAC ASIC.
- *External Loopback* accesses the local SuMAC ASIC on the GSN board and pings it through the external loopback connector. This test also sends the HIPPI-6400 ADMIN micropacket through the external loopback connector to the driver for processing.
- *Point to Point* accesses the local SuMAC ASIC on the GSN board, pings a remote SuMAC ASIC on a GSN board that is located on the other side of the physical wire, and sends a HIPPI-6400 ADMIN micropacket to the GSN driver on the opposite point.
- *HIPPI-6400 Switch* accesses the local SuMAC ASIC on the GSN board, pings a remote SuMAC ASIC on the switch on the other side of the physical wire, and sends an ADMIN micropacket to the remote SuMAC ASIC.

2.6.1.1 Prerequisites for Running olgsn

The `olgsn` test has the following prerequisites:

- Your system must have an IP35 processor.
- You must have root privilege.
- To run the external loopback, point-to-point, or HIPPI-6400 switch tests, you must have a loopback connector installed or a cable connected to another GSN device.

2.6.1.2 Running olgsn

Perform the following procedure to run the `olgsn` 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:
   `./olgsn` [*options*]

   Refer to Table 2-12 for descriptions of the test options.

Table 2-12 olgsn Command-line Options

Option	Description
`-gsn` <*gsn device number*>	Performs this test on the specified GSN device. The default is 0.
`-iloops` <*number of internal test loops*>	Executes the specified number of test loops. The default is 100.

2.6.1.3 Output from olgsn

If `olgsn` passes, it displays the following message (highlighted green):

```
RSLT olgsn          PASS
```

If the `olgsn` test fails, it displays the following message (highlighted red):

```
RSLT olgsn          FAIL
```

2.6.2 TCP Socket 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.

2.6.2.1 Prerequisites for Running olvst

The `olvst` test has the following prerequisites:

- Your system must have an IP27, IP29, IP31, or IP35 processor.
- You must have root privilege.

2.6.2.2 Running olvst

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 2-13 for descriptions of the command-line options.

Table 2-13 olvst Command-line Options

Option	Description
`-a|-action` <*action*>	Performs the specified action. The following 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 2-13 (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 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 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`

2.6.2.3 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
```

2.6.3 HIPPI Interface Test

The HIPPI interface test (`olvht`) is a directed test that exercises HIPPI channel pairs. The `olvht` test runs in two modes that use raw channel HIPPI data to exercise the configured HIPPI network devices in the system that you are testing:

- *Loopback mode* sends and receives data through a loopback cable on the HIPPI board. It then compares the actual data with the expected values.
- *Remote mode* runs on either a single system with a loopback cable installed or between two connected systems.

  To exercise the HIPPI interface on a single system, the test sends and receives data through the loopback cable on the HIPPI board. It then compares the actual data with the expected values.

  To exercise the HIPPI interface using two systems, the test writes data from the remote system to the local system. It then compares the actual data read with the expected values. It is better to run the test between two different machines; this will verify that the complete path between hosts is functional.

If `olvht` fails, the failing hardware is most likely the HIPPI network interface that it tested. The failing FRU is the HIPPI board that contains the failing network interface.

2.6.3.1 Prerequisites for Running olvht

The `olvht` test has the following prerequisites:

- Your system must have an IP27, IP29, IP31, or IP35 processor.
- You must have root privilege.
- You must stop all user programs that use the HIPPI interface to receive accurate results.
- You must be familiar with the terms *I-Field* and *ULP id*. If you are not familiar with these terms, use `olvst` instead.
- If the HIPPI device has been configured for use with TCP, use the `ifconfig(8)` command to configure the HIPPI interface *down*.

  Note: If you are not familiar with raw channel HIPPI, use `olvst(1)` to test the HIPPI device. The `olvst` test requires that the TCP interface be configured “up” with `ifconfig`.

2.6.3.2 Running olvht

Perform the following procedure to run the `olvht` 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:
   `./olvht <-I|-IField` *I-Field*`> <-u|-ULpid`** *ULP-id*`> <-d|-device`** *device-name*`>` [*options*]

Refer to Table 2-14 for descriptions of the command-line options.

Note: If you want to test HIPPI communication between two systems, enter the `olvht` command on both systems. (Configure one system to read data and one system to write data.)

Table 2-14 olvht Command-line Options

Option	Description
`-a|-action` <*action*>	Performs the specified action. The following are valid *action* values: `ping` elicits ECHO_RESPONSE packets from the host. `rw` reads and writes data on the same machine. This is the default. The machine must be cabled in loopback to itself. `read` listens for and reads data from the cooperating process. Ensure that you start a read action before you start a write action. `write` writes data to the specified host. Ensure that you start a `read` action before you start a `write` action. **Note:** If you use the `-blocksize` option with the `read` action, ensure that you specify the `-noESP` command; otherwise, a core file will be produced at the end of the test.
`-b|-blocksize` <*blocksize-max*`[:`*blocksize-min*`[:`*block-step*`]] | pass`>	Uses the specified block size for I/O operations. *blocksize-max* specifies the maximum block size. *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. **Note:** The block size must be the same on both the receiving and sending hosts. **Note:** If you use this option with the `read` action, ensure that you specify the `-noESP` command; otherwise, a core file will be produced at the end of the test.
`-d|-device` <*device-name*>	Performs the I/O operations on the specified HIPPI device. (Example: `/dev/hippi0`, `/dev/hippi1`, etc.) This option is required.
`-e|-echofile` <*echo-file-name*>	Writes the first data buffer to the specified file.
`-I|-IField` <*I-Field*>	Specifies the I-Field value. If you are looped back to the same interface, the *I-Field* value can be anything because the HIPPI switch does not exist in the path from source to destination. This option is required. **Caution:** Do not run this test if you are unfamiliar with the term *I-Field*; use `olvst` instead.
`-o|-options` <*options*>	Turns on the list of options, separated by colons. There is no default. The following are valid *option* values: `fast` does not compare data. `verbose` displays the progress of the test.

Table 2-14 (continued) olvht Command-line Options

Option	Description
`-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 are valid *pattern* values: `bits` sets each byte to 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 position. `slide1` sets bit 0 and clears all other bits of the first byte. Subsequent words are circularly left-shifted by 1 bit position. `random` sets each byte to a random value. `all` = all patterns are used, one per pass. This is the default. **Note:** The pattern must be the same on both the receiving and sending hosts.
`-u|-ULpid` <*ULP-id value*>	Specifies the ULP-id value. This value must be at least `128`. This option is required. **Caution:** Do not run this test if you are unfamiliar with the term *ULP id*; use `olvst` instead.
`-W|-WaitOnRead` <*wait-on-read value*>	Specifies the number of 0.10-second intervals that the HIPPI driver waits for a read action to occur. The maximum value is `254`.
`NUMREADS=`<*n*>	Specifies the number of reads to post.
`TIMEOUT=`<*n*>	Specifies the time-out value.

Example:

```
./olvht -a read -u 128 -I 0x0 -d /dev/hippi0 -p 10000
```

`olvht` runs a read process (`-a read`) on /dev/hippi0 (`-d /dev/hippi0`). `olvht` uses a directly connected interface with a loopback cable (`-I 0x0`). `olvht` runs 10,000 passes (`-p 10000`). `olvst` uses ULP ID 128 (`-u 128`).

Note: There must be an `olvht` write process on the same system or a remote system. A sample write process to run on the same system for this example is: `olvht -a write -u 128 -I 0x0 -d /dev/hippi1 -p 10000`

2.6.3.3 Output from olvht

If olvht passes, it displays the following message (highlighted green):

```
RSLT olvht        PASS
```

The following sample shows output from an unresolved olvht test:

```
Mon Jul 10 11:54:37 CDT 2000
IRIX64 klsys7 6.5 07070640 IP35
REV               Online HIPPI Channel Test 1.0 (Compiled Jul  7 2000
REV               00:34:57)
CMDL              ./olvht
TEST olvht        HIPPI Channel Test                Test(1/1), Loop(1/1)
**** ERROR 000000 The I-field must be supplied.
**** ERROR 000000 The ulpid must be supplied.
**** ERROR 000000 A device name must be specified.
RSLT olvht        UNRESOLVED   Usage error.  Run olvht -h for help.
LOOP              Completed Loop 1   of 1, duration: 0.000 sec       PASS
META              ITERATION=1   PASSES         NON-PASSES
META              olvht         0              1
META              TOTAL         0              1
```

If the olvht test fails, it displays the following message (highlighted red):

```
RSLT olvht        FAIL
```

2.7 Real-time Interrupt Test

The real-time interrupt (`olrti`) test checks the capability of the real-time interrupt to send and receive interrupts. It tests the system by sending interrupts from the RTO connector to the RTI connector. The `olrti` test has the following subtests:

- *Internal (*`-internal`*)* tests the state of the internal loopback.
- *External (*`-external`*)* tests the port by passing an interrupt from the RTO connector to the RTI connector via an external loopback cable.

By default, `olrti` runs both the internal and external loopback tests.

2.7.1 Prerequisites for Running olrti

The `olrti` test has the following prerequisites:

- Your system must have a IP27, IP29, IP31, or IP35 processor.
- You must have root privilege.
- To test external loopback (the `-external` command or default parameters), you must connect a loopback cable to the RTO and the RTI connectors.

  Note: The connection is a $^1/_8$-in. stereo minijack. Tip = 5V | Ring = Interrupt | Sleeve = Ground. For other pin configurations, consult the `ei(1)` man page.

2.7.2 Running olrti

Perform the following procedure to run the `olrti` 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:
   `./olrti` [*options*]

Refer to Table 2-15 for descriptions of the command-line options.

Table 2-15 olrti Command-line Options

Option	Description
`<-external> -v|-vertex` <*EI-Vertex*>	Specifies the location of the hardware vertex for external interrupts. Use this option with the `-external` option.
`-internal`	Tests only the internal loopback capability. The default tests both the internal and external loopback capabilities.
`-external <-v|-vertex` *EI-Vertex*>	Tests only the external loopback capability. This option requires the `-vertex` option. The default tests both the internal and external loopback capabilities.
`-on`	Asserts the RTO connector to High.

Table 2-15 (continued) olrti Command-line Options

Option	Description
`-off`	Asserts the RTO connector to Low. This is the default.
`-pulse <PERIOD=`*time*`| FREQ=`*frequency*`>`	Outputs a pulse train. This option requires the `PERIOD` or `FREQ` option.
`-square <PERIOD=`*time*`| FREQ=`*frequency*`>`	Outputs a square wave. This option requires the `PERIOD` or `FREQ` option.
`-strobe`	Outputs a single interrupt.
`-receive <TIMEOUT=`*time*`>`	Monitors for interrupts until a time-out occurs. This option requires the `TIMEOUT` option.
`<-pulse|-square> PERIOD=<`*time*`>`	Specifies the period of time (in microseconds) between pulses or the width of a square wave. Use this option with the `-pulse` and `-square` options.
`<-pulse|-square> FREQ=<`*frequency*`>`	Specifies the frequency (in Hertz) of a pulse train or square wave. Use this option with the `-pulse` and `-square` options.
`<-on|-off| -pulse|-square> LENGTH=<`*time*`>`	Specifies the length (in seconds) of output. Use this option with the `-on`, `-off`, `-pulse`, and `-square` options.
`<-receive> TIMEOUT=<`*time*`>`	Specifies the amount of time (in seconds) that may pass between interrupts in receive mode. Use this option with the `-receive` option.

Example 1:

```
./olrti -v /hw/module/001c10/Ibrick/xtalk/15/pci/4 -internal
```

`olrti` runs an internal loopback test (`-internal`) on the I brick located at /hw/module/001c10/Ibrick/xtalk/15/pci/4 (`-v /hw/module/001c10/Ibrick/xtalk/15/pci/4`).

Example 2:

```
./olrti -v /hw/module/001c10/Ibrick/xtalk/15/pci/4
```

`olrti` runs internal and external loopback tests on the I brick located at /hw/module/001c10/Ibrick/xtalk/15/pci/4 (`-v /hw/module/001c10/Ibrick/xtalk/15/pci/4`). The external loopback test requires a loopback cable between the RTO and the RTI connectors.

2.7.3 Output from olrti

If `olrti` passes, it displays the following message (highlighted green):

```
RSLT olrti      PASS
```

If `olrti` fails, it displays the following message (highlighted red):

```
RSLT olrti      FAIL
```

2.8 System Stress Tests

The system stress tests simulate user loads on the system by testing several hardware components simultaneously.

2.8.1 Pandora

The `pandora` test is a programmable system-level stress test that simulates heavy user loads on a system. This test stresses the entire system, including:

- Processors (integer and floating-point unit)
- Memory
- I/O devices (SCSI devices, etc.)
- Network devices (Ethernet, etc.)
- Router interconnect hardware
- Graphics (multipipe system) hardware

2.8.1.1 Prerequisites for Running pandora

The `pandora` test has the following prerequisites:

- Your system must have an IP27, IP29, IP31, or IP35 processor.
- You must have root privilege.
- You must stop all user programs or other diagnostics to receive accurate results.

2.8.1.2 Running pandora

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 2-16 for descriptions of the command-line options.

Table 2-16 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.

Table 2-16 (continued) pandora Command-line Options

Option	Description
`-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`.
`-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.

2.8.1.3 Output from pandora

The following sample shows output from a passing `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
```

2.8.1.4 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`

2.8.2 Grizzly

The `grizzly` test is a stress test that simulates normal and heavy system loads. This test mimics the process, memory access, and memory usage patterns of systems that run a combination of several customer applications. The `grizzly` test targets:

- CPU
- FPU
- Instruction, data, and secondary caches
- Memory
- Hubs
- Routers
- Disk interfaces

2.8.2.1 Prerequisites for Running grizzly

The `grizzly` test has the following prerequisites:

- Your system must have an IP27, IP31, or IP35.
- You must have root privilege.
- You must stop all user programs or other diagnostics to receive accurate results.
- You should run this test from superuser mode, especially if you want to use less than half of physical memory.

2.8.2.2 Running grizzly

Perform the following procedure to run the `grizzly` test:

1. Enter the following command to change to the directory that contains the diagnostics:
   `cd /usr/diags/bin`
2. Use one of the following methods to start `grizzly`:
   - Enter the following command to run `grizzly` from a script file with preset parameters:
     `./griz_script`
   - Enter the following command to run `grizzly` with the parameters that you specify:
     `./grizzly` [*options*]

Refer to Table 2-17 for descriptions of the command-line options.

Table 2-17 grizzly Command-line Options

Option	Description
`-iter` <*1 arg(s)*>	Specifies the number of iterations.
`-numProc` <*1 arg(s)*>	Starts the specified number of processes or threads.
`-regionSize` <*1 arg(s)*>	Uses the specified size of the memory region.

Table 2-17 (continued) grizzly Command-line Options

Option	Description
`-numRegion` <*1 arg(s)*>	Uses the specified number of memory regions.
`-numFiles` <*1 arg(s)*>	Uses the specified number of files.

Example 1:

```
./griz_script
```

`grizzly` tests 80% (or more) of the configured memory and all CPUs.

Example 2:

```
./grizzly -runtime 10
```

`grizzly` runs for 10 minutes.

2.8.2.3 Output from grizzly

The following sample shows output from a passing `grizzly` test:

```
REV              GRIZZLY System Diagnostics Version 5.1
REV              Fri Jul  7 00:34:51 PDT 2000 by sherwood on beel
CMDL             ./grizzly -runtime 2 -notrace -noinfo
TEST Grizzly     Grizzly System Test               Test(1/1), Loop(1/1)
WARN             cannot find /usr/tmp/voidfs ???, Continuing
BEGN ITER        Starting Grizzly Test Loop 1 gtime 0 parmset 0
END  ITER        Completed Loop 1    duration: 8.217 sec   PASS
BEGN ITER        Starting Grizzly Test Loop 2 gtime 111302 parmset 1
END  ITER        Completed Loop 2    duration: 15.313 sec   PASS
BEGN ITER        Starting Grizzly Test Loop 3 gtime 283098 parmset 3
END  ITER        Completed Loop 3    duration: 22.559 sec   PASS
BEGN ITER        Starting Grizzly Test Loop 4 gtime 487810 parmset 5
END  ITER        Completed Loop 4    duration: 23.163 sec   PASS
BEGN ITER        Starting Grizzly Test Loop 5 gtime 650072 parmset 5
END  ITER        Completed Loop 5    duration: 21.638 sec   PASS
BEGN ITER        Starting Grizzly Test Loop 6 gtime 781645 parmset 5
END  ITER        Completed Loop 6    duration: 14.883 sec   PASS
TOUT             ****************************************
TOUT             *======= Time's Up, My Dear Friend ========*
TOUT             *== Do Not have time to run another Loop ==*
TOUT             ****************************************
NOTE             There were no failures detected
RSLT Grizzly     PASS          Grizzly Ran Successfully (No Failures)
LOOP             Completed Loop 1   of 1, duration: 110.576 sec     PASS
META             ITERATION=1   PASSES         NON-PASSES
META             Grizzly       1              0
META             TOTAL         1              0
```

If `grizzly` fails, it displays the following message (highlighted red):

```
RSLT Grizzly     FAIL
```

Reader Comment Form

Title: Online Diagnostics **Number: 108-0286-005**

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 _____

[or attach your business card]

Attn: Service Publications and Training
890 Industrial Boulevard
P.O. Box 4000
Chippewa Falls, WI 54729-0078
USA