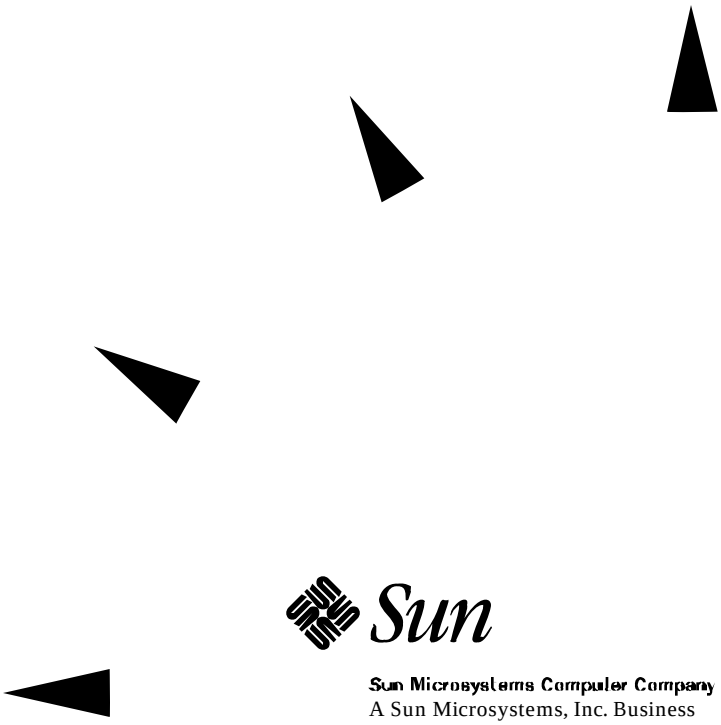
SunDiag User's Guide

Addendum for SMCC Hardware



2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part No: 801-7263-10
Revision A, November 1994

Sun Microsystems Computer Company
A Sun Microsystems, Inc. Business

Please
Recycle

Adobe PostScript

Contents

Figures

Tables

Preface

About This Book

The SunDiag™ 4.4 on-line system exerciser enables you to run multiple diagnostic hardware tests from a single interface. The SunDiag diagnostic tests described in this document can be run from the SunDiag OPEN LOOK® window interface, through serial ports, or individually through shell command lines.

As part of the Solaris® 2.4 Hardware: 11/94 document set, this manual accompanies the *SunDiag User's Guide* and contains information about individual SunDiag tests that are specific to Sun Microsystems Computer Corporation™ (SMCC) hardware. For more basic information about the SunDiag system exerciser, including a description of the user interfaces, refer to the *SunDiag User's Guide.*

How This Book Is Organized

Chapter 1, "Overview," introduces new tests and features added to this release of SunDiag 4.4 system exerciser. This chapters also addresses several Solaris operating environment issues.

Chapter 2, "CPU Test Descriptions," describes those tests that appear in the "CPU Devices" section of the SunDiag Control Panel.

Chapter 3, "SBus Test Descriptions," describes those tests that appear in the "SBus Devices" section of the SunDiag Control Panel.

Chapter 4, "User Test Descriptions," describes two tests — `sunbuttons` and `sundials` — that are not automatically detected by SunDiag, and must be run either from the command line or by creating a `.usertest` file.

Appendix A, "Loopback Connectors," provides information about the loopback connectors required by some of the SunDiag tests described in this book.

What Typographic Changes and Symbols Mean

`This typeface` represents text as it appears on your screen, which includes system messages, the names of commands and individual tests, and pathnames to files or directories. For example: "The SunDiag test `audbri` is located in the `/opt/SUNWdiag/bin` directory."

`This boldface type` represents text you type on a command line. For example: "Type **`/opt/SUNWdiag/bin/sundiag`** at the superuser prompt to start SunDiag."

Italics represent variables that are dependent on the system being tested. For example, **`D=`***device_name* is an option to many SunDiag tests where more than one device can be tested. You must specify the particular device to test by entering the **`D=`** immediately followed by your choice of device. If the Ethernet connection you wanted to test was designated by the device named `/dev/le0`, you would replace **`D=`***device_name* with **`D=/dev/le0`**.

The following table further describes the typefaces and symbols used in this book.

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `system% You have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`system% su` `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename*.
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide*. These are called *class* options. You *must* be root to do this.
Code samples are included in boxes and may display the following:		
%	UNIX C shell prompt	`system%`
$	UNIX Bourne and Korn shell prompt	UNIX Bourne and Korn shell prompt
#	Superuser prompt, all shells	Superuser prompt, all shells

Overview 1

1.1 About This Book

This manual accompanies the S*unDiag User's Guide* and contains information about individual SunDiag tests that are specific to SMCC hardware. For more basic information about the SunDiag™ system exerciser, refer to the S*unDiag User's Guide*.

1.2 What's New in Version 4.4

If you're familiar with earlier versions of SunDiag system exerciser, here's a quick summary of the tests that have been added with this release:

`tcxtest` (S24 Frame Buffer Test)	*page 2-80*
`pcmiatest` (PCMCIA Modem Card Test)	*page 3-57*
`irtest` (Infrared Interface Test)	*page 3-59*
`plntest` (SPARCstorage Array Controller Test)	*page 3-61*

These teste were added with the SunDiag 4.3 release:

`cg14test` (Color Graphics Frame Buffer Test)	*page 2-46*
`sxtest` (Pixel Processor Test)	*page 2-63*
`spdtest` (SPARCprinter Test)	*page 3-45*
`rtvctest` (SunVideo Test)	*page 3-48*

These tests were added with the SunDiag 4.2 release:

`leotest` (SBus Printer Test)	*page 3-30*
`pstest` (Prestoserve NFS Accelerator Test)	*page 3-18*
`spiftest` (Serial Parallel Interface Test)	*page 3-20*

1.3 Solaris Issues

Here is a list of Solaris operating environment issues that relate to the operation of the SunDiag system exerciser.

New Device Drivers

When adding a new device driver in the Solaris operating environment, you must reboot the machine with the following command before SunDiag kernel will recognize the new driver:

```
ok boot -r
```

When you use the **`boot -r`** command, the system will probe all attached hardware devices and assign nodes in the filesystem to represent only those devices actually found. It will also configure the logical namespace in `/dev` as well as the physical namespace in `/devices`. If you have removed a device from the system, then you also need to reboot the system with **`boot -r`** command before the SunDiag kernel sees the correct devices.

See the `kernel`(1M) man page for more information.

Dual Frame Buffers

The frame buffer (FB) locking feature needs to be enabled if, and only if, the frame buffer to be tested is running OpenWindows™ software.

Running OpenWindows software on more than one frame buffer is not supported by the SunDiag system exerciser.

The `-w` Option

The **`-w`** option to the `sundiag` command tells SunDiag to write the system hardware configuration to the `/var/adm/sundiaglog/sundiag.conf` file.

1.4 Displaying the POST Log File on Sun-4m Systems

On Sun-4m systems (for example, the SPARCserver™ 1000 and the SPARCcenter™ 2000 series systems), you can display the most recent power-on self test (POST) report from the SunDiag Log Files Menu.

When running the SunDiag software on a Sun-4m system, the Log Files Menu will add a POST Msgs option. By selecting this option and clicking on the Display button, you will display the most recent POST system report created from information on the system's OpenBoot™ PROM.

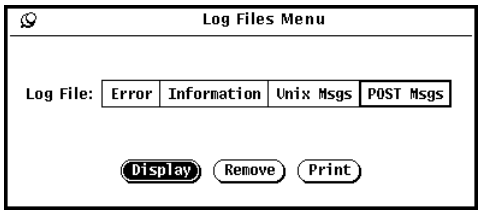


Figure 1-1 Log Files Menu With POST Msgs Option

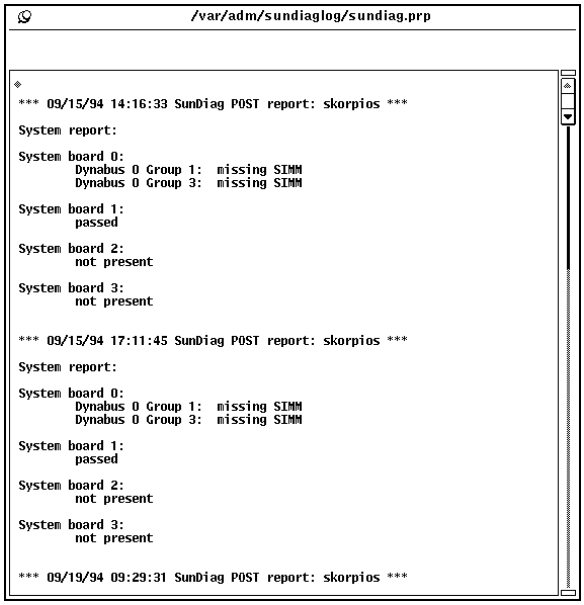


Figure 1-2 SunDiag POST Message System Report

The System report messages will be saved in the `/var/adm/sundiaglog/sundiag.prp` log file. From the Log Files Menu, you can print or remove (remove all but the most recent report) this log file.

Follow the instructions below to print this system report from the command line:

1. **As root, change directories to the SunDiag `bin` directory.**
   This directory is `/opt/SUNWdiag/bin` by default.

2. **Type the `prp` command to print the POST status report.**

```
# ./prp

System report:

System board 0:
    Dynabus 0 Group 1:  missing SIMM
    Dynabus 0 Group 3:  missing SIMM

System board 1:
    passed

System board 2:
    not present

System board 3:
    not present
```

1.5 Overrun/Underrun Diskette Test Errors

Diskette overrun/underrun error messages will appear in your test system's console window if you test the diskette drive while the system is very heavily loaded.

The SunDiag software is designed to stress test a system, so testing the diskette drive while running many other SunDiag tests may cause overrun/underrun errors. These overrun/underrun errors will cause the diskette tests to fail.

If an error occurs, you may see a messages similar to the ones below in the SunDiag Console window:

```
09/22/94 11:53:34 diskette rawtest.2 ERROR: Big read failed on disk, in-between blocks 1386
and 1512: I/O error.
09/22/94 11:53:40 lostagain SunDiag INFO: *Failed test* (diskette)rawtest.2 passes: 0
errors: 1
```

Also, one of the following messages may be printed in the console after an overrun/underrun error:

```
fd0: write failed (sr1 sr2 sr3)
fd0: overrun/underrun
```

or:

```
fd0: read failed (sr1 sr2 sr3)
fd0: overrun/underrun
```

sr1, *sr2*, and *sr3* represent the values of the floppy disk controller's status registers. Please refer to the `fd(7)` manpage for more details about these messages.

Note – These overrun/underrun errors are caused by a hardware limitation and cannot be fixed in the software.

1.6 *Running* `cdtest` *with an Operating System CD-ROM*

The *SunDiag User's Guide* mistakenly states that you cannot use an operating system CD-ROM when testing your CD-ROM drive with `cdtest`. You can use `cdtest` with the Solaris operating environment CD-ROM.

To run the `cdtest` with an Operating System CD-ROM:

1. **Insert the Solaris CD-ROM into the drive under test.**
2. **Start the SunDiag software.**
3. **Display the `cdtest` Option Menu.**
4. **Select SunOS from the CD Type menu.**
   See Figure 1-3.

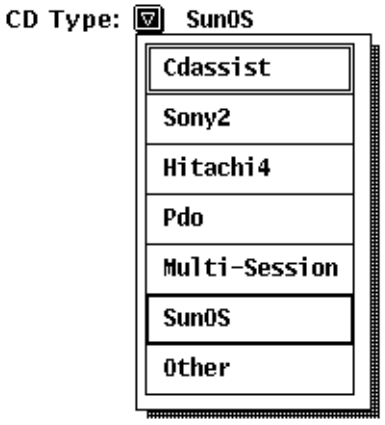


Figure 1-3 Selecting SunOS from the `cdtest` Option Menu

5. **Click the Apply button on the option menu.**

CPU Test Descriptions 2

These chapters describe the tests that are designed to test specific SMCC products. These tests will be displayed under the CPU Devices section of the SunDiag control panel:

2

2.1 *Sun Multimedia Codec Test (*`audbri`*)*

This test checks the functionality of several different Sun Multimedia Codec 16-bit audio options. Depending on the system under test, different subtests will be available for testing.

For a system with a dual rate ISDN (DBRI) chip and a Sun SpeakerBox™ attached, the following subtests are available:

- Crystal test
- Loopback test
- Calibration function
- Controls test
- Audio test

For a SPARCstation LX system (with an on-board DBRI/Codec audio chip without a SpeakerBox attached) the following subtests are available:

- Crystal test
- Audio test

For a SPARCstation 5 system (with an on-board Codec audio chip) the following subtests are available:

- Loopback test
- Audio test

Upon start-up, the SunDiag probe determines which audio devices are present, and it will limit the `audbri` Option Menu accordingly.

2.1.1 `audbri` *Test Descriptions*

Crystal Test

The crystal subtest measures the accuracy of the crystal that generates the sample rate clock. It does this by playing a 1 second signal and then measuring the actual time it takes for that signal to be played. This measurement is performed for each of the 8 standard sample rates.

Loopback Tests

Loopback subtests verify the performance of these audio ports: the headphone port, microphone port, line-in port, and line-out port. This subtest plays and records a known signal, calculates play and record gain, and analyzes the S/N plus distortion at various sample frequencies. It also measures the channel separation at each of the sample frequencies. Both the line-out/line-in and headphone/line-in loopback subtests require a stereo loopback cable.

Note – The speaker/microphone loopback subtest requires special hardware, and is used by manufacturing centers and special test facilities. Do not invoke the Speaker/Microphone loopback test unless you have the special hardware.

Controls Test

This is an interactive subtest which tests the three control buttons on the Sun SpeakerBox. This subtest plays music and you are asked to press the Volume Down, Volume Up, and Mute buttons in a specified order. If there is no input from while this subtest, the music will play for about 30 seconds, stop, and return an error.

Audio Test

You decide if this subtest passes or fails. A short selection of music is played. If you decide the music sounds adequate, then the subtest passes. If you do not hear the music, or it is badly distorted, then you know there is a problem.

2.1.2 `audbri` *Option Menus*

Upon start-up, the SunDiag probe determines which audio devices are present, and it will limit the `audbri` option Menu accordingly. The three possible option menus are shown below.

2.1.2.1 `audbri` *Option Menu with a SpeakerBox*

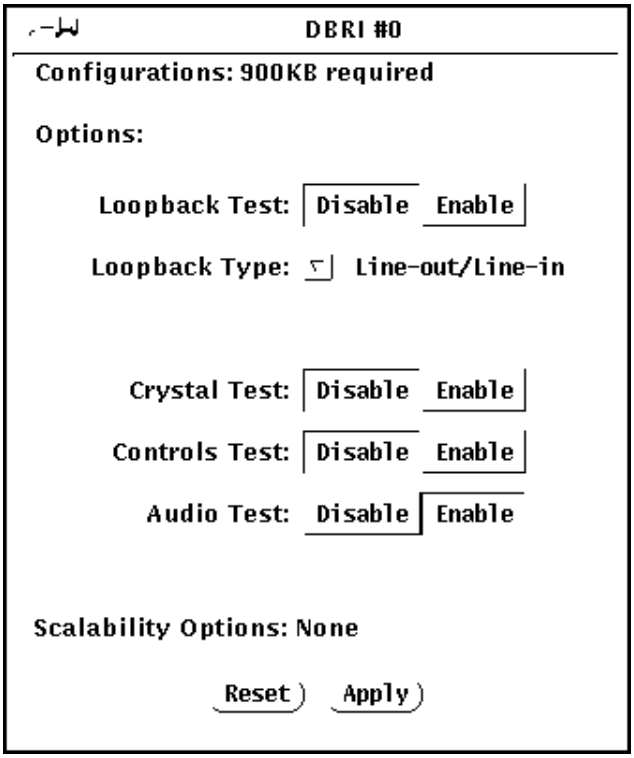


Figure 2-1 `audbri` Options Menu with a SpeakerBox Attached

2.1.2.2 `audbri` *Option Menu for SPARCstation 5*

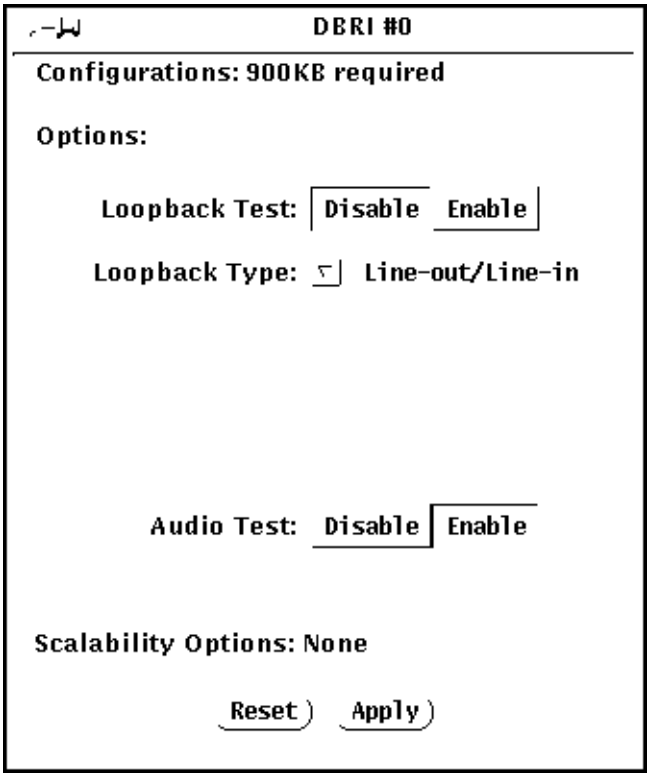


Figure 2-2 `audbri` Option Menu for a SPARCstation 5

2.1.2.3 `audbri` *Option Menu for a SPARCstation LX without a SpeakerBox*

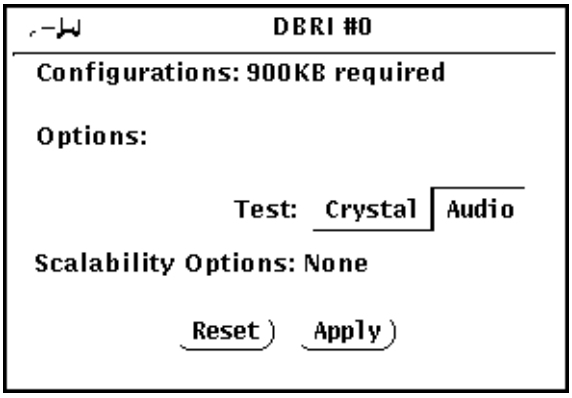


Figure 2-3 `audbri` Option Menu for a SPARCstation LX

2.1.2.4 `audbri` *Option Descriptions*

Note – The Calibration and Reference File options can only be selected through the command line.

Type

Press MENU to select the type of test to run. The choices are Line-in / Line-out and Line-in / Headphone.

Loopback

This exclusive setting enables you to toggle the Loopback subtest on and off.

Calibration

Used with the SpeakerBox to Microphone Loopback subtest. This exclusive setting enables you to toggle the Calibration function on and off. When enabled, this function creates calibration files for the Loopback subtest to use as a baseline in future testing. The default reference file names are listed below.

Crystal Test

Click SELECT to enable or disable the Crystal subtest.

Controls Test

Click SELECT to enable or disable the Controls subtest.

Audio Test

Click SELECT to enable or disable the Audio test. This is the only subtest enabled by default.

Reference File

Used with the SpeakerBox to Microphone Loopback subtest. If the SpeakerBox to Microphone Loopback subtest and Calibration function are enabled, a new calibration file will be created.

If the SpeakerBox to Microphone Loopback subtest is enabled and the Calibration function is disabled, the Loopback checks this text field for the calibration file to test against. The default reference file created in `/opt/SUNWdiag/bin` is:

`audbri_sbmic.data`

2.1.3 `audbri` *Command Line Syntax*

`/opt/SUNWdiag/bin/audbri B C D=/dev/sound/`*<unit_no>* **`F=`***reference_file_path* **`I=/dev/`***ioctl_device* **`M L S T=`***loopback_test_type* **`X`** *standard_arguments*

Arguments	
`B`	Brief test. This is the same as specifying **`q`** for quick test.
`C`	Loopback Calibration for SpeakerBox to Microphone
`D=/dev/`*audio_device*	Specifies the audio device to be tested. The default is **`D=/dev/sound/`***<unit_no>*
`F=`*reference_file_path*	The default files created are: `/opt/SUNWdiag/bin/audbri_sbmic.data` If you use others, specify that path and filenames with this option. See T=*loopback_test_type*.

Arguments	(Continued)
I=/dev/*ioctl_device*	Specifies the audio ioctl device to be tested; the default is /dev/audioct/<*unit_no*>.
M	Directs this subtest to run the Audio test.
L	Run the Loopback Test.
S	Run the Sun SpeakerBox Controls and Mute Indicator Test.
T=*loopback_test_type*	Specifies the type of Calibration / Loopback Test. The choices are listed below. The default is **1**. **0** Speaker / Microphone Note: this subtest requires special hardware, and is used by manufacturing centers and special test facilities. Do not invoke the Speaker / Microphone loopback test unless you have the special hardware. **1** Line-in / Line-out **2** Line-in / Headphone
X	Run the Audio Crystal Test

2.1.4 audbri *Quick Test Description*

Running this test in quick mode restricts testing to the Crystal test only.

2.2 *ISDN/DBRI Test (*`isdntest`*)*

This test verifies the functionality of the ISDN portion of the Dual Basic Rate ISDN (DBRI) chip.

2.2.1 `isdntest` *Test Description*

`isdntest` is actually a set of several subtests. Three main channels exist within an ISDN: D, B1 and B2. In each of the following subtests, unless otherwise indicated, the D channels will be in Basic Rate HDLC data mode, the B1 channels in 56 kbps HDLC data mode, and the B2 channels in 64 kbps HDLC data mode. D channel packet size is 256 bytes, and B channel packet size is 1024 bytes. The packet count will be 10 packets. Each channel runs as an independent child process.

The first subtest is the local loopback test. It first checks the initial activation state of the Network Termination (NT) and Terminal Equipment (TE) interfaces to make sure they are deactivated. It then activates each interface using the “force activation” capability of DBRI. Each interface is then put into local loopback mode (See Figure 2-4). Data residing in host memory is written to each interface, which loops the data back onto itself. The data is then read back into host memory and verified. Each channel, D, B1 and B2 is tested, with the exception of the TE D channel, which cannot be tested in local loopback mode. This test runs internal to the DBRI chip. This subtest does *not* require an NT to TE external loopback connector.

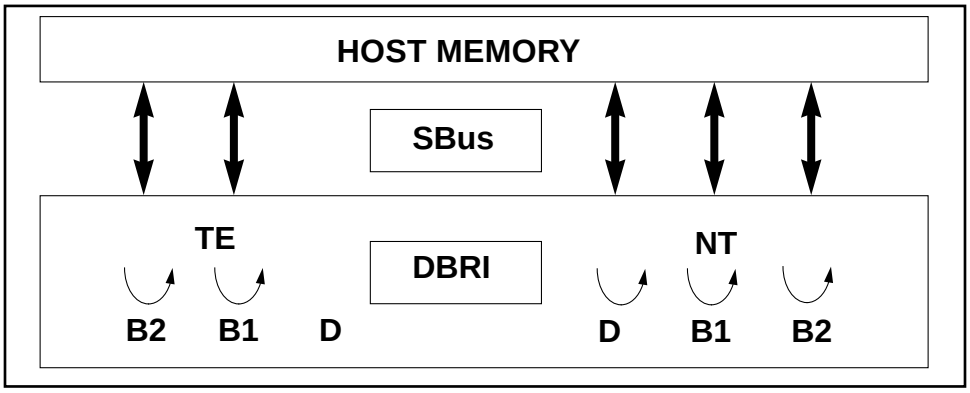


Figure 2-4 `isdntest` Local Loopback Subtest

The next subtest is the activation / deactivation test. This subtest runs through the activation / deactivation sequence for the NT and then the activation sequence for the TE. The T101 and T103 timers are set to 5 seconds. This subtest requires an NT to TE external loopback connector.

The remote loopback capability is tested next. The TE interface is put into remote loopback mode, and the NT transmits data to the TE on all three channels, D, B1 and B2 (See Figure 2-5). The TE loops all data back to the NT and reads a copy of it. Data is then verified. The whole process is then repeated with the TE transmitting to the NT, which is placed in remote loopback mode. This subtest requires an NT to TE external loopback connector.

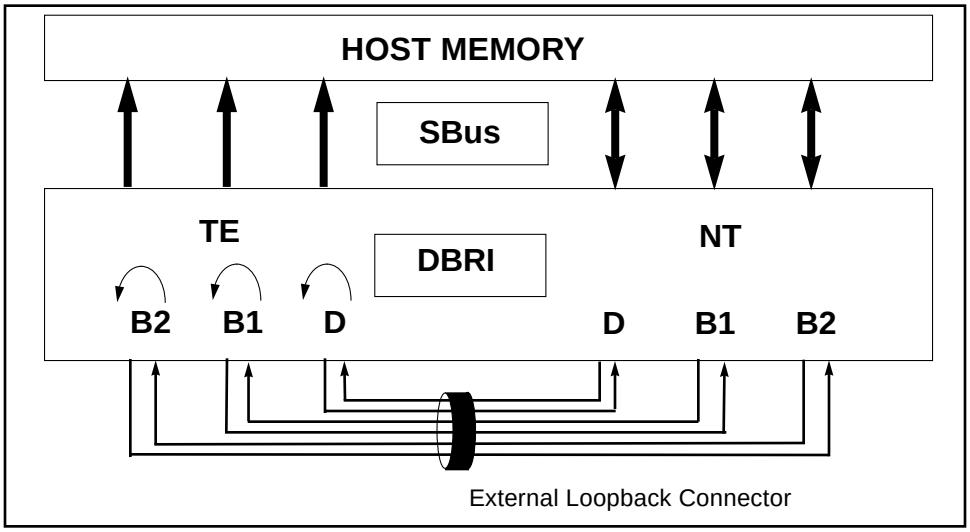


Figure 2-5 `isdntest` Remote Loopback Subtest

Next, a read / write test is performed on all 6 of the ISDN channels: TE D, TE B1, TE B2, NT D, NT B1 and NT B2. The external loopback connector connects each channel on the TE interface to its corresponding channel on the NT (See Figure 2-6). Six unique data patterns are used, one for each path. Packets read are compared against packets written. The test is repeated with the B1 channels placed in 64 kbps HDLC data mode and the B2 channels in 56 kbps HDLC data mode. This subtest requires an NT to TE external loopback connector.

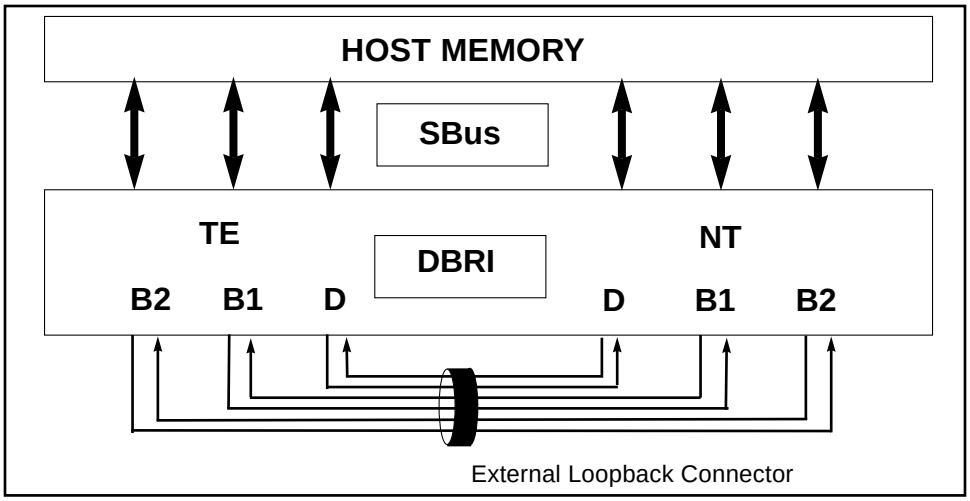


Figure 2-6 `isdntest` Read / Write Subtest

The next subtest is a packet size test. A read / write test, similar to the previous one, is performed with a packet count of 100. Each packet transmitted and received is a unique size, computed randomly. This subtest requires an NT to TE external loopback connector.

The last subtest is a data path test. Using the ISDN_SET_CHANNEL ioctl, data is routed through a series of short pipe interconnects within DBRI (See Figure 2-7). This subtest requires an NT to TE external loopback connector.

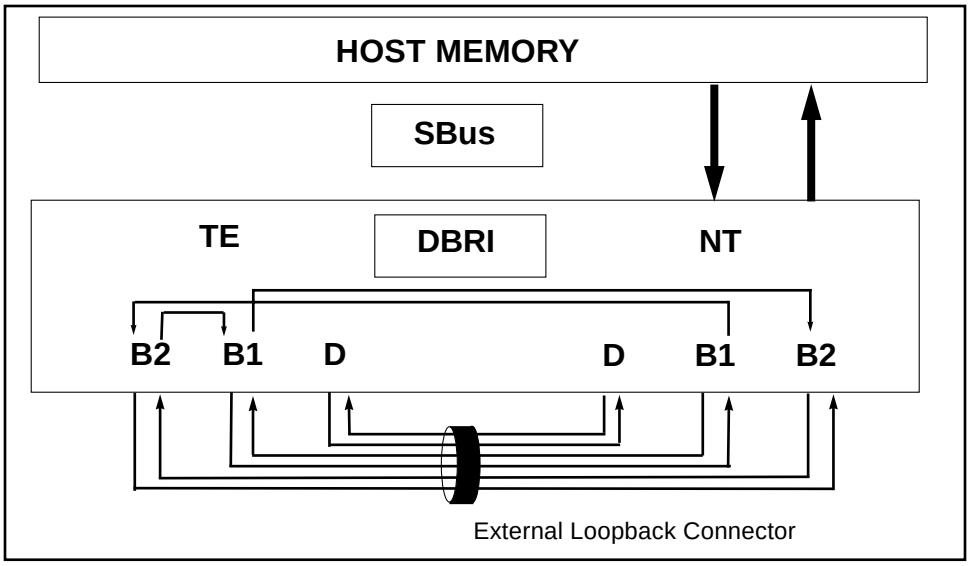


Figure 2-7 `isdntest` Data Path Subtest

2.2.2 `isdntest` *Options*

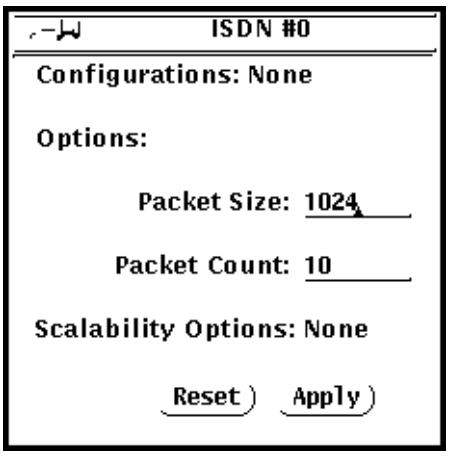


Figure 2-8 `isdntest` Options Menu

Packet Size

Packet Size indicates the size, in bytes, of the B channel packets. The default size is 1024 bytes for the B channels and 256 for the D channels. The maximum packet size is 8186 bytes for the B channels, and the minimum size is 1 byte. The D channel packet size will always be set to 256, except during the packet size test, where it is set to random values between 1 and 256.

Packet Count

Packet Count indicates how many packets are to be transmitted and received for all channels. The default packet count is 10 packets. The maximum packet count is 100 packets.

2.2.3 `isdntest` *Command Line Syntax*

`/opt/SUNWdiag/bin/isdntest` **`S=`***packet_size* **`C=`***packet_count standard_arguments*

Arguments	
`S=`*packet_size*	*packet_size* is the size, in bytes, of the B channel packets. The default size is `1024` bytes for the B channels and 256 for the D channels. The maximum packet size is 8186 packets for the B channels, and the minimum size is 1 packet. The D channel packet size will always be set to 256, except during the packet size test, where it is set to random values between 1 and 256.
`C=`*packet_count*	*packet_count* indicates how many packets are to be transmitted and received for all channels. The default count is `10` packets, and the maximum packet count is 100 packets.

2.2.4 `isdntest` *Quick Test Description*

Running this test in quick mode restricts testing to the local loopback subtest only.

2.2.5 `isdntest` *Error Messages*

`Initial state on /dev/isdn/0/nt/mgt is ISDN_ACTIVATED`

Using the NT management device driver on device 0, the initial activation state on the NT interface is ISDN_ACTIVATED, which is incorrect.

```
Unable to activate with /dev/isdn/1/te/mgt.
TE state = ISDN_DEACTIVATED    NT state = ISDN_DEACTIVATED
```

Using the TE management device driver on device 1, the TE and NT interfaces did not activate within the allowed period of time. The current activation state on both interfaces is ISDN_DEACTIVATED.

```
Data miscompare for NT B2 channel reader.
Packet 6 offset 58 contains C8, should be A9.
```

The NT B2 channel was comparing packets read to those written and found a miscompare.

2.3 *Graphics Tower (*`gttest`*)*

SunDiag tests the Sun Graphics Tower with a sequence of subtests that can accurately locate and identify failing FRUs (Filed Replaceable Units). All tests are nondestructive and maintain the system integrity during and after the tests are run.

Caution – Do not run any other application that uses the GT accelerator port while running `gttest`. This combination will cause SunDiag to return incorrect errors.

Note – `gttest` requires approximately 1.5M bytes of disk space in the `/tmp` directory to extract its working files. If this space is not available, the diagnostic will fail and report warning and error messages indicating lack of disk space.

By default, SunDiag runs all of the available tests, except the Stereo test. See the Test Descriptions section below.

Note – To avoid excessive test cycle times when testing the GT Graphics Subsystem, follow these instructions:

1. Enable Single Pass on the SunDiag Options menu.
2. Enable Verbose on the SunDiag Options menu.
3. Do not select any other diagnostic tests.

Following these procedures ensures that `gttest` will run once, report its status as each test routine executes, and then exit.

Note – Disable all screen savers before testing any graphics device. Type **`xset s off`** at a UNIX prompt to disable the Solaris screen saver.

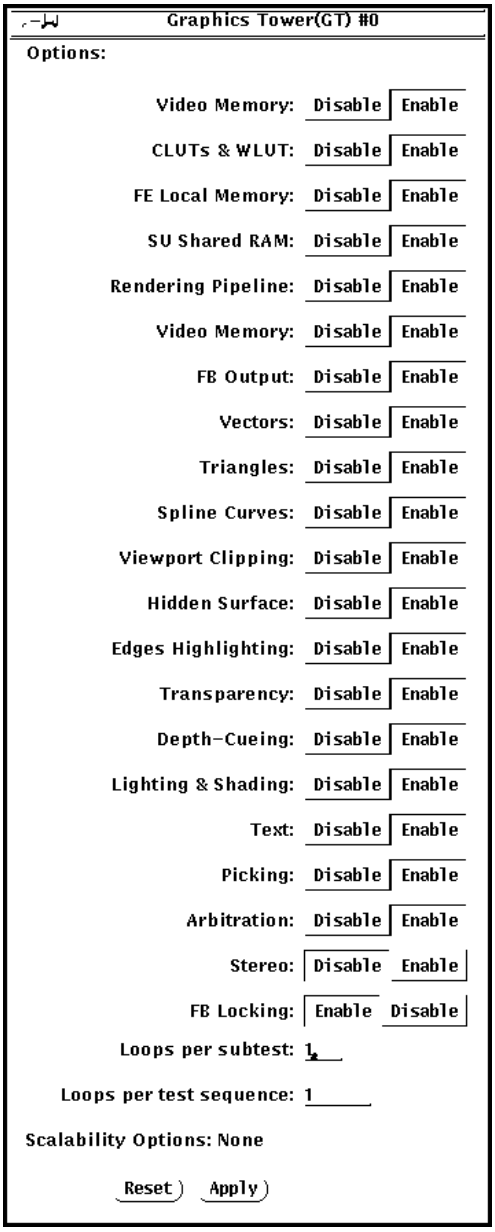


Figure 2-9 `gttest` Options Menu

2.3.1 `gttest` *Test Description*

The subtests are run in the order shown in Figure 2-10. The subtests assume that the GT Graphics Subsystem has an active working interface with the SPARCstation CPU. If for any reason SunDiag cannot make the connection, further testing is not possible and SunDiag will report a fatal unrecoverable error.

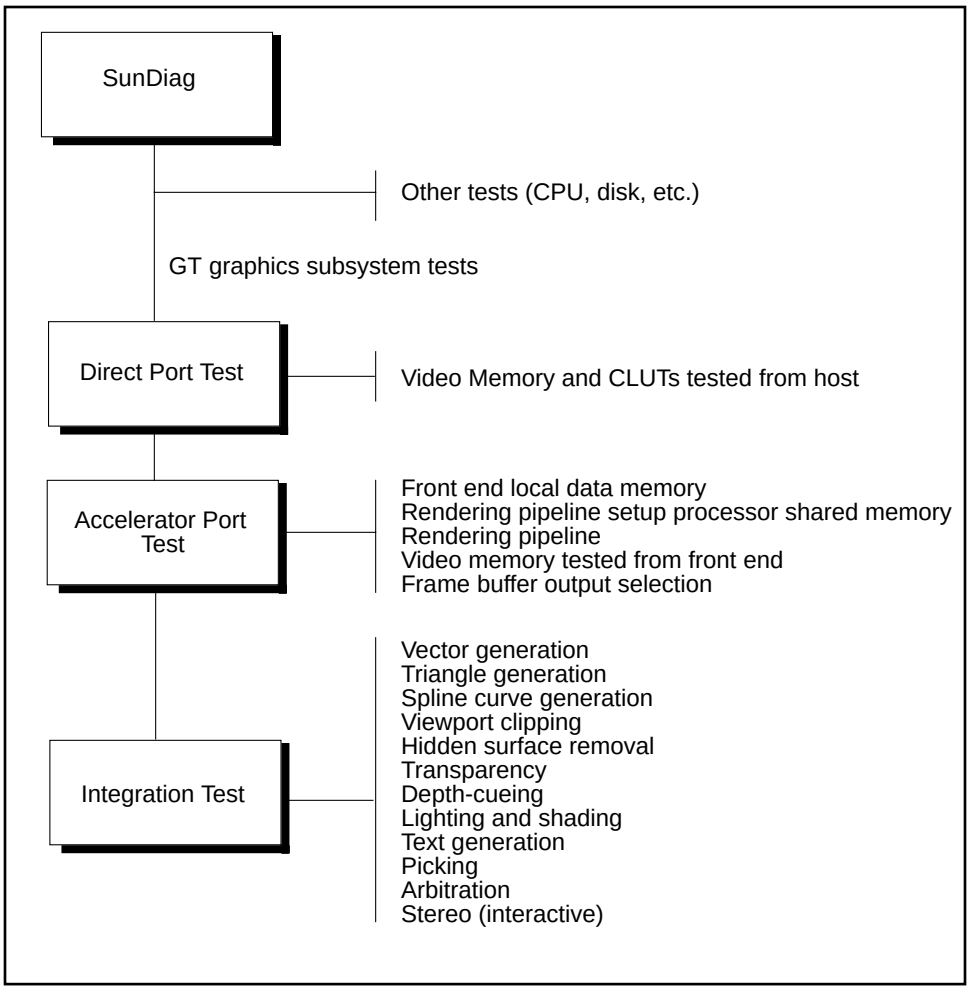


Figure 2-10 `gttest` Overview

2.3.1.1 Direct Port Tests

The direct ports tests check the non-accelerated portion of the GT using the following subtests.

Video Memory Array, Tested from Host

This subtest checks the frame buffer.

The video memory array subtest selects and tests 64 by 64 pixel regions covering all video memory planes, including the 2 8-bit alpha/overlay planes, 2 24-bit image planes, 24-bit depth (Z buffer) plane, 10-bit WID plane, and two cursor planes. If the subtest detects an error, SunDiag reports the defective plane and location.

CLUTs

This subtest checks the frame buffer.

This subtest performs a non-destructive read-write test on the frame buffer color look up tables. If this subtest detects a failure, SunDiag reports the location of the failure.

At the beginning of this subtest, red, green, and blue stripes display for visual verification of the digital-to-analog converters (DACs).

2.3.1.2 Accelerator Port Tests

The accelerator port tests check the accelerated portion of the GT using the following subtests.

Front End Local Data Memory

This subtest checks the Graphics Processor Front End Board Local Data Memory.

The Local Data Memory subtest is a nondestructive read-write memory test. This subtest aborts at the first error and reports a memory failure.

Setup Processor Shared Memory Test

This subtest checks the graphics processor rendering pipeline.

The Setup Processor shared memory subtest verifies that the i860 microprocessor can write and read the setup processor shared memory without error.

Rendering Pipeline

This subtest checks the graphics processor rendering pipeline.

The rendering pipeline subtest checks each of the three rendering pipeline stages: setup processor, edge walker, span interpolator, and Pixel Bus Multiplexer. The Edge Walker and Span Interpolator subtests are a series of small tests that verify the functionality of the edge walker and span interpolator ASICs in vectors, triangles, Gouraud shading, alpha blending, and anti-aliasing rendering. The results of the tests are verified by means of checksum values accumulated from data output by the Rendering Pipeline. SunDiag reports any subtest failures.

Video Memory Array, Tested from Front End

This subtest checks the frame buffer.

This test makes sure that the video memory array can be accessed by the i860 microprocessor via the Local Bus. This is a destructive read-write test which verifies that all the Frame Buffer video memory locations are good. If this subtest detects an error, SunDiag reports the defective plane and location.

Frame Buffer Output Section

This subtest checks the frame buffer.

The Frame Buffer Output Section contains a diagnostic feedback register in the RAMDACs. The Frame Buffer Output Section subtest creates various windows in the Window ID plane then sets up the look up tables (LUTs) associated with these values. This subtest then writes known values to the video memory of these windows. Next, the frame buffer is switched into trace mode, which reads the video through the diagnostic feedback register and puts the data into the shadow memory. Finally, this subtest compares the contents of the shadow memory with the expected values via checksum and determines if the Output Section is operating properly.

If this subtest detects an error, the test reports the error, and the actual and expected values are displayed on the system Console.

2.3.1.3 Integration Test

The integration test is a sequence of subtests running GT display list programs. These subtests ensure the GT Graphics Subsystem integrity at the system level. The subtests test all features of the hardware at the application level, reading the results from the frame buffer and verifying the results by comparing against known good images.

These tests use a frame buffer region of 1152 by 900 pixels, in the upper left corner of the screen, regardless of the size screen attached to the system. The tests use previously-generated test images for each color plane (red, green, and blue). These test images are stored in a reduced size (1/64th the normal size) to save disk space.

Vector Generation

This subtest tests all Graphics Tower boards. The faulty component can be determined by analyzing the errors reported in the Direct Port and Accelerator Port tests.

This subtest renders fairly large vector objects with aliased, anti-aliased, and shaded vectors.

Triangle Generation

This subtest tests all Graphics Tower boards. The faulty component can be determined by analyzing the errors reported in the Direct Port and Accelerator Port tests.

This subtest renders objects with aliased, anti-aliased shaded, and shaded triangles.

Spline Curves Generation

This subtest tests all Graphics Tower boards. The faulty component can be determined by analyzing the errors reported in the Direct Port and Accelerator Port tests.

This subtest renders an object with both parametric and NURBS[1] curves of different orders.

1. Non-Uniform Rational B-Splines.

Viewport Clipping

This subtest tests all Graphics Tower boards. The faulty component can be determined by analyzing the errors reported in the Direct Port and Accelerator Port tests.

This subtest renders and clips an object around and in front on the screen.

Hidden Surface Removal

This subtest tests all Graphics Tower boards. The faulty component can be determined by analyzing the errors reported in the Direct Port and Accelerator Port tests.

This subtest renders objects with the Z-buffer-compare attribute turned on.

Polygons Edges Highlighting

This subtest tests all Graphics Tower boards. The faulty component can be determined by analyzing the errors reported in the Direct Port and Accelerator Port tests.

This subtest renders an object with the edge highlighting attribute turned on.

Transparency

This subtest tests all Graphics Tower boards. The faulty component can be determined by analyzing the errors reported in the Direct Port and Accelerator Port tests.

This subtest renders a scene with two transparency modes (standalone and alpha blend) in various degrees. This results in a two-pass transparency of the objects in the scene.

Depth-Cueing

This subtest tests all Graphics Tower boards. The faulty component can be determined by analyzing the errors reported in the Direct Port and Accelerator Port tests.

This subtest renders an object with the depth-cueing attribute turned on.

Lighting and Shading

This subtest tests all Graphics Tower boards. The faulty component can be determined by analyzing the errors reported in the Direct Port and Accelerator Port tests.

This subtest renders an object with multiple light sources and Gouraud shading for front and back surfaces.

Text Generation

This subtest tests all Graphics Tower boards. The faulty component can be determined by analyzing the errors reported in the Direct Port and Accelerator Port tests.

This subtest generates diverse text lines with different attributes for checking.

Picking

This subtest tests all Graphics Tower boards. The faulty component can be determined by analyzing the errors reported in the Direct Port and Accelerator Port tests.

This subtest has two parts: a pick detect test and a pick echo test.

Animation and Arbitration

This subtest checks the frame buffer.

This subtest renders a moving, double-buffered object into the image plane while a second Solaris process performs a read-write test to the cursor and WID planes from the direct port on the Frame Buffer. This subtest simulates conditions in the real world, where rendering processes and windows operations run concurrently.

Stereo (Interactive)

This subtest checks the frame buffer.

This subtest displays an object in stereo mode. You must verify the proper operation by looking at the screen with stereo glasses. To terminate to the test, you must press **q**.

FB Locking

See the “Special Note on Testing Multiple Framebuffers” section in Chapter 1 of the *SunDiag User’s Guide* for details.

2.3.2 `gttest` *Command Line Syntax*

`/opt/SUNWdiag/bin/gttest D=`*device_name* **`S=`***subtest_number* **`F=`***#_of_subtest_loops* **`B=`***#_of_test_loops* **`L`** *standard_arguments*

Arguments

`D=`*devicename*

devicename is the full path name of the device under test. The default is `/dev/gt0`.

`S=`*subtest_number*

subtest_number is the test number of the subtest to be run. Select from the subtests below. You can run multiple subtests by adding the subtest numbers. For example, `n=0x3` runs both test 1 and test 2; `n=0x180` runs both test 0x080 and test 0x0100. Note that you do not need the leading zeros. To run all tests, type **`n=0x7FFFF`**.

`0x 000 001`	Direct port—video memories
`0x 000 002`	Direct port—CLUTs and WID LUT
`0x 000 004`	Accelerator port—front end Local Data Memory
`0x 000 008`	Accelerator port—setup processor shared memory
`0x 000 010`	Accelerator port—rendering pipeline
`0x 000 020`	Accelerator port—video memories
`0x 000 040`	Accelerator port—Frame buffer output section
`0x 000 080`	Integration test—vectors
`0x 000 100`	Integration test—triangles
`0x 000 200`	Integration test—spline curves
`0x 000 400`	Integration test—viewport clipping.
`0x 000 800`	Integration test—hidden surface removal
`0x 001 000`	Integration test—polygon edges highlighting
`0x 002 000`	Integration test—transparency
`0x 004 000`	Integration test—depth cueing
`0x 008 000`	Integration test—lighting and shading
`0x 010 000`	Integration test—text
`0x 020 000`	Integration test—picking
`0x 040 000`	Integration test—arbitration
`0x 080 000`	Integration test—stereo (interactive)

`F=`*#_of_subtest_loops*

#_of_subtest_loops is the number of loops for each subtest. The default is 1 (one loop)

Arguments	(Continued)
B=*#_of_test_loops*	*#_of_test_loops* is the number of loops of each test sequence. The default is 1 (one loop).
L	Disables framebuffer locking. See the "Special Note on Testing Multiple Framebuffers" in Chapter 1 of the *SunDiag User's Guide* for details.

2.3.3 `gttest` *Quick Test Description*

Running this test in quick mode does not change the test procedure.

2.3.4 `gttest` *Command Line Examples*

Here are three examples of SunDiag `gttest` on-line commands. Make sure to change directories to `/opt/SUNWdiag/bin` before running `gttest` from the command line. `gttest` is hard-wired to look for its data file, `gttest.data`, in `/opt/SUNWdiag/bin`.

1. A Simple accelerator port test, Frame Buffer output section, single pass:

```
# cd /opt/SUNWdiag/bin
# gttest S=0x40
```

2. All direct port tests, five loops of sequence:

```
# cd /opt/SUNWdiag/bin
# gttest S=0x3 B=0x5
```

3. All subtests (except the interactive tests), two loops of each subtest, four loops of each test sequence:

```
# cd /opt/SUNWdiag/bin
# gttest S=0x7FFFF F=2 B=4
```

2.3.5 `gttest` *Error Messages*

The GT SunDiag error messages are described below. The error messages are listed in alphabetical order.

`Arbitration Test: Accelerator port drawing in double buffer mode, [plane group], [mode] Mode error at x=[x] y=[y], exp=[expected], obs=[observed], xor=[xor]. Suspect daulty HFB.`

Failed the Arbitration integration test. [plane group] is one of the following: Red Plane Group A, Overlay Plane Group A or B, Image Plane Group A or B, Cursor Plane Group, Cursor Enable Plane group, Z Buffer Plane Group, Hardware Window ID Plane Group, or Software Window ID Plane Group. [mode] is either byte or stencil access mode. The location of the anomaly as well as the expected and observed values are also given. For pixel access mode, the bank of the memory error is disclosed as well. “HFB” if the GT Frame Buffer board.

`Arbitration Test: Accelerator port drawing in double buffer mode, Direct port simultaneous write to both buffers, read from buffer [A or B] : [plane_group], Shapes error at [address]. Suspect faulty HFB.`

Failed the Arbitration integration test. [plane_group] is one of the following: Image plane A or B, Depth plane, WID plane, Cursor plane, or Fast Clear Plane A or B. [address] is the linear address of the bad memory cell. “HFB” is the GT Frame Buffer Board.

`Background process wouldn’t die. System error.`

A software error. You may have to re-boot the SPARCstation.

`Byte/Stencil Access Mode error at x=# y=#, exp=0x#,obs=0x#, xor=0x#. Suspect faulty HFB.`

The direct port video memory test has found an error at pixel (x,y) in the current plane group. Byte/Stencil Access Mode applies to all plane groups that access all 32 bits of the frame buffer memory (in other words, the eight bit image and overlay planes). The test expected to find *exp* but observed obs, yielding xor when the two values are exclusive or’d with each other. There may be a bad bit in the video memory.

`Cannot allocate enough memory for testing. Check swap space size and memory reserved for test.`

Out of memory error. Increase swap space and/or kill other processes.

`Cannot create screen raster for device [device]. Check device for existence and/or permissions.`

The device that you specified (the default is `/dev/fbs/gt0`) is not available to the test. Make sure that you are executing the test on a machine with a GT, and that you have permission to access it.

`Can't open display list file [filename]. Suspect incomplete or incorrect hardware installation. Files may also have been corrupted because file system ran out of space in /tmp.`

Indicates a software initialization problem. [filename] is the file that SunDiag can't open. Also, approximately 1.5 Mbytes of free space is required in /tmp.

`Can't open [filename] to dump frame buffer.`

The test needed to open a file to dump the contents of the frame buffer. Use `df` to check drive space, and also to check file permissions.

`Can't read display list file [filename]. Suspect incomplete or incorrect hardware installation. Files may also have been corrupted.`

Indicates a software initialization problem. [filename] is the display list file that SunDiag can't read.

`CLUT #n, index #, color [color], expected 0x#, received 0x#.`

An error was found in one the fifteen color look up tables tested by SunDiag. The error was found in the nth CLUT. The index is out of 256 entries in each CLUT. Each CLUT has eight bits each for red, green, and blue. The color indicates in which set of eight bits the error was found. By using the expected and received values, you can figure out which bit is incorrect.

`Communication with Graphics Engine failed. Suspect incomplete or incorrect software installation. Front-End Firmware may also be dead.`

SunDiag is unable to communicate with the GT Graphics Subsystem. A software error or hardware error with the GT SBus Adapter Board, HAC Cable, or Front End Board.

`Cursor plane error: Failed with error code [code]: [failure]. Suspect faulty HFB.`

Failed accelerator port video memory test. [code] is the error code number. [failure] is an explanation of the error indicated by [code]. “HFB” is the GT Frame Buffer Board.

`Depth cueing: Error(s) found in [RED], [GREEN], [BLUE} components.`

Failed the depth cueing integration test. Only the failing component (RED, GREEN, or BLUE) appears in the message.

`Depth plane error: Failed with error code [code]: [failure]. Suspect faulty HFB.`

Failed accelerator port video memory test. [code] is the error code number. [failure] is an explanation of the error indicated by [code]. “HFB” is the GT Front End Board.

`Display list file is too big for the remaining VM! hdl_size=0x#, vm_size=0x#`

Increase swap space or close other running processes.

`Failed to allocate unique WID for 24-bit plane. Suspect incomplete or incorrect software installation.`

A problem with system initialization.

`Failed to get monitor mode: [ERROR] Software error.`

A software error. May have to re-boot the SPARCstation.

`Failed to set diagnostic mode. Software error.`

A software error. May have to re-boot the SPARCstation.

`Failed to set monitor mode. Software error.`

A software error. May have to re-boot the SPARCstation.

`Fast Clear Plane [A or B] error: Failed with error code [code]: [failure]. Suspect faulty HFB.`

Failed accelerator port video memory test. [code] is the error code number. [failure] is an explanation of the error indicated by [code]. “HFB” is the GT Front End Board.

`At FE firmware program counter 0x#, expected display list instruction 0x#, observed 0x#.`

You may have a bad Front End processor board.

`Front End (Firmware) not responding. This may indicate that the Firmware has died. Try to run gtconfig.`

A hardware problem. SunDiag was unable to communicate with the Front End Board. Indicates a problem with the SBus Adapter Board, HAC cable, or Front End Board.

`Got XCPU interrupt, but user_mcb_ptr->trap_instruction = 0x#, expect 0x#. System software error.`

A software error. May have to re-boot the SPARCstation.

`Got XCPU interrupt, but it’s not a trap instruction, error code = # : [message]. This may indicate that the Firmware has died. Try to run gtconfig.`

A software error. [message] further describes the problem. May have to re-boot the SPARCstation.

`Hidden Surface Removal: *** Error(s) found in [RED], [GREEN], [BLUE} components.`

Failed the hidden surface removal integration test. Only the failing component (RED, GREEN, or BLUE) appears in the message.

`hk_disconnect failed. System software error.`

A software error. May have to re-boot the SPARCstation.

`hk_munmap failed. System software error.`

A software error. May have to re-boot the SPARCstation.

`hk_open failed. GT system is either not initialized or not connected.`

A software error. May have to re-boot the SPARCstation.

`Image plane [A or B] error: Failed with error code [code]: [failure]. Suspect faulty HGPFE.`

Failed accelerator port video memory test. [code] is the error code number. [failure] is an explanation of the error indicated by [code]. “HGPFE” is the GT Front End Board.

`LDM error: Failed with error code [code]: [failure]. Suspect faulty HGPFE.`

Failed the accelerator port test of the Front End Local Data Memory. [code] is the error code number. [failure] is an explanation of the error indicated by [code]. “HGPFE” is the GT Frame End Board.

`Lighting and Shading: *** Error(s) found in [RED], [GREEN], [BLUE] components.`

Failed the lighting and shading integration test. Only the failing component (RED, GREEN, or BLUE) appears in the message.

`Pick Detect misses:%d lines and/or triangles inside the pickbox and/or %d lines and triangles outside the pickbox.`

Failed the Picking integration test. Only the failing component (RED, GREEN, or BLUE) appears in the message.

`Pick Echo failed: *** Error(s) found in [RED], [GREEN], [BLUE} components.`

Failed the Picking integration test. Only the failing component (RED, GREEN, or BLUE) appears in the message.

`Picking: *** Error(s) found in [RED], [GREEN], [BLUE} components.`

Failed the picking integration test. Only the failing component (RED, GREEN, or BLUE) appears in the message.

`Pixel Access Mode error at x=# y=#, bank=#, exp=0x#,obs=0x#, xor=0x#. Suspect faulty HFB.`

The direct port video memory test has found an error at pixel (x,y) in the current plane group. Pixel Access Mode applies to all plane groups that do not access the frame buffer memory four bytes at a time. (In other words, all planes except eight bit planes). The memory for the pixel resides in the given memory bank. The test expected to find `exp` but observed `obs`, yielding `xor` when the two values are exclusive or'd with each other. There may be a bad bit in the video memory.

`Poly Edges Highlighting: *** Error(s) found in [RED], [GREEN], [BLUE} components.`

Failed the poly edges highlighting integration test. Only the failing component (RED, GREEN, or BLUE) appears in the message.

`Rendering Pipeline error: Failed with error code [code]: [failure]. Suspect faulty HGPRP.`

Failed the accelerator port Rendering Pipeline test. [code] is the error code number. [failure] is an explanation of the error indicated by [code]. "HGPRP" is the GT Front End Board.

`Spline Curves: *** Error(s) found in [RED], [GREEN], [BLUE} components.`

Failed the spline curves integration test. Only the failing component (RED, GREEN, or BLUE) appears in the message.

`SU Shared RAM error: Failed with error code [code]: [failure]. Suspect faulty HGPRP.`

Failed the accelerator port Rendering Pipeline Setup Processor shared RAM test. [code] is the error code number. [failure] is an explanation of the error indicated by [code]. "HGPRP" is the GT Front End Board.

`System initialization failed. Suspect incomplete or incorrect software installation. Front-End Firmware may also be dead.`

System initialization failed. Most likely a hardware problem with the GT Front End Board.

`'tar' never finished. System software problem.`

Make sure that the tar program is installed correctly on your system. Also, use `df` to see if you have enough disk space left in your `/tmp` directory.

`tar: [error]`

Make sure that the tar program is installed correctly on your system. Also, use `df` to see if you have enough disk space left in your `/tmp` directory.

`Texts: Error(s) found in [RED], [GREEN], [BLUE} components.`

Failed the texts integration test. Only the failing component (RED, GREEN, or BLUE) appears in the message.

`This program requires the default setting of 5 HW and 5 SW WID Planes. Run gtconfig.`

Each of the GT's ten WID planes can be allocated as a hardware or software WID plane. Some programs may change the allocation from the default (5 hardware, 5 software) and forget to change it back. Run `gtconfig` to set the allocation back to the default.

`This program requires the default setting of 5 HW and 5 SW WID Planes. Enter "unsetenv XNEWS_WID_PLANES" and try again.`

Each of the GT's ten WID planes can be allocated as a hardware or software WID plane. Programs that use the Shapes libraries that need to change the allocation from the default (5 hardware WID planes and 5 software WID planes) do so by setting the environment variable XNEWS_WID_PLANES to the desired number of hard WID planes. A program has set this environment variable, and you must unset it in order to run the GT SunDiag tests.

`Transparency: *** Error(s) found in [RED], [GREEN], [BLUE} components.`

Failed the transparency integration test. Only the failing component (RED, GREEN, or BLUE) appears in the message.

`Triangles: *** Error(s) found in [RED], [GREEN], [BLUE} components.`

Failed the triangles integration test. Only the failing component (RED, GREEN, or BLUE) appears in the message.

`Unable to open /dev/fbs/gt0. Check device for existence and/or permission.`

SunDiag is unable to open the GT device driver. Make sure that there is a /dev/fbs/gt0 device driver and that the permissions are correct.

`Vectors: *** Error(s) found in [RED], [GREEN], [BLUE} components.`

Failed the vectors integration test. Only the failing component (RED, GREEN, or BLUE) appears in the message.

`vfork: [error]`

An error has occurred while trying to fork a child process. Increase swap space, or close other running processes.

`Viewport Clipping: *** Error(s) found in [RED], [GREEN], [BLUE} components.`

Failed the viewport clipping integration test. Only the failing component (RED, GREEN, or BLUE) appears in the message.

`WID plane error: Failed with error code [code]: [failure]. Suspect faulty HFB.`

Failed accelerator port video memory test. [code] is the error code number. [failure] is an explanation of the error indicated by [code]. “HFB” is the GT Front End Board.

2.4 *cgsix Frame Buffer, GX and GX+ Options Test (*`cg6test`*)*

`cg6test` verifies the cgsix frame buffer and the GX options offered with most SPARC™ based workstations and servers.

Note – Disable all screen savers before testing any graphics device. Type **`xset s off`** at a UNIX prompt to disable the Solaris screen saver.

2.4.1 `cg6test` *Test Description.*

This test stresses the frame buffer with the subtests described below.

Cursor Test

This is a visual test of the overlay registers of the RAMDAC. A pointer is drawn on the screen and moved around to predetermined locations. There is a problem if the pointer disappears. This visual test ensures that the overlay is working properly.

Fast Copy in Double Buffer Test Mode

Two full-size screen rasters images are created in double buffer mode. Different patterns are written to each of them. The hidden buffer is copied to the visible buffer, and the data is compared. An error message is returned if there are inconsistencies. Then the buffer is flipped and the process is repeated.

Note – This test only applies to Sun Microsystems GX+ graphic accelerators with double-buffering capacity.

TEC Test

The TEC verifies that the Transformation Engine and Cursor control logic are being accessed. This confirms that further TEC access will be performed correctly.

FBC Test

The FBC test verifies that the Frame Buffer Controller logic is being accessed. This confirms that further FBC access will be performed correctly.

Frame Buffer Test

This test verifies that the frame buffer memory is working. A walking 1's pattern is written to memory, with a specific color signifying one of eight bits. The screen is divided into eight equally wide vertical stripes. A walking one is written to each stripe, causing eight iterations of these stripes. The value written is read back and checked. If the values do not match, an error is reported.

Screen Test Using Blits

This test draws blocks of color and performs blit transfers to other portions of the screen. First, the entire screen is drawn with cyan color, then a black block is put in the upper left corner. This subtest blits this block on the upper right, lower right, and lower left corners, then or's the whole image.

Blit Test

A block of data is drawn, and blit into a location at the bottom right rectangle.

Line Test

Lines with different data values are drawn on the screen and appear in different colors. The data is read back and compared with the expected values. An error is returned in the case of a mismatch.

Polygon Test

Hourglass-shaped polygons are drawn on the screen, using the four vertices. After all the polygons are rendered in the video memory, they are read back and the data compared with expected values. If there is a mismatch, an error is displayed.

Colormap Test

All 256 locations in the color map are loaded with a greyscale both backwards and forwards manner. This means decreasing values are loaded to all R, G, and B values.

Warning – If the system under test has a monochrome or greyscale monitor, visual color problems are undetectable.

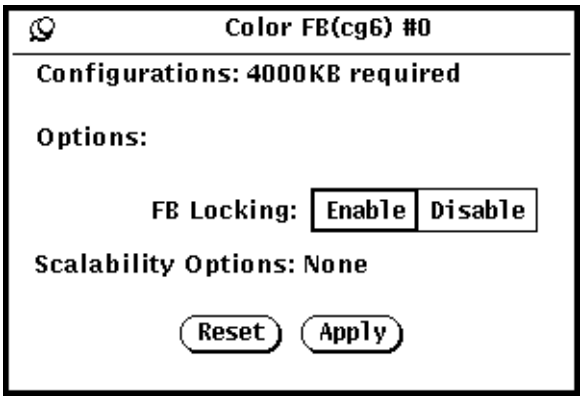


Figure 2-11 `cg6test` Options Menu

2.4.2 `cg6test` *Options*

FB Locking

See the "Special Note on Testing Multiple Framebuffers" section in Chapter 1 of the *SunDiag User's Guide* for details.

2.4.3 `cg6test` *Command Line Syntax*

`/opt/SUNWdiag/bin/cg6test L` *standard_arguments*

Arguments	
`L`	Disables framebuffer locking. See the "Special Note on Testing Multiple Framebuffers" in Chapter 1 of the *SunDiag User's Guide* for details.

Note – Extra Swap Space Required: 5 MB

2.4.4 `cg6test` *Quick Test Description*

Running this test in quick mode does not change the test procedure.

2.4.5 `cg6test` *Error Messages*

`cg6test` returns the error messages described below for subtest failures:

`CG6` <*subtest*> `failure, x pos =` <*failure location*>`, y pos =` <*failure location*>`, exp =` <*expected value*>`, actual =` <*actual value*>

The named subtest could not complete successfully.

`CG6plus VRAM(s) to probe: U#`

The polygon test prints these messages for a TurboGX framebuffer.

`Colormap error - red, loc =`<*failure location*>`, exp =` <*expected value*>`, actual =` <*actual value*>
`Colormap error - green, loc =`<*failure location*>`, exp =` <*expected value*>`, actual =` <*actual value*>
`Colormap error - blue, loc =`<*failure location*>`, exp =` <*expected value*>`, actual =` <*actual value*>

`Couldn't create new screen for` <*device*>`.`

SunDiag could not open up a memory area for simulating frame buffer memory, where: <*device*> `= /dev/fb, /dev/cgsix0, or /dev/cgsix`*n*

`Could not create child raster`

The color map test and frame buffer tried to create a child raster. If there is not enough memory available, this test may fail.

`Error in opening device /dev/cgsix`

The device could not be opened. A wrong device name was supplied. Rebooting with the **`-r`** option is required if new devices are installed in Solaris 2.x systems. See "New Device Drivers" on page 1-2 of this book.

`Error: ERROR_MALLOC_FAILED`

Not enough memory was available during color map and frame buffer memory testing.

`Failed to create raster`

The raster to do a graphics operation could not be opened. There might not be enough memory available, or the wrong raster name was supplied.

`Failed to get cmap`

Not enough memory was available.

`Failed to create context`

Not enough memory was available.

`render_main: can't map lego`

`cg6test` was not able to map the framebuffer board register addresses.

`render_main: TEC_EXCEPTION`
`build_view_matrix: TEC_EXCEPTION`

The TEC section of the frame buffer logic had an exception during the named routine.

`clear_window: DRAWSTATUS`

A DRAWSTATUS error occurred during the `clear_window` routine.

2.5 *cgtwelve Frame Buffer, GS Test (*`cg12`*)*

The GS is an integrated frame buffer and 3D graphics accelerator for desktop SPARCstations.

Note – The user interface is locked out while this test is being run. You can stop this test with the Stop button on the SunDiag window, or temporarily halt the test by pressing Control-C.

Note – Disable all screen savers before testing any graphics device. Type **`xset s off`** at a UNIX prompt to disable the Solaris screen saver.

2.5.1 `cg12` *Test Description*

The `cg12` test operates on two levels, the host level and the C30 level.

Note – The GS accelerator includes a Texas Instruments TMS320C30 DSP chip. Throughout this manual, it is referred to as the C30.

The Host Level

The host level includes test code based on the SunView window system, the GPSI libraries, and a specially-compiled version of the Pixrect™ libraries. This is the main part of the `cg12` test. The test results are verified by reading out the image memory with the Pixrect library, reducing the resolution, and then verifying with the supplied test images. The test images are generated on a control system for each of the color planes (red, blue, and green).

There are two reasons for reducing the size of the test images.

- Large test images occupy too much disk space on the root directory of the system.
- The test images may not match the result of the tests pixel by pixel even when the hardware is functioning perfectly. This is due to the rounding error of different floating point hardware and the eventual displacements produced by different Firmware revisions (C30 GPSI).

The test results are first reduced 64 times by averaging all 8x8 pixel blocks to a single value and then comparing them with the test images. Each compared pixel is allowed to be within a certain tolerance range to compensate for the variations mentioned above. The tolerance range is small enough to sense functional abnormalities and yet big enough to accommodate expected variations. In case of error, the respective 8x8 pixel block turns black and its location is reported.

C30 Level

The C30 level with test code is linked to the GPSI code. The host posts an unpublished GPSI SunDiag command with a number of parameters (10). The GPSI command interpreter calls a SunDiag subroutine that examines the parameters passed to it and determines which test to call. The test result is passed back to the host through one of the parameters. The host hangs and waits until it receives the completion flag from the test. If the C30 test code dies, the host will time out after a couple of seconds.

Figure 2-12 shows the tests implemented in the `cg12` test. You can select the subtests to be run and the loop counts for each subtest and each board. The default selection for all subtests and loop counts is one.

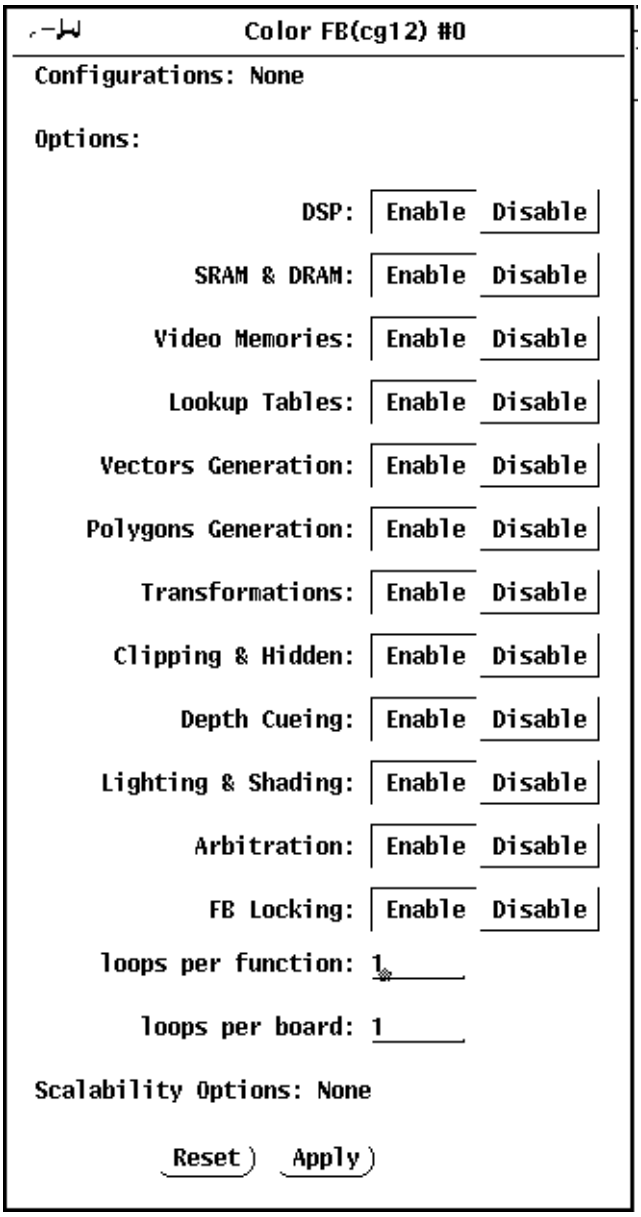


Figure 2-12 `cg12` Options Menu

2.5.2 `cg12` *Options*

DSP

Covers all the internal C30 registers (`int` and `float`), on-chip RAM, and integer and floating point instruction executions (parallel instructions). The RAM test is a nondestructive read/write test with the patterns 0x5a5a5a5a and 0xa5a5a5a5. During the RAM test, all interrupts are disabled. The registers are tested with the patterns 0x55555555 and 0xaaaaaaaa.

SRAM & DRAM

The SRAM test switches on each page of the 4 SRAM pages and executes the DSP test on the selected SRAM page. The DRAM test is the same RAM test run again on the DRAM space, except that it excludes the space where the test code resides. (The test code resides in DRAM.)

Video Memories

Selects and tests all 64x64 Pixrect regions covering the entire video memory planes (including overlay, zbuffer, window ID). The image planes are tested both single and double buffered. All regions are tested with random patterns generated each time the test is called.

Look up Tables

The look up tables in the RAMDAC are tested with a read/write test and a random table. The random table is generated each time the test is called.

Vectors Generation

This test has three parts. First it generates concentric circles with flat shaded vectors in red, green, blue, yellow, and white. If the result is successfully verified (described above), then an object with color shaded vectors is rendered. If the result of this test is also correct, then an object (hexnut) is generated with wide textured lines. The result of each part is then verified as above with test images previously stored.

Polygons Generation

Flat shaded polygons are rendered, and the result is verified with previously stored test images.

Transformations

An object (the spaceship Enterprise) is rendered and transformed in 3D several times to give the impression of moving in space. It leaves a path when it moves away. The path has to match the expected path in the test images previously stored.

Clipping & Hidden

This test has three parts: clipping, hidden surface removal, and picking. In the clipping test, two objects are generated and the view port is clipped so that only parts of them are visible. In the hidden surface removal test, objects ("pacmen") are generated and should overlay each other correctly in the order of the Z component. In the picking test, a pick window is set and triangles and polygons are generated inside and outside. The results of the clipping and hidden surface removal test are verified with test images previously stored. In the picking test, the pick events are counted.

Depth Cueing

An object is rendered with the Depth Cue Attribute turned on. The result is verified with test images previously stored.

Lighting & Shading

A cylinder composed with polygons and triangles is rendered with multiple light sources and Gouraud shading mode. The zbuffer is on. The result is verified with test images previously stored.

Arbitration

Two Solaris processes are running simultaneously. One process renders flat shaded polygons and vectors at random locations on screens into the 24-bit image plane, while the other accesses the overlay plane, the overlay enable plane, and the window ID plane. If the second process has problems with read and write to and from any of the mentioned planes, the test reports the error. Otherwise the test passes. A visual image with three color stripes (RGB) is shown at the end for visual verification of the RAMDAC.

FB Locking

See the "Special Note on Testing Multiple Framebuffers" section in Chapter 1 of the *SunDiag User's Guide* for details.

2.5.3 `cg12` *Command Line Syntax*

`/opt/SUNWdiag/bin/cg12 D=`*device_name* **`s=`***subtest_number* **`F=`***#_of_loops* **`B=`***#_of_loops* **`L`** *standard_arguments*

Arguments	
`D`=*device_name*	The full path name of the device must be specified.
`s`=*subtest_number*	*subtest_number* specifies the number (a binary representation) of the subtest choices to be run. The default is `-1` (all subtests run). More than one subtest choice can be selected by ordering their subtest numbers. Example: `n = 3` specifies that the DSP and SRAM & DRAM subtests are run. The following subtest choices can be specified: `-1` All subtests `1` DSP `2` SRAM and DRAM `4` Video Memories `8` Look Up Tables `16` Vectors Generation `32` Polygons generation `64` Transformations `128` Clipping and Hidden `256` Depth Cueing `512` Lighting and Shading `1024` Arbitration
`F`=*#_of_loops*	Number of loops for each subtest
`B`=*#_of_loops*	Number of loops for each board
`L`	Disables framebuffer locking. See the “Special Note on Testing Multiple Framebuffers” in Chapter 1 of the *SunDiag User’s Guide* for details.

Note – Extra Swap Space Required: 3 MBytes

Note – Make sure to change directories to `/opt/SUNWdiag/bin` before running `cg12` from the command line. `cg12` is hard-wired to look for its data files, `cg12.data` and `cg12.data.gsxr`, in `/opt/SUNWdiag/bin`.

2.5.4 `cg12` *Quick Test Description*

Running this test in quick mode restricts testing to the DSP subtest (subtest flag set to `1`).

2.5.5 `cg12` *Error Messages*

The following errors described below are reported in case of hardware failures:

`Registers Test failed`

One or more DSP registers failed. The DSP chip is flawed.

`On-chip Memory Test failed`

DSP chip problem; the on-chip memory failed the memory test.

`Integer Instructions Test failed`

DSP chip problem; integer instructions were not performed correctly.

`Float Instruction Test failed`

DSP chip problem; float instructions were not performed correctly.

`DRAM error`

The DRAM does not pass the memory test. An unbundled software diagnostics package, the SunDiagnostic Executive, can help you locate the failure.

`SRAM error in page (0-3)`

SRAM page (0-3) does not pass the memory test. An unbundled software diagnostics package, the SunDiagnostic Executive, can help you locate the failure.

`<Plane>: Pixrect error at <address>`

The Video Memory indicated failed the memory test.<*Plane*> can be window ID, 8-bit compatibility, 24-bit color, or Overlay and Overlay Enable.

`error at index <index>, write <pattern>, read <pattern>`

The hardware look-up table is defective at the location index. The written and expected patterns are shown against the actual pattern.

`Error(s) found in (RED, GREEN, BLUE) component(s)`

The generated patterns do not match the expected patterns in their indicated components.

The vectors generation, polygon generation, transformations, clipping and hidden, depth cueing, and lighting & shading tests have the same error message:

`<type> pick test: detected <num> misses <position> of the pick window out of total <total> picking events`

The Picking test has generated an interrupt error. <*type*> = `triangle` or `polygon`, <*num*> = number of pick events missed, <*position*> = `in`, `out` or `on`, and <*total*> = number of pick events generated.

`Pixrect error in <Plane> at <address>`

The access in the indicated plane has error(s) at the indicated address when the drawing engine is drawing graphics at the same time. Possibly a local bus error.

Note – The above error messages are for hardware only. Any other error messages reported are software error messages and will indicate any necessary action.

2.6 *Color Graphics Frame Buffer Test (*`cg14test`*)*

This test checks the cg14 frame buffer card. `cg14test` is specific to the VSIMM (Video SIMM)/SX Memory Controller devices in the SPARCstation 10 SX.

Warnings

Because of possible conflicts between SunDiag cg14 framebuffer tests and OpenWindows applications that use the cg14 framebuffer, the following restrictions apply when running the `cg14test` SunDiag test:

- Do not run any graphic applications other than OpenWindows while the SunDiag software is running framebuffer tests
- Do not run any OpenWindows programs that generate video updates outside or on top of the SunDiag window
- Do not close the SunDiag window to an icon while it is running framebuffer tests
- Make sure to enable the framebuffer locking option from the Options window (see “FB Locking” on page 2-54)

2.6.1 `cg14test` *Test Description*

`cg14test` has 9 distinct test groups:

1. MDI and VBC Chip Control Registers
2. Memory Chips
3. MDI Chip Cursor Registers
4. MDI Chip Clut registers
5. DAC Chip Registers
6. MDI Chip XLU Registers
7. CG14 Display (visual only)
8. MDI Chip Testmode Readback in 8-bit mode
9. Driver IOCTRLs

Hardware Test Groups (test groups 1 - 6)

Testing is done by opening `/dev/fbs/cgfourteenX`, mmap'ing (R/W Shared) the MDI Control Address Space, modifying the target test location (using direct writes to the mmap'ed address space), reading from the mmap'ed address space for verification, and closing the device.

Visual Pattern Test Group (test group 7)

Testing is done by loading a visual pattern of 256 colors, then rotating the pattern around by adjusting CLUT1. This subtest must be verified visually.

Data Propagation Test Group (test group 8)

Testing is done by loading the frame buffer (FB) memory with four neutral data patterns, then setting a target FB pixel with data that will trigger the testmode readback latch. The result is read from the readback register after vertical blanking occurs. Two different trigger patterns used at each FB pixel. All four MDI pixel paths (A - D) are used, and the pixel locations for each trigger are designed to detect gross MDI input data opens or short, VRAM SAM addressing, and VRAM -> SAM transfer addressing.

The screen will show four horizontal bars, which will be either greyscale of colored. These bars will change each time the trigger data is inverted, and as it completes the testing of a raster pattern.

Note – This test will test in 8 bits per pixelmode. If the resolution and VRAM size allows, 32 bits per pixelmode will also be tested automatically.

Driver Test Group (test group 9)

Test all IOCTL calls that have not yet been used to verify proper driver communication to the hardware. Call the driver to perform a hardware update, and then confirm that update was successful by using the complementary driver read, or reading the mmap'ed address space and comparing it against the stimulus.

Notes

1. `cg14test` performs appropriate steps, from this list, before and after each test (if possible), to maintain context and prevent visual confusion:

a. Save register data before overwriting it.

b. Disable video if possible.

c. Do the specific test.

d. Restore the saved register data info.

2. The data used for register testing will be optimized to include all 0's, all 1's, and walking a 1 through each bit under test.

2.6.1.1 MDI and VBC Chip Control Registers (test group 1)

Master Control Register bits 7-0 w/r verify.
Packed Pixel Register bits 3-0 w/r verify.
Master Status Register bits 7-4 r/o verify 0x00 and 0x30 occur.
Horizontal Blank Start Register bits 9-0 w/r verify.
Horizontal Blank Clear Register bits 9-0 w/r verify.
Horizontal Sync Set Register bits 9-0 w/r verify.
Horizontal Sync Clear Register bits 9-0 w/r verify.
Composite Sync Clear Register bits 9-0 w/r verify.
Vertical Blank Start Register bits 11-0 w/r verify.
Vertical Blank Clear Register bits 11-0 w/r verify.
Vertical Sync Set Register bits 11-0 w/r verify.
Vertical Sync Clear Register bits 11-0 w/r verify.
Transfer Cycle Set Register bits 9-0 w/r verify (MDI revision 0 only).
Transfer Cycle Clear Register bits 9-0 w/r verify (MDI revision 0 only).
Fault Status Address Register bits 15-0 w/r verify.
Auto-increment Address Space Register bits 7-0 w/r verify.
Video Base Register bits 23-12 w/r verify.

2.6.1.2 Memory Chips (test group 2)

VRAM Testing

The Data Bus Test will use 18 NTA patterns (Nair, Thatte, and Abraham's testing procedure for RAM) to check for data and address faults. This test will be performed in MDI_CHUNKY_XBGR_MAP access mode only.

The test operates as follows.

- Ascend through the FB memory clearing it to 0s

- NTA pattern test number *x* reads a location to make sure test data *y* is there, and then writes new data *z* to that location. The location ascends through the FB sequentially.

Table 2-1 `cg14test` NTA Testing Patterns

NTA Test Pattern Number = *x*	**Test Data** = *y*	**New Data** = *z*
1.0	0x00000000	0x01010101
1.5	0x01010101	0xffffffff
2.1	0xffffffff	0xf1f1f1f1
2.2	0xf1f1f1f1	0x33333333
3.1	0x33333333	0xf0f0f0f0
3.2	0xf0f0f0f0	0x0f0f0f0f
4.1	0x0f0f0f0f	0x55555555
4.2	0x55555555	0xaaaaaaaa
5.1	0xaaaaaaaa	0x05050505 (1x) 0x88888888 (2x)
5.2	0x88888888	0xf5f5f5f5
6.1	0xf5f5f5f5	0x00000000 (1x) 0x5f5f5f5f (2x)
6.2	0x5f5f5f5f	0x11111111
7.1	0x11111111	0x00000000 (1x) 0xcccccccc (2x)
7.2	0xcccccccc	0xdbdbdbdb
8.1	0xdbdbdbdb	0x6d6d6d6d
8.2	0x6d6d6d6d	0x6b6b6b6b
9.1	0x6b6b6b6b	0x0000000
9.2	0x00000000	-

Memory Retention

VRAM Data Retention checks for gross problems with the VRAM refresh. Since refresh is active during this test, no retention problems should occur unless the refresh is defective.

This test turns off the video, writes 0's to all the VRAM, waits the specified memory_hold time (the default is 5 Seconds), then reads/compares all VRAM data. This process is repeated with data of f's, then the video is restored and the test is complete.

There are two new command line parameters related to this test, **R=***number* and **H=***number*. **R=** allows the user to specify the refresh interval from 128-1023. This is the time between refresh cycles and the system default is 123. **H=** allows the user to specify the retention test hold time in seconds.

Test Write Recovery

A write recovery test will be used in all the EMC mapping modes to write data to 0's followed by immediately reading that data location to see if the VRAM can recover from a write correctly. This is done to all sequential ascending locations. Next, a second independent pass of memory is made with the complementary data of 0xffffffff being written to descending locations of the FB memory buffer.

The EMC mapping access modes are:

a. MDI_CHUNKY_XGBR_MAP
b. MDI_CHUNKY_BGR_MAP
c. MDI_PLANAR_X16_MAP
d. MDI_PLANAR_C16_MAP
e. MDI_PLANAR_X32_MAP
f. MDI_PLANAR_B32_MAP
g. MDI_PLANAR_G32_MAP
h. MDI_PLANAR_R32_MAP

2.6.1.3 MDI Chip Cursor Registers (test group 3)

Cursor Plane 0 Register bits 31-0 w/r verify.
Cursor Plane 1 Register bits 31-0 w/r verify.
Cursor Plane 0 Register bits 31-0 w/r verify. (w/auto increment)
Cursor Plane 1 Register bits 31-0 w/r verify. (w/auto increment)
Cursor Control Register bits 2-0 w/r verify.
Cursor Color Register 1 bits 28-0 w/r verify.
Cursor Color Register 2 bits 28-0 w/r verify.
X-Cursor Location Register bits 11-0 w/r verify.

Y-Cursor Location Register bits 11-0 w/r verify.
Cursor Plane 0 Non-Auto Registers test
Cursor Plane 0 Auto Registers test
Cursor Plane 1 Non-Auto Registers test
Cursor Plane 1 Auto Registers test
Cursor Planes Retry A test
Cursor Planes Retry B test

2.6.1.4 MDI Chip Clut Registers (test group 4)

LUT1 Registers 0-255 bits 31-27 & 23-0 w/r verify.
LUT1 Registers 0-255 bits 31-27 & 23-0 w/r verify. (w/auto increment)
LUT1D Registers 0-255 bits 31-27 & 23-0 w/r verify.
LUT1D Registers 0-255 bits 31-27 & 23-0 w/r verify.(w/auto increment)
LUT2 Registers 0-255 bits 31-27 & 23-0 w/r verify.
LUT2 Registers 0-255 bits 31-27 & 23-0 w/r verify. (w/auto increment)
LUT2D Registers 0-255 bits 31-27 & 23-0 w/r verify.
LUT2D Registers 0-255 bits 31-27 & 23-0 w/r verify.(w/auto increment)
LUT3 Registers 0-255 bits 31-27 & 23-0 w/r verify.
LUT3 Registers 0-255 bits 31-27 & 23-0 w/r verify. (w/auto increment)
LUT3D Registers 0-255 bits 31-27 & 23-0 w/r verify.
LUT3D Registers 0-255 bits 31-27 & 23-0 w/r verify.(w/auto increment)

2.6.1.5 DAC Chip Registers (test group 5)

RAMDAC Registers

Address Register bits 7-0 (0x7 maximum) w/r verify.
Mode Register bits 7-0 (skip bit 5) bits w/r verify.

Control Registers

ID Register bits 7-0 r/o verify data is 0x8C.
Pixel-Mask Register bits 7-0 w/r verify. (skipped if dac rev = 2)
Command2 Register bits 7-0 w/r verify. (skipped if dac rev = 2)
Command3 Register bits 7-0 w/r verify. (skipped if dac rev = 2)

2.6.1.6 MDI Chip Xlut Registers (test group 6)

XLUT Registers 0-255 bits 7-0 w/r verify.
XLUT Registers 0-255 bits 7-0 w/r verify. (w/auto increment)
XLUTD Registers 0-255 bits 7-0 w/r verify.
XLUTD Registers 0-255 bits 7-0 w/r verify. (w/auto increment)

2.6.1.7 CG14 Display (visual only) (test group 7)

This visual displays 256 boxes on the screen (each in a different color), and then shifts the CLUT1 entries giving the visual impression of the pattern mirroring itself from left to right horizontally. The pattern then rotates up, down, followed by mirroring itself horizontally left to right.

2.6.1.8 MDI Chip Testmode Readback [TMRB] (test group 8)

Note – This test always runs in 8 bit mode, but it will also run in 32 bit mode if the Framebuffer size and resolution permit.

Test Mode Readback Register bits 23-0 r/o verify.

2.6.1.9 Driver IOCTRLs (test group 9)

- MDI_GET_CFGINFO check # of cluts, pixel height, pixel width, and pixel mode against h/w.
- FBIOGATTR check real_type, fb_height, fb_width, fb_depth, fb_cmsize, and fb_size against cfginfo values.
- FBIOGTYPE check fb_type, fb_height, fb_width, fb_depth,fb_size, and fb_cmsize against driver defines or cfginfo values.
- FBIOGVIDEO check status returned against h/w.
- FBIOSVIDEO set off, off, on, on, off verifying against h/w.
- FBIOVERTICAL (imbedded in FBIOSVIDEO use!)
- MDI_VRT_CNTL turn off, off, on, on, off the video interrupt enable and verify the h/w agrees.
- MDI_SET_PIXELMODE set different modes and verify against the h/w.
- MDI_SET_PPR set the different modes and verify against the h/w.
- MDI_SET_COUNTERS set HSS, HSC, XCC, HBC, XCS, HBS, CSC, VSS, VSC, VBC, VBS, HCT, and VCT then verify against h/w.
- MDI_GET_GAMMALUT check driver gammalut info against ramdac.

- MDI_SET_GAMMALUT write to driver then check against ramdac.
- MDI_SET_DEGAMMALUT set driver degammalut, then read the driver info and compare it to what was set.
- MDI_GET_DEGAMMALUT checked in conjunction with SET_DEGAMMALUT.
- MDI_GAMMA_CORRECT turn off, off, on, on, off the driver gamma correction and check against the diaginfo status.
- MDI_SET_XLUT set xlut and verify against h/w.
- MDI_GET_XLUT get xlut and verify against h/w.
- MDI_SET_CLUT set clut (1-3 as applicable) and verify against h/w.
- MDI_GET_CLUT get clut (1-3 as applicable) and verify against h/w.
- FBIOPUTCMAP set and verify clut1 matches.
- FBIOGETCMAP verify clut1 matches get.
- FBIOSATTR set emu_type to FBTYPE_MDICOLOR and verify with
- FBIOGATTR check.
- FBIOGCURMAX verify x and y size match driver defines.
- FBIOSCURSOR verify set at 3 locations matches h/w.
- FBIOGCURSOR verify driver knows what set(s) just did.
- FBIOSCURPOS verify set at 3 locations matches h/w.
- FBIOGCURPOS verify driver knows what set(s) just did.
- MDI_SET_CURSOR set then check CCR, XCU, and YCU cursor h/w registers.

2.6.2 `cg14test` *Options*

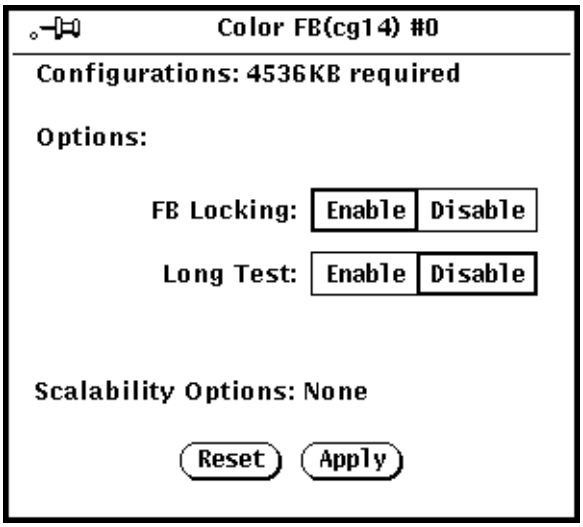


Figure 2-13 `cg14test` Option menu

2.6.2.1 *FB Locking*

See the “Special Note on Testing Multiple Framebuffers” section in Chapter 1 of the *SunDiag User’s Guide* for details.

2.6.2.2 *Long Test*

If Long test is enable, the “color bar” screen(s), in the MDI Testmode Readback test, will check all SAM transfers in clock=0 mode and clock=1 mode. If Long test is disabled, clock=1 will run checks on the first 8 addresses and first SAM transfer only.

2.6.3 `cg14test` *Command Line Syntax*

`/opt/SUNWdiag/bin/cg14test D=`*device* **`L X Passes=`***x* **`I R=`***x* **`H=`***x* **`T`**
standard_arguments

Arguments	
`D=`*device*	Specify the path of the cg14 device file to be tested. For example: **`D=/dev/fbs/`***device_name*
`L`	Disables window system locking option. See the "Special Note on Testing Multiple Framebuffers" in Chapter 1 of the *SunDiag User's Guide* for details. Do not use when device is the window system display.
`X`	Extra long TMRB test. Specify to run longer version of MDI readback. *Note*: The **`q`** option overrides this option.
`Passes=`*x*	*x* is the number of passes to run. The default is 1 pass.
`I`	Enable optional driver ioctl tests for cursor. **`Note:`** Do not the move mouse during the cg14test when this option is run.
`R=`*x*	Specify the time between refresh cycles (128-1023). The default is 123.
`H=`*x*	Specify the retention test hold time in seconds.
`T`	Logic analyzer trigger on TMRB/memory errors.

2.6.4 `cg14test` *Quick Test Description*

If the **`q`** (quick test) option from the standard arguments is specified, `cg14test` runs 1 minute faster, since the MDI readback test is skipped. All of the other test are run when **`q`** is invoked.

2.6.5 `cg14test` *Error Messages*

Error messages in `cg14test` specify the failing test name, including the chip and suspected faulty function. If DEBUG and VERBOSE modes are off, the end of the message contains a prioritized list of user-replaceable FRUs (Field Replaceable Units).

Note that the Visual checker pattern shifting test may not generate any error messages, and must be verified visually.

The following information is appended to the end of any error messages:

```
; *** suspected FRU(s)
```

followed by a sorted list containing any of the following FRUs for the specific failure:

- Operator error
- CG14 video board
- CPU board
- Video cable
- Video monitor
- CG14 device file
- SunOS™
- SunDiag
- Unsupported feature

2.6.5.1 *Error Message List*

Note – In all error messages below, n is a hexadecimal number.

MDI Chip Messages

```
MDI Master Control Register exp=0xnn obs=nn
MDI Packed Pixel Register exp=0xnn obs=nn
MDI H-Blank Start Register exp=0xnnnn obs=nnnn
MDI H-Blank Clear Register exp=0xnnnn obs=nnnn
MDI H-Sync Set Register exp=0xnnnn obs=nnnn
MDI H-Sync Clear Register exp=0xnnnn obs=nnnn
MDI C-Sync Clear Register exp=0xnnnn obs=nnnn
MDI V-Blank Start Register exp=0xnnnn obs=nnnn
MDI V-Blank Clear Register exp=0xnnnn obs=nnnn
MDI V-Blank Set Register exp=0xnnnn obs=nnnn
MDI V-Blank Clear Register exp=0xnnnn obs=nnnn
MDI Xfer-Cycle Set Register exp=0xnnnn obs=nnnn
MDI Xfer-Cycle Clear Register exp=0xnnnn obs=nnnn
```

```
MDI Fault Status Address Register exp=0xnnnn obs=nnnn
MDI Cursor Plane 0 Register 0xnn exp=0xnnnnnnnn
obs=nnnnnnnn
MDI Cursor Plane 1 Register 0xnn exp=0xnnnnnnnn
obs=nnnnnnnn
MDI X-Control Location Register exp=0xnnnn obs=nnnn
MDI Y-Control Location Register exp=0xnnnn obs=nnnn
MDI Control Color Register 1 exp=0xnnnnnnnn obs=nnnnnnnn
MDI Control Color Register 2 exp=0xnnnnnnnn obs=nnnnnnnn
MDI Cursor Control Register exp=0xnn obs=nn
MDI Cursor Plane 0 Non-Auto Register 0xnn exp=0xnnnnnnnn
obs=nnnnnnnn
MDI Cursor Plane 0 Auto Register 0xnn exp=0xnnnnnnnn
obs=nnnnnnnn
MDI Cursor Plane 1 Non-Auto Register 0xnn exp=0xnnnnnnnn
obs=nnnnnnnn
MDI Cursor Plane 1 Auto Register 0xnn exp=0xnnnnnnnn
obs=nnnnnnnn
MDI Cursor Planes Retry test maximum retry limit exceeded
MDI Clut1 Register 0xnn exp=0xnnnnnnnn obs=nnnnnnnn
MDI Clut1 Auto Register 0xnn exp=0xnnnnnnnn obs=nnnnnnnn
MDI Clut1 Diag Register 0xnn exp=0xnnnnnnnn obs=nnnnnnnn
MDI Clut1 Diag Auto Register 0xnn exp=0xnnnnnnnn
obs=nnnnnnnn
MDI Clut2 Register 0xnn exp=0xnnnnnnnn obs=nnnnnnnn
MDI Clut2 Auto Register 0xnn exp=0xnnnnnnnn obs=nnnnnnnn
MDI Clut2 Diag Register 0xnn exp=0xnnnnnnnn obs=nnnnnnnn
MDI Clut2 Diag Auto Register 0xnn exp=0xnnnnnnnn
obs=nnnnnnnn
MDI Clut3 Register 0xnn exp=0xnnnnnnnn obs=nnnnnnnn
MDI Clut3 Auto Register 0xnn exp=0xnnnnnnnn obs=nnnnnnnn
MDI Clut3 Diag Register 0xnn exp=0xnnnnnnnn obs=nnnnnnnn
MDI Clut3 Diag Auto Register 0xnn exp=0xnnnnnnnn
obs=nnnnnnnn
MDI Xlut Register 0xnn exp=0xnn obs=nn
MDI Xlut Auto Register 0xnn exp=0xnn obs=nn
MDI Xlut Diag Register 0xnn exp=0xnn obs=nn
MDI Xlut Diag Auto Register 0xnn exp=0xnn obs=nn
```

PCG Chip Message

```
MDI Pixel Clock Frequency %d is not supported
```

where %d is an integer indicating the unsupported clock frequency.

VBC Chip Messages

```
VBC Video Base Register exp=0xnnnnnnnn obs=nnnnnnnn
VBC Control Register exp=0xnnnnnnnn obs=nnnnnnnn
VBC VCR Register, RRI value, exp=0xnnnnnnnn obs=nnnnnnnn
```

RAMDAC Chip Messages

```
DAC Address Register exp=0xnn obs=nn
DAC Mode Register exp=0xnn obs=nn
DAC Control ID Register exp=0xnn obs=nn
DAC Control Dac-Test Register exp=0xnnn obs=nnn
DAC Control Sync-Test Register exp=0xn obs=n
DAC Control Pixel-Mask Register exp=0xnn obs=nn
DAC Control Command2 Register exp=0xnn obs=nn
DAC Control Command3 Register exp=0xnn obs=nn
DAC CMAP Register red[0xnn] exp=0xnnn obs=nnn
DAC CMAP Register green[0xnn] exp=0xnnn obs=nnn
DAC CMAP Register blue[0xnn] exp=0xnnn obs=nnn
```

VRAM Chip Messages

```
MEM (%s), NTA %s offset= 0xnnnnnnnn exp=0xnnnnnnnn
obs=nnnnnnnn
MEM (%s), WRRD %s offset= 0xnnnnnnnn exp=0xnnnnnnnn
obs=nnnnnnnn
MEM (%s), Data Retention offset=0xnnnnnnnn exp=0xnnnnnnnn
obs=nnnnnnnn
```

where first %s can be one of the following:

```
CHUNKY_XBGR_MAPPED
CHUNKY_BGR_MAPPED
PLANAR_X16_MAPPED
PLANAR_C16_MAPPED
PLANAR_X32_MAPPED
```

PLANAR_B32_MAPPED
PLANAR_G32_MAPPED
PLANAR_R32_MAPPED

and second %s will be a pattern number for NTA (see Table 2-1) and indicate the direction (incremental or decremental) of writes for WRRD.

CG14 Driver IOCTL Messages

`IOCTL(%r) %s`

where:

Table 2-2 `cg14test` Driver IOCTL Error Messages *(1 of 3)*

`%r =`	`%s =`
MDI_CFGINFO	
MDI_DIAGINFO	“call failed, errno=%u” ; “number of cluts, exp 0x%x obs 0x%x” ; “pixel height, exp 0x%x obs 0x%x” ; “pixel width, exp 0x%x obs 0x%x” ; “unknown pixel mode, obs 0x%x” ; “pixel mode, exp 0x%x obs 0x%x
MDI_SET_PIXELMODE	“unable to set %d-bit mode” ;
MDI_SET_PPR	“ppr, exp 0x%x obs 0x%x” ;
MDI_SET_COUNTERS	“hss, exp 0x%x obs 0x%x” ; “hsc, exp 0x%x obs 0x%x” ; “hbc, exp 0x%x obs 0x%x” ; “hbs, exp 0x%x obs 0x%x” ; “vss, exp 0x%x obs 0x%x” ; “vsc, exp 0x%x obs 0x%x” ; “vbc, exp 0x%x obs 0x%x” ; “vbs, exp 0x%x obs 0x%x” ; “csc, exp 0x%x obs 0x%x” ; “xcs, exp 0x%x obs 0x%x” ; “xcc, exp 0x%x obs 0x%x” ;
MDI_SET_GAMMALUT	“red” ; “green” ; “blue” ;
MDI_GET_GAMMALUT	
MDI_GAMMA_CORRECTION	“gamma correction, exp 0x%x obs 0x%x” ;
MDI_SET_CLUT	“clut, offset 0x%x exp 0x%x obs 0x%x” ;
MDI_GET_CLUT	
MDI_SET_XLUT	

Table 2-2 `cg14test` Driver IOCTL Error Messages *(2 of 3)*

%r =	**%s =**
MDI_GET_XLUT	“driver xbuf, offset 0x%x exp 0x%x obs 0x%x” ; “xlut, offset 0x%x exp 0x%x obs 0x%x” ;
FBIOGATTR	“type, exp 0x%x obs 0x%x” ; “height, exp 0x%x obs 0x%x” ; “width, exp 0x%x obs 0x%x” ; “cmap size, exp 0x%x obs 0x%x” ; “fb size, exp 0x%x obs 0x%x” ; “depth, exp 0x%x obs 0x%x” ; “real_type, exp 0x%x obs 0x%x” ;
FBIOGTYPE	
FBIOPUTCMAP	
FBIOGETCMAP	“red, offset 0x%x exp 0x%x obs 0x%x” ; “green, offset 0x%x exp 0x%x obs 0x%x” ; “blue, offset 0x%x exp 0x%x obs 0x%x” ;
FBIOSCURSOR	
FBIOGCURSOR	“set, exp 0x%x obs 0x%x” ; “enable, exp 0x%x obs 0x%x” ; “x hot position, exp 0x%x obs 0x%x” ; “y hot position, exp 0x%x obs 0x%x” ; “cmap index, exp 0x%x obs 0x%x” ; “cmap count, exp 0x%x obs 0x%x” ; “cmap red pointer, exp 0x%x obs 0x%x” ; “cmap green pointer, exp 0x%x obs 0x%x” ; “cmap blue pointer, exp 0x%x obs 0x%x” ; “cursor x size, exp 0x%x obs 0x%x” ; “cursor y size, exp 0x%x obs 0x%x” ; “image pointer, exp 0x%x obs 0x%x” ; “mask pointer, exp 0x%x obs 0x%x” ;
FBIOSCURPOS	
FBIOGCURPOS	
FBIOGCURMAX	“x position, exp 0x%x obs 0x%x” ; “y position, exp 0x%x obs 0x%x” ;
FBIOVERTICAL	
FBIOSVIDEO	

Table 2-2 `cg14test` Driver IOCTL Error Messages *(3 of 3)*

`%r =`	`%s =`
FBIOGVIDEO	“video status, exp 0x%x obs 0x%x” ; “unknown video status, obs 0x%x” ;
MDI_VRT_CNTL	“vertical interrupt enable, exp 0x%x obs 0x%x” ;
MDI_GET_DEGAMMALUT	“degamma index mismatch” ; “degamma count mismatch” ; “degamma lut, offset 0xnn exp 0xnn obs 0xnn” ;
MDI_SET_DEGAMMALUT	
FBIOSATTR	“unable to set emu_type, exp 0x%x obs 0x%x” ;
MDI_SET_CURSOR	“cursor control reg, exp 0xnn obs 0xnn” ; “cursor enable, offset 0xnn exp 0xnnnnnnnn obs 0xnnnnnnnn”

VRAM/MDI Composite Chip Path Message

```
MDI chip TestMode ReadBack, %d-bit %s mode, offset= 0x%x
pixelpipe=%c clock=%e exp=0x%x obs=0x%x
```

where: `%d` is 8 or 32
`%c` is pipe A, B, C, or D,
`%x` is a hexadecimal number.
`%e` is 0 or 1,
`%s` is Greyscale, CLUT1, CLUT2, CLUT3, or Direct

2.7 *Pixel Processor Test (*`sxtest`*)*

This test checks models of SPARCstation 10 machines equipped with an on-board Pixel Processor module. `sxtest` is specific to the VSIMM (Video SIMM)/SX Memory Controller) devices in the SPARCstation 10 SX.

Warnings

Because of possible conflicts between cg14 SunDiag framebuffer tests and OpenWindows applications that use the cg14 framebuffer, the following restrictions apply when running the `sxtest` SunDiag test:

- Do not run any graphic applications other than OpenWindows while the SunDiag software is running framebuffer tests
- Do not run any OpenWindows programs that generate video updates outside or on top of the SunDiag window
- Do not close the SunDiag window to an icon while it is running framebuffer tests
- Make sure to enable the framebuffer locking option from the Options window for the system console cg14 device (see “FB Locking” on page 2-64)
- If `sxtest` is run with its VRAM enabled, then framebuffer locking *must* also be enabled

2.7.1 `sxtest` *Description*

This test locates load error, store error, ALU error, logic error etc. of the Pixel Processor by reading and verifying data from the control registers of the Pixel Processor, virtual memories, or video memories. This test also verifies the integration function of the CG14 frame buffer and its device driver, video memories, and data memories. `sxtest` also writes a test pattern to the frame buffer for visual verification.

`sxtest` is a series of 16 modules, described below.

2.7.2 `sxtest` *Options*

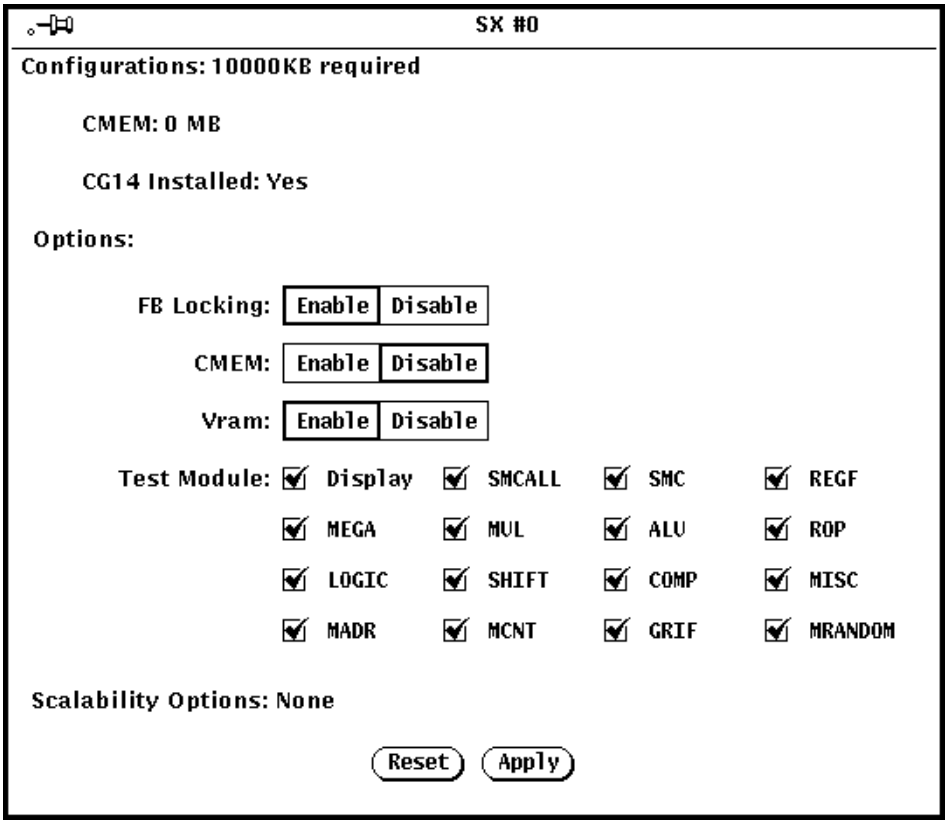


Figure 2-14 `sxtest` Option Menu

FB Locking

See the “Special Note on Testing Multiple Framebuffers” section in Chapter 1 of the *SunDiag User’s Guide* for details. Frame buffer locking is enabled by default on the window server running the OpenWindows software.

CMEM (Contiguous Memory)

Choose either Enable or Disable. You must choose Disable if your system has less then 4 MBytes of contiguous memory available.

VRAM (Video Random Access Memory)

Choose either to Enable or Disable the video random access memory.

Warning – If `sxtest` is run with its VRAM enabled, then framebuffer locking *must* be enabled or the SunDiag software will result in errors.

2.7.3 `sxtest` *Module Descriptions*

2.7.3.1 *Display (Module0)*

Click enable to display visual patterns.

3 subtests call the SPAM library and display pictures to verify the integrity of a subset of the kernel and the SPAM libraries via the SPARCstation 10 SX video system. These routines are ported from the SPAM demonstration programs. All subtests in this module are skipped if the cg14 frame buffer does not exist, or if the VRAM is disabled.

- `rect_test` — The screen is filled with random rectangles. The rectangles are drawn in CHUNKY_XBGR mode if 32-bit mode OpenWindows is running. If not, they are drawn in CHUNKY_C8 mode with the SPAM library routine `sl_rect_fill_32`.
- `shaa` — A picture of shaded lines is drawn in CHUNKY_BGR mode with the SPAM library routines `sl_line_shaa_32`, `sl_span_load_8` and `sl_rect_fill_8`.

Note – The `shaa` test will be skipped if test is running on an 8 bit window.

- `lines` — The screen is filled with lines of various colors. These lines are drawn in CHUNKY_XBGR mode if 32-bit mode OpenWindows is running; If not, they are drawn in CHUNKY_C8 mode with SPAM library routine `sl_line_fill_8`.

2.7.3.2 *SMCALL (Module1)*

Click enable for a brief test of all `sxtest` functionalities.

11 subtests are called from `spam.smcall` to verify the general function of the SMC chip. All subtests have a cg14 version and a non-cg14 version. These subtests will repeat four times, each time with the `IQ FIFO` programmed to a different number of entries (8, 16, 32, and 64).

- `shift_ldst`
- `instr_mix *`
- `arith_ldst`
- `cmp_ldst`
- `select_ldst`
- `interlock_all *`
- `logic_ldst`
- `mult_ldst`
- `rop`
- `scat_ldst`
- `delt_ldst`

---

* These subtests are skipped if the VRAM option is set to disable.

2.7.3.3 SMC (Module2)

Click enable to test `sx` instructions with VRAM page crossing.

17 subtests are called from `spam.smc`. The `smc_pagex1` sub-tests verify `spam_ldla` instructions when effect addresses are crossing memory page. The `smc_rampx` sub-tests verify Ram Page Crossing Error logic with `spam_ldla` instructions. Both DRAM address space and all VRAM Mode address spaces are covered. All subtests have a cg14 version and a non-cg14 version.

Because 8 Mbyte VRAMs and 4 Mbyte VRAMs are assembled with different RAM chips, you must set the *VRAM* option correctly.

- `smc_pagex1_0 *`
- `smc_pagex1_1`
- `smc_pagex1_2`
- `smc_pagex1_3`
- `smc_pagex1_4`
- `smc_pagex1_5`
- `smc_pagex1_6`
- `smc_pagex1_7`

- smc_pagex1_8
- smc_rampx_1_8k
- smc_rampx_2_8k
- smc_rampx_3_8k
- smc_rampx_4_8k
- smc_rampx_5_8k
- smc_rampx_6_8k
- smc_rampx_7_8k
- smc_rampx_8_8k
- smc_rampx_1_16k
- smc_rampx_2_16k
- smc_rampx_3_16k
- smc_rampx_4_16k
- smc_rampx_5_16k
- smc_rampx_6_16k
- smc_rampx_7_16k
- smc_rampx_8_16k
- sp_pagex8_1 *
- sp_pagex9_1 *

These subtests are only tested when there are 4 Mbytes of VRAM on the cg14 board.

These subtests are only tested when there are 8 Mbytes of VRAM on the cg14 board.

* These subtests are skipped if the CMEM option is set to disable.

2.7.3.4 *REGF (Module3)*

Click enable to test the register file pointer logic.

22 subtests are called from `spam.regfile` to verify the register file's logic with assorted SPAM instructions.

- `readpointer1` *
- `readpointer2` *
- `readpointer3` *
- `readpointer4` *
- `writepointer1` *
- `writepointer2` *
- `writepointer3` *
- `writepointer4` *
- `readpointer5` *
- `writepointer5` *
- `rdptr0` †
- `wrptr0` †
- `rdptr1` †
- `wrptr1` †
- `rdptr2` †
- `wrptr2` †
- `rdptr3` †
- `wrptr3` †
- `rdptr4` †
- `wrptr4` †
- `rdptr5` †
- `wrptr5` †

---

* These subtests are skipped if the VRAM option is disabled.

† These subtests are skipped if the CMEM option is disabled.

2.7.3.5 *Megacell (Module4)*

Click enable to test basic ASIC cell elements.

8 subtests verify the megacell functionalities of SMC chip. Megacell is the basic cell for SMC chip. These subtests are converted with test vectors from LSI. All subtests have a cg14 version and a non-cg14 version.

- `16bitadder` — tests 16-bit adder in comp block.
- `alu_adder1` — tests 32-bit adder in comp block.
- `alu_adder2` — does delta operations.
- `alu_comparator1` — tests 32bit comparator in ADGEN.
- `alu_shifter1` — tests shifter in alu block.
- `m16` — tests 16 bit multiplier.
- `registerfile_RR32X32M` — tests registerfile.
- `registerfile_RR8X3207M` — tests write buffer. This subtest is skipped if VRAM option is set to disable.

2.7.3.6 *MUL (Module5)*

Click enable to test the multiplier operations.

8 subtests are called, and each subtest has 2500 randomly generated MUL SPAM macros. Each subtest tests SPAM MUL instruction sets by executing random SPAM MUL macro patterns. For example:

```
spam_dot(S_0,R42,R45,R31,5)
spam_mulr(L_16,R44,R29,R52,1)
spam_mul(S_15,R115,R114,R58,4)
spam_mul(L_16,R89,R110,R81,8)
spam_mulr(S_8,R21,R76,R53,1)
spam_saxpr(S_8,R54,R46,R98,2)
spam_dotr(L_16,R75,R40,R20,5)
spam_dot(L_16,R44,R45,R84,4)
spam_saxp(L_0,R93,R96,R44,8)
spam_mulr(L_0,R86,R56,R56,5)
spam_dotr(L_0,R14,R62,R40,2
spam_saxpr(S_15,R112,R85,R95,7)
```

- sp_mul0
- sp_mul1
- sp_mul2
- sp_mul3
- sp_mul4
- sp_mul5
- sp_mul6
- sp_mul7

2.7.3.7 ALU (Module6)

Click enable to test ALU operations.

5 subtests are called, and each subtest has 2500 randomly generated ALU SPAM macros. Each subtest tests SPAM ALU instruction sets by executing random SPAM ALU macro patterns. For example:

```
spam_subv(R101,R31,R42,1)
spam_subs(R90,R44,R90,14)
spam_subv(R44,R70,R29,14)
spam_sum(R58,R95,R114,9)
spam_adds(R54,R46,R98,10)
spam_addi(R9,51,R68,9)
spam_abs(R76,R28,7)
spam_addv(R80,R59,R93,11)
```

- sp_alu0
- sp_alu1
- sp_alu2
- sp_alu3
- sp_alu4

2.7.3.8 ROP (Module7)

Click enable to test the ROP operations.

5 subtests are called, and each subtest has 2500 randomly generated ROP SPAM macros. Each subtest tests SPAM ROP instruction sets by executing random SPAM ROP macro patterns. For example:

```
spam_selb(R101,R31,R42,1)
spam_ropl(R90,R27,R44,14)
spam_sels(R19,R16,R112,15)
spam_ropm(R47,R29,R96,16)
spam_selb(R52,R43,R29,5)
spam_ropb(R115,R114,R58,7)
spam_selv(R57,R75,R16,2)
spam_ropm(R110,R93,R83,13)
```

- sp_rop0
- sp_rop1
- sp_rop2
- sp_rop3
- sp_rop4

2.7.3.9 *LOGIC (Module8)*

Click enable to test the logical operations.

5 subtests are called, and each subtest has 2500 randomly generated LOGIC SPAM macros. Each subtest tests SPAM LOGIC instruction sets by executing random SPAM LOGIC macro patterns. For example:

```
spam_xors(R101,R31,R42,1)
spam_xori(R90,101,R90,14)
spam_xorv(R30,R19,R95,13)
spam_ands(R108,R16,R125,1)
spam_andv(R115,R114,R58,7)
spam_ors(R46,R89,R8,16)
spam_orv(R57,R75,R16,2)
spam_andi(R9,51,R68,9)
```

- sp_logic0
- sp_logic1
- sp_logic2
- sp_logic3
- sp_logic4

2.7.3.10 SHIFT (Module9)

Click enable to test the shift operations.

5 subtests are called, and each subtest has 2500 randomly generated SHIFT SPAM macros. Each subtest tests SPAM SHIFT instruction sets by executing random SPAM SHIFT macro patterns. For example:

```
spam_sllv(R101,R31,R42,1)
spam_slli(R90,5,R90,14)
spam_srai(R30,19,R95,13)
spam_srli(R108,16,R125,1)
spam_sllv(R52,R43,R29,5)
spam_slfi(R46,25,R8,16)
spam_slfs(R57,R75,R16,2)
spam_srav(R54,R44,R93,8)
spam_srlv(R58,R60,R96,16)
```

- `sp_shift0`
- `sp_shift1`
- `sp_shift2`
- `sp_shift3`
- `sp_shift4`

2.7.3.11 COMP (Module10)

Click enable to test the compare operations.

5 subtests are called, and each subtest has 2500 randomly generated COMP SPAM macros. Each subtest tests SPAM COMP instruction sets by executing random SPAM COMP macro patterns. For example:

```
spam_cmpv_gt(R101,R31,R42,1)
spam_cmps_lt(R90,R44,R90,14)
spam_cmps_eq(R95,R112,R19,12)
spam_cmpv_gt(R44,R43,R29,14)
spam_cmpv_lt(R115,R114,R58,7)
spam_cmps_gt(R46,R89,R8,16)
spam_cmps_eq(R57,R75,R16,2)
spam_cmpv_le(R54,R46,R98,10)
spam_cmpv_eq(R9,R51,R68,9)
spam_cmps_gt(R76,R103,R28,7)
```

```
spam_cmpv_eq(R52,R37,R50,8)
spam_cmpv_ge(R61,R86,R16,12)
```

- sp_comp0
- sp_comp1
- sp_comp2
- sp_comp3
- sp_comp4

2.7.3.12 *MISC (Module11)*

Click enable to test the miscellaneous operations.

5 subtests are called, and each subtest has 2500 randomly generated MISC SPAM macros. Each subtest tests SPAM MISC instruction sets by executing random SPAM MISC macro patterns. For example:

```
spam_scat(R45,-1,R29,1)
spam_gath(R95,-6,R114,9)
spam_delt(R89,R9,R16,16)
spam_plot(R54,R46,R98,10)
spam_plot(R53,R20,R75,16)
spam_scat(R91,-2,R70,9)
spam_gath(R120,-2,R51,15)
spam_delt(R59,R95,R120,1)
```

- sp_misc0
- sp_misc1
- sp_misc2
- sp_misc3
- sp_misc4

2.7.3.13 *MADR (Module12)*

Click enable to test the address lines of `sx`.

8 subtests are called; each subtests verifies 0x100000 SPAM address with `spam_stld` and `spam_ldld` instructions. All address bits and data bits of 4 MBytes of VRAM and 4 MBytes of DRAM will be tested after running through the 8 subtests.

- `0x00000000-0x000fffff`
- `0x00100000-0x001fffff`
- `0x00200000-0x002fffff`
- `0x00300000-0x003fffff`
- `0xfc000000-0xfc0fffff *`
- `0xfc100000-0xfc1fffff *`
- `0xfc200000-0xfc2fffff *`
- `0xfc300000-0xfc3fffff *`

---

* These subtests are skipped is the CMEM option is disabled.

2.7.3.14 *MCNT (Module13)*

Click enable to test the load and store functions with different repeat counts.

12 subtests are called; these test the SPAM store functions by varying address offset and item count.

- `spsd_stba_cnt`
- `spsd_stbd_cnt`
- `spsd_stbds_cnt`
- `spsd_stcd_cnt`
- `spsd_stla_cnt`
- `spsd_stld_cnt`
- `spsd_stlds_cnt`
- `spsd_stpd_cnt`
- `spsd_stqd_cnt`
- `spsd_stsa_cnt`
- `spsd_stsd_cnt`
- `spsd_stsds_cnt`

2.7.3.15 GRIF (Module14)

Click enable to test the graphic interface logic.

36 subtests are called; these test the SPAM graphic interface login with load/store instructions. All subtests are skipped if cg14 doesn't exist.

- spsd_stbd_dram
- spsd_stbd_xbgr
- spsd_stbd_bgr
- spsd_stbd_8x
- spsd_stbd_8c
- spsd_stbd_x32
- spsd_stbd_b32
- spsd_stbd_g32
- spsd_stbd_r32
- spsd_stsd_dram
- spsd_stsd_xbgr
- spsd_stsd_bgr
- spsd_stsd_8x
- spsd_stsd_8c
- spsd_stsd_x32
- spsd_stsd_b32
- spsd_stsd_g32
- spsd_stsd_r32
- spsd_ldbd_dram
- spsd_ldbd_xbgr
- spsd_ldbd_bgr
- spsd_ldbd_8x
- spsd_ldbd_8c
- spsd_ldbd_x32
- spsd_ldbd_b32
- spsd_ldbd_g32
- spsd_ldbd_r32
- spsd_ldsd_dram
- spsd_ldsd_xbgr
- spsd_ldsd_bgr
- spsd_ldsd_8x
- spsd_ldsd_8c
- spsd_ldsd_x32
- spsd_ldsd_b32

- spsd_ldsd_g32
- spsd_ldsd_r32

2.7.3.16 MRANDOM (Module15)

Click enable to test the random test store and load functions with different modes.

5 subtests are called; these test the SPM store/load functions by varying address offset and mode field. For example:

```
spam_stld(X,0x0,R32,32)
spam_ldld(0x60c,R64,32)
spam_stld(MP,0x810,R32,32)
spam_stld(PC,0xa14,R64,32)
spam_stld(SP,0x1020,R64,32)
spam_stld(M,0x1830,R64,32)
spam_stld(C,0x1c38,R64,32)
spam_stld(MC,0x2040,R32,32)
spam_stld(S,0x2244,R64,32)
spam_ldld(0x264c,R64,32)
spam_stld(C,0x2850,R32,32)
spam_stld(MPC,0x2a54,R64,32)
```

- sp_mem1
- sp_mem2
- sp_mem3
- sp_mem4
- sp_mem5

2.7.4 `sxtest` *Command Line Syntax*

`/opt/SUNWdiag/bin/sxtest L mm=`*0xn* **`np=`***#_of_passes* **`nl=`***#_of_local_loops*
`sn=`*subtest#* **`mn=`***module#* **`fp=`***from_pass#* **`tp=`***to_pass#* **`fm=`***from_module#* **`tm=`***to_module#*
`cmem=`*n* **`vram=`***n standard_arguments*

Arguments	
`L`	Disables frame buffer locking. See the "Special Note on Testing Multiple Framebuffers" in Chapter 1 of the *SunDiag User's Guide f*or details. Frame buffer locking is enabled by default on the window server running the OpenWindows software.
`mm=`*0xn*	Selects which modules will be tested in a pass. 0x*n* is a 16-bit hexadecimal number, where bit 0 represents module 0, and bit 15 represents module 15. For example, **`mm=000`**1 selects module vis, **`mm=0010`** selects module 4, and **`mm=ffff`** selects all 16 modules.
`np=`*#_of_passes*	Use this option to run `sxtest` to a particular pass count, starting with zero passes.
`nl=`*#_of_loops*	Use this option to run `sxtest` to a particular loop count, starting with zero loops.
`sn=`*subtest#*	A particular subtest number.
`mn=`*module#*	Use **`mn`** in conjunction with **`sn`** to directly re-invoke the failed test at a specific pass and module number.
`fp=`*from_pass#*	Specifies a beginning pass number.
`tp=`*to_pass#*	Specifies an ending pass number.
`fm=`*from_module#*	Specifies a beginning module number.
`tm=`*to_module#*	Specifies an ending module number. Use these last four arguments to narrow `sxtest` to a specific test scope.
`cmem=`*n*	Choose either to enable or disable the contiguous memory. The possible values are 1 for enable or 0 to disable. **Note:** You must choose disable (0) if your system is equipped with less then 4 Mbytes of contiguous memory.
`vram=`*n*	Choose either to enable or disable the video random access memory. The possible values are 1 to enable or 0 to disable.

2.7.5 `sxtest` *Quick Test Description*

Specifying the **q** (quick test) option from the standard arguments causes `sxtest` to test only the Display and SMCALL subtests.

2.7.6 `sxtest .usertest` *Example*

The following is an example of `sxtest` as used in a `.usertest` file:

```
sx, sxtest, cmem=0, vram=0, nl=2
```

In this example, `sxtest` exercises all 16 modules, each with three different subtests.

2.7.7 `sxtest` *Error Messages*

```
SIGNATURE: lineno: 1917

mem crc16[chunky_xbgr+0x00000000,chunky_xbgr+0x003fffff]:

      o:0x5951 e:0x0356 o^e:0x5a07

mem ccitt[chunky_xbgr+0x00000000,chunky_xbgr+0x003fffff]:

       o:0x5fb9 e:0x1e7b o^e:0x41c2
```

```
loop_pass = 1, i_module =       megacell     4
```

lineno tells which line in the simulation program detected the comparison error. mem and chunky_xbgr indicate the problem in VRAM space.

- o = observed data pattern
- e = expected data pattern
- o^e = exclusive or of the observed data pattern and the expected data pattern

loop_pass indicates the failed loop pass number, and i_module identifies which module failed, including the module number.

If chunky_xbgr is replaced with dram then there is a problem with memory DRAM SIMM.

If mem is replace with reg, then there is a problem with register xx.

```
REGRD_DBL: lineno: 179 o:0x0000000c0xf3455350
e:0xe488a88d0xf3455350
```

Read double registers error.

```
REGRD: lineno: 96 o:0x00000000 e:0x80000000
```

Read register error.

RD_BURST32

32 bytes burst read error.

RD_BURST64

64 bytes burst read error.

RD_BURST128

128 bytes burst read error.

2.8 *S24 Frame Buffer Test (*`tcxtest`*)*

Through a series of protocol, memory, acceleration, and colormap tests, `tcxtest` checks the functionality of the S24 Frame Buffer SBus card.

2.8.1 `tcxtest` *Test Descriptions*

`tcxtest` has four distinct test groups:

1. AFX Protocol Tests (in 8/16/32/64 bit mode)

- WRC

2. Frame Buffer Memory Tests (in 8/16/32/64 bit mode)

- `address`
- `constant`
- `random`

3. Acceleration Tests (both User and Raw modes)

- `blit`
- `stip`

4. Colormap and Cursor Tests

- `cursor`

2.8.1.1 `tcxtest` *Sub-Test Descriptions*

WRC

By performing multiple writes and reads, and then verifying the results, the WRC test exercises the FIFO inside the S24 chip. The WRC test is composed of these three sub-tests: `test_afx_alt_wr`, `test_memafx`, and `test_afx_random`.

If these tests fail, they will print an error message showing the expected and observed data.

- `test_afx_alt_wr`

This test performs 16 writes to alternative pages (for example: WR (Page1), WR (Page2), WR (Page1+off), WR (Page2+off), etc.). It then reads back the data and compares it with the expected results.

This test also writes to the frame buffer space 16 times, followed by a write to a different page in the frame buffer space. The test then reads this data back and verifies it with the expected results.

- `test_memafx`

The CPU in the SWIFT chip has closely coupled interfaces for the DRAM and the AFX bus. This test checks the arbitration between the two accesses.

This test performs a number of alternating writes to the AFX and the CPU memory. After writing to different locations, the test reads and verifies the data. By performing an access across the page boundaries, the test covers both the cached and non-cached accesses.

- `test_afx_random`

After writing to one page in the DRAM memory, the test performs a few random writes/reads to random locations in the AFX space. The test then writes to a different page in the DRAM space, where it performs random accesses again.

This test does not perform any data verification, it just checks to see if any of these random accesses will cause a time out.

constant

The constant pattern test writes a data pattern to the whole memory. This pattern is read back and compared with the expected data. Once the memory fill operation is completed, the test reads the memory back and verifies that the value read is correct.

address

The address pattern test writes a data pattern (which is same as the value of the address) to the whole memory. This pattern is then read back and verified that it is the correct value.

random

The random pattern test writes a random data pattern to the whole memory. This pattern is read back and compared with the expected data. After the memory fill operation is completed, the test reads the memory and verifies the values read are correct.

blit

The blit test has two parts, the raw blit test and the user blit test.

The raw blit test draws a 64x64x24 pixel image at the top left corner of screen. It then blits the image to the screen. The destination images are read back and compared with the original image to verify the raw blit operation executed correctly.

The user blit test draws a 64x64x24 pixel image at the top left corner of screen. It then blits the image to the screen. The destination images are read back and compared with the original image. The user blit test is the same as the raw blit test, except the user blit test uses the user data space for the blit command.

stip

This test performs numerous corner cases for stipple. The test writes to the destination with different data values using a stipple operation. The destination data is read back and verified.

`cursor`

This test performs a data register regression test. It writes a walking 1 pattern to the cursor data registers. The data is then read back and verified with the expected results. The test is repeated using a walking 0 as the data pattern.

2.8.2 `tcxtest` *Options*

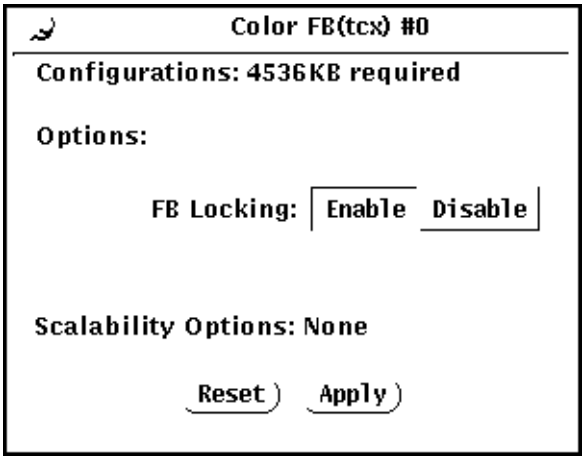


Figure 2-15 `tcxtest` Option Menu

2.8.2.1 *FB Locking*

Click to enable or disable Frame Buffer locking. See the “Special Note on Testing Multiple Framebuffers” section in Chapter 1 of the *SunDiag User’s Guide* for details.

2.8.3 `tcxtest` *Command Line Syntax*

`/opt/SUNWdiag/bin/tcxtest D=`*device_name* **`L X=`***bit_mode* **`T=`***test*
`S=`*[dfb8, dfb24, dfb32]* *standard_arguments*

Arguments	
`D=`*device_name*	Specify the filename of the device to be tested. For example: **`D=tcx0`**
`L`	Disables frame buffer locking. See the "Special Note on Testing Multiple Framebuffers" in Chapter 1 of the *SunDiag User's Guide f*or details. Frame buffer locking is enabled by default on the window server running the OpenWindows software.
`X=`*bit_mode*	Specify the data transfer size. The supported values are: **`8`** byte **`16`** short **`32`** long **`64`** double word
`T=`*test*	Specify a particular test. To specify an individual test, replace *test* with: `a` Address `c` Constant `r` Random `b` Blit `s` Stipple `h` Cursor `w` WRC **Note:** When you select either the Blit or Stipple tests, both the user and raw mode tests will be executed.
`S=`*[dfb8, dfb24, dfb32]*	Specify which framebuffer memory space to use. `dfb8` Dumb framebuffer 8 bit space. Memory is accessed only by bytes. `dfb24` Dumb framebuffer 24 bit space. Memory is accessed only by 24 bit reads and writes. `dfb32` Dumb framebuffer 32 bit space. Memory is accessed by 32 bit reads and writes.

SBus Test Descriptions 3

These chapters describe the tests that are designed to test specific SMCC SBus products. These tests will be displayed under the SBUS Devices section of the SunDiag control panel:

3

3.1 *SBus Printer Card Tests (*`lpvitest` *and* `bpptest`*)*

SBus printer cards have two printer ports: one for SPARCprinters™, and one for any parallel port printer. `lpvitest` verifies the functionality of SPARCprinters, and `bpptest` verifies the functionality of bi-directional parallel port printers.

3.1.1 *Printer Test Hardware and Software Requirements*

The SBus Printer Card and device drivers must be installed in order to run `lpvitest` or `bpptest`. The printer to be tested must be connected to the SPARCprinter or bi–directional parallel port, and powered-up. If both a SPARCprinter and a parallel port printer are hooked up to the SBus card, you can test both simultaneously.

Note – In a SPARCstation 10™, SPARCstation LX™, or SPARCclassic™ system, you can connect a printer directly to the onboard parallel port to run bpptest.

If you are testing the SPARCprinter port, be sure the magnets on the SPARCprinter paper tray are set to the correct paper size. For more information, see the *SPARCprinter Installation and User's Guide*.

If you run `lpvitest` and `bpptest` in conjunction with the `vmem` test, you need at least 10 megabytes of available swap space. If you have less than 10 megabytes of swap space available, add 5 more, regardless of the number of SBus cards installed.

3.1.2 `lpvitest` *Test Description*

This test verifies that your SBus card and SPARCprinter are working properly by attempting to transfer a data pattern from the SBus card to the SPARCprinter and printing the pattern.

There are two indications if the card and printer are functioning properly. First, you will be able to see from the SunDiag Status Window that `lpvitest` has made a successful pass, and second, the pattern transmitted to the printer has printed correctly.

If this test passes successfully, you know that the SBus DMA circuitry, the SPARCprinter, and the device driver are functioning properly.

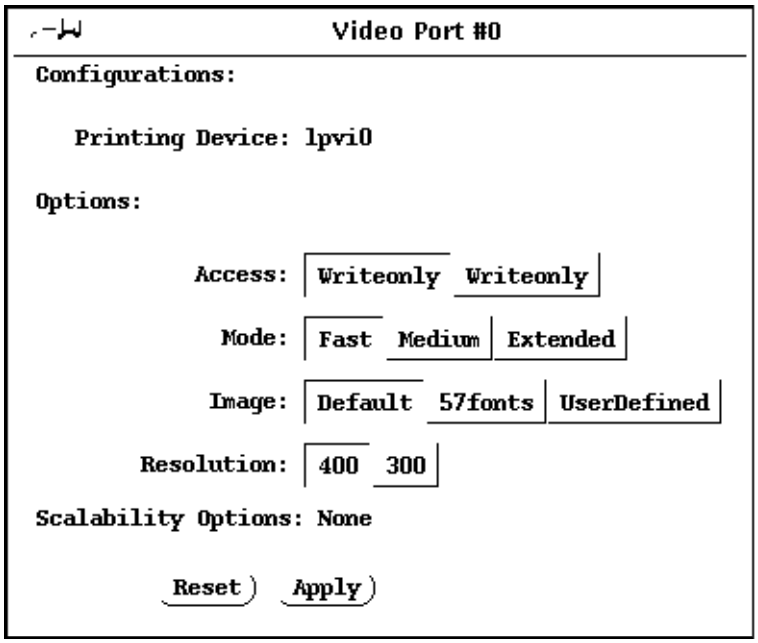


Figure 3-1 `lpvitest` Option Window

3.1.3 `lpvitest` *Options*

Access

Direction of data transfer; this field is informational only. Writeonly is the only option currently available. This indicates that the only data being transferred is going from the SBus printer card to the SPARCprinter. There is no return signal.

Mode

Print image intervals. This option allows you to select how often to print the test image. The default setting is Fast; the choices are:

`Fast`	Prints an image every 10 seconds.
`Medium`	Prints an image every 12 minutes.
`Extended`	Prints an image every 30 minutes.

Image

Which image to print. This option allows you to choose which test image to print. The choices are:

`Default`	A pattern of vertical lines on one page and a checkerboard pattern on another.
`57fonts`	An image of the 57 different fonts that the printer supports.
`Userdefined`	You can use any rasterfile as a test image. Just place the file in the `/opt/SUNWdiag/bin` directory and save it as the filename **`u_image`**.

Resolution

This setting defines the printer resolution of the printed test pattern. The choices are 300 or 400 dots per inch.

Note – Patterns such as the default test pattern will be printed at different sizes at different dpi resolutions. The text in the 57fonts pattern will print in the same size, using the two different resolutions.

3.1.4 `lpvitest` *Command Line Syntax*

`/opt/SUNWdiag/bin/lpvitest D=`*device_name* **`W I=`***image* **`R=`***resolution* **`M=`***mode standard_arguments*

Arguments		
`D`=*device_name*	*device_name* is the name of the device. This should be of the form **`/dev/lpvi`**#, where # is the number of the device.	
`W`	Specifies write–only. This flag is mandatory.	
`I`=*image*	This is the name of the file containing the test image. Possible values are:	
	imagefile	Any file containing user-defined images. Before running SunDiag, copy the raster image file to the `/opt/SUNWdiag/bin` directory and save it with the filename **`u_image`**. The size of the image is adjusted based on the resolution.

Arguments *(Continued)*		
	57fonts	Contains an image of 57 fonts. The size of the image is adjusted based on the resolution.
	default	The default images print on two pages, one image is a pattern of vertical lines and the other is a checkerboard.
R=*resolution*	*resolution* is the resolution of the output in dots per inch (DPI). Possible values are **300** and **400**.	
M=*mode*	*mode* is the print speed mode. This is the rate at which the test image is printed. Possible values are:	
	fast	Prints the test image at 10–second intervals.
	medium	Prints the test image at 12–minute intervals.
	extended	Prints the test image at 30–minute intervals.

Extra Swap Space Required: 1 MB Per Board

3.1.5 `lpvitest` *Quick Test Description*

Running this test in quick mode causes testing to stop after a single data transfer.

3.1.6 `bpptest` *Test Description*

This test verifies that your SBus card and parallel port printer are working properly by attempting to transfer a data pattern from the SBus card to the printer and printing the pattern.

There are two indications if the card and printer are functioning properly. First, you will be able to see from the SunDiag Status Window that `bpptest` has made a successful pass, and second, the pattern transmitted to the printer has printed correctly.

If this test passes successfully, you know that the SBus DMA circuitry, the printer, and the device driver are functioning properly.

Note – Large Postscript files or raster files may require the printer to have 2 Mbytes or more memory. Otherwise, half of the print may appear on one sheet and half on another sheet.

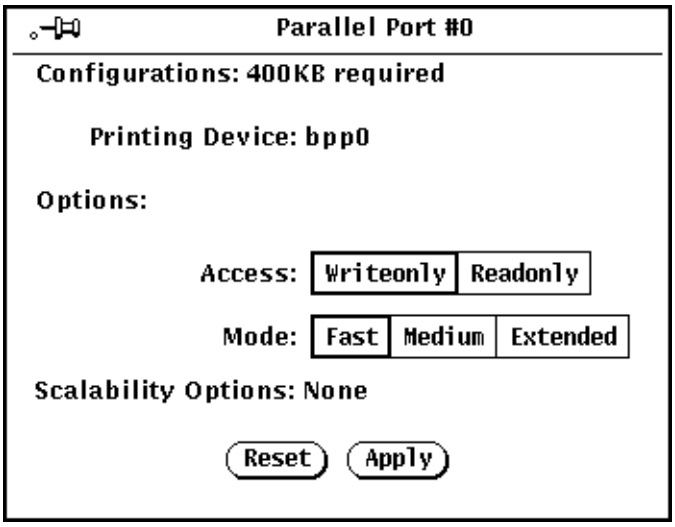


Figure 3-2 `bpptest` Option Window

3.1.7 `bpptest` *Options*

Access

Direction of data transfer; this field is informational only. Writeonly is the only option currently available. This indicates that the only data being transferred is going from the SBus printer card to the SPARCprinter. There is no return signal.

Mode

Print intervals. This option allows you to select how often to print the test image. The default setting is Fast; The choices are

`Fast`	Prints an image every 10 seconds.
`Medium`	Prints an image every 12 minutes.
`Extended`	Prints an image every 30 minutes.

3.1.8 `bpptest` *Command Line Syntax*

`/opt/SUNWdiag/bin/bpptest D=`*device_name* **`W M=`***mode*

Arguments		
`D`=*device_name*	device_name is the name of the device. This should be of the form **`/dev/bpp`***#*, where # is the minor number of the device.	
`W`	Stands for write-only. This flag is mandatory.	
`M`=*mode*	The test image is a continuous printout of the ASCII character set. mode is the print rate of the test image. The image itself is a continuous printout of the ASCII character set. Possible rates are:	
	`fast`	Prints the test image at 10–second intervals.
	`medium`	Prints the test image at 12–minute intervals.
	`extended`	Prints the test image at 30–minute intervals.

3.1.9 `bpptest` *Quick Test Description*

Running `bpptest` in quick mode causes testing to stop after a single data transfer.

3.1.10 *Error Messages*

If the SPARCprinter prints a blank page, verify that the magnet settings on the paper tray are set to the correct paper size. For more information, see the *SPARCprinter Installation and User's Guide*.

If you see a message like this:

```
lpvi0 lpvitest ERROR: Device busy 16
```

you may need to halt currently running print jobs.

3.2 SBus Expansion Subsystem (`xbtest`)

This test verifies the functionality of the Sun SBus Expansion Subsystem hardware and its peripherals.

3.2.1 `xbtest` Test Description

`xbtest` has two mutually exclusive test modes: transparent and nontransparent. Transparent mode tests SBus Expansion Subsystem peripherals, such as SBus cards and disk drives. You should not select transparent mode if the SBus Expansion slots are empty.

Note – Nontransparent mode will test the Expansion Subsystem itself. You should not select nontransparent mode if there are any SBus cards installed in the SBus Expansion slots.

3.2.1.1 Transparent Mode

After you invoke `xbtest`, the system will fork into two processes. The first, the parent process, will wait for the exit of the child process and then exit. The child process will issue an `WAIT_FOR_ERROR_PAK` ioctl call and then enter a sleep mode.

When the device driver receives an error, it wakes up and passes an error packet to the child process. The child process dumps the contents of the error packet, and also exits.

If the system does not receive an error message before the end of the test time period, the trap handler routine calls `clear_for_wait_error`. The procedure will then exit to the parent process. The system will then exit back to the SunDiag system control.

3.2.1.2 Nontransparent Mode

If you do not have an SBus Expansion Subsystem SBus card in any slot of your system, you must add the following line to the `/etc/system` file:

```
set xbox:xbox_no_cards_in_slot0=1
```

After adding this line to `/etc/system`, reboot the machine using the **`-r`** option. You will now be able to run the Nontransparent mode of `xbtest`.

This mode tests the Expansion Subsystem hardware; the subtests are described below. The systems will repeat each of these tests ten times. Time-out checks are included to avoid indefinite hangs. Failure of any test should not result in a system panic, although full recovery is not guaranteed.

Self Diagnostic Test

1. Do a hard reset.
2. Check for expected value from XAC register.
3. Do a DVMA XAC Interrupt Test
   a. Cause a DVMA transfer by asserting DVTE + INTT in control register 1 of XAC.
   b. Wait for interrupt.
   c. Compare error status packet with expected values.
4. Do a DVMA XBC Interrupt Test
   a. Cause a DVMA transfer by asserting DVTE + INTT in control register 1 of XBC.
   b. Wait for interrupt.
   c. Compare error status packet with expected values.

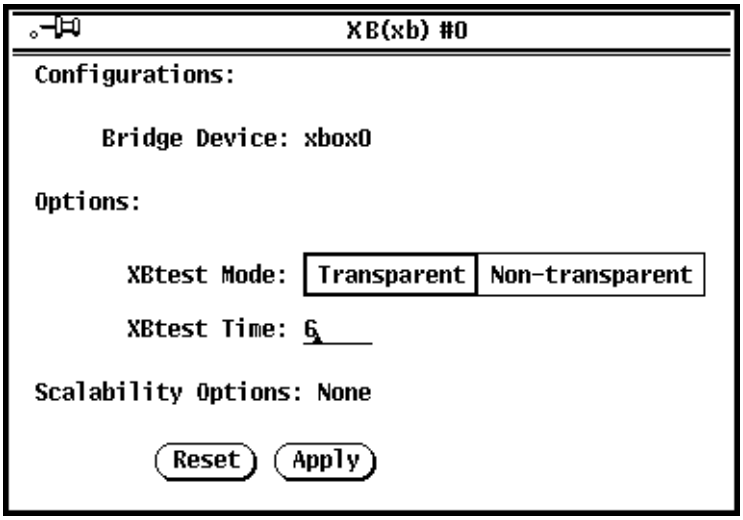


Figure 3-3 `xbtest` Option Window

3.2.2 `xbtest` *Configurations*

The top section of the `xbtest` Options menu displays the Sbus card being tested.

3.2.3 `xbtest` *Options*

XBtest Mode

Change the test mode by clicking SELECT on one of the two exclusive setting boxes. See Section 3.2.1.2, “Nontransparent Mode” before attempting to run Non transparent mode.

XBtest Time

This option specifies how long this `xbtest` waits for an error to be returned. You can change the XBtest Time setting by selecting the text field and typing the number of minutes you want the test to run.

3.2.4 `xbtest` *Command Line Syntax*

When running the `xbtest` from the command line, you must specify the physical pathname of the subsystem unit to be tested. For example, to run `xbtest` on a SPARCstation™ 10 (Sun-4m) system, you would type:

```
# /opt/SUNWdiag/bin/xbtest D=/devices/iommu@f,e0000000/sbus@f,e0001000/SUNW,xbox@1,0:diag M=t
```

`/opt/SUNWdiag/bin/xbtest` **`D=`***device_name* **`WE`** **`T=`***test_time* **`SD`** **`M=`***mode* *standard_arguments*

Arguments	
`D=`*device_name*	*device_name* specifies the physical path name of the subsystem unit to be tested. You need to search the `/devices` tree to find the physical pathname of the subsystem. **Note:** You must include the *device_name* path when running `xbtest` from the command line.
`WE`	Wait for Error. This option directs `xbtest` to run in transparent mode. This option will run `xbtest` continually until an error is returned, or until the time interval specified with the **`T=`***test_time* option has ended.
`T=`*test_time*	Used only with the **`WE`** option. *test_time* specifies how long this test will wait for an error to be returned. Substitute *test_time* with the number of minutes you want `xbtest` to wait. The default is 6 minutes.
`SD`	Self Diagnostic test. This options directs `xbtest` to run in nontransparent mode. See Section 3.2.1.2, “Nontransparent Mode,” on page 3-8, before attempting to run nontransparent mode.
`M=`*mode*	*mode* can be one of the following: **`t`** to run `xbtest` in transparent mode **`nt`** to run `xbtest` in nontransparent mode

3.2.5 `xbtest` *Quick Test Description*

In quick test mode, `xbtest` executes the transparent wait-for-error test, using the default wait time of 6 minutes. If you select the nontransparent self-diagnostic test option, `xbtest` executes the option only once. In normal mode, `xbtest` executes the nontransparent self-diagnostic test three times.

3.2.6 `xbtest .usertest` *File Example*

The following are samples of `.usertest` commands:

```
xbox0 transparent mode, xbtest, S D=(physical path name for xbox0) WE T=1
xbox0 nontransparent mode, xbtest, S D=(physical path name for xbox0) SD
```

3.2.7 `xbtest` *Error Messages*

Whenever errors are returned by the system, SunDiag will display the following error message in the SunDiag console window:

```
error status dirty bit n
```

The system will return a hexadecimal number *n* to indicate the error incurred. The list of errors is in Table 3-1.

Note – The SBus Expansion Subsystem is a Field Replaceable Unit and should only be serviced by Sun service personnel. If you see one of these error messages, call your local Sun service representative.

Table 3-1 SBus Expansion Subsystem error status type bit

Error Message	Mnemonic	Bit Number
expansion sbus read error (err ack)	xrea	c
expansion sbus read error (rsvd ack)	xrra	b
expansion sbus read error (late)	xrle	a
expansion sbus timeout error	xbto	9
write 0 error	wrxr	8
buffer write error (err ack)	bwea	7
buffer write error (rsvd ack)	bwra	6
buffer write error (late error)	bwle	5
cable resend timeout error (dpr0)	crtl	4
cable ack timeout error	cato	3

Table 3-1 SBus Expansion Subsystem error status type bit *(Continued)*

Error Message	Mnemonic	Bit Number
cable parity error	cadp	2
cable serial interrupt parity error	csip	1
child not ready error	cnrd	0

The system may return one or more of the following error messages, where *%d* is the number of times the test failed:

Table 3-2 Miscellaneous `xbtest` Errors

probe xbox device failed
action_on_error ioctl command fail
xbox status ioctl command fail
can not switch to non transparent mode
self diag test failed in `TEST_NON_TRANSPARENT` count %d
self diag test failed in `TEST_XAC_DVMA.int` count %d
self diag test failed in `TEST_XBC_DVMA.int` count %d
XAC_TRANSPARANT ioctl command fail
XAC_NON_TRANSPARANT ioctl command fail
(*device name*): main:fork (*device name*)
XAC_WAIT_FOR_ERROR ioctl command fail
XAC_CLEAR_WAIT_FOR_ERROR ioctl command fail

3.3 *HSI/S Boards Test (*`sunlink`*)*

This test verifies the functionality of the SBus HSI boards. `sunlink` tests the HDCL and SDLC protocol of SBus HSI boards.

3.3.1 `sunlink` *Test Description*

`sunlink` downloads the DCP microcode, initializes the selected channel, and configures the selected channel to the protocol being tested.

Next, `sunlink` opens a datagram socket and tries to modify the socket to accept ioctl communications with the driver, and receive sync mode information from it.

`sunlink` then opens the ports, linking the upper and lower layers with ioctl calls. After initialization, this test checks for activity before attempting to send or receive data. An error message is returned if activity is detected; otherwise the transmit buffer is filled with random data. Random data is used by default; other patterns may be specified. The data is then transmitted. If the transmission succeeds, `sunlink` then receives the returned data and verifies that it is identical to what was sent.

Finally, statistics about the send and receive are gathered from the socket.

A full `sunlink` test takes approximately 8 mins/port. `sunlink` does a brief check of the board ports before the actual test begins. If the port is bad, the test immediately aborts and returns an error message.

Note – This test will not pass unless you install the correct loopback connectors or port to port cables on the ports you are testing. The ports specified for test in the option menu must have loopback connectors attached. See Appendix A for loopback connector part numbers and wiring instructions.

3.3.2 `sunlink` *Configurations*

The Configurations field displays the available ports.

3.3.3 `sunlink` *Options*

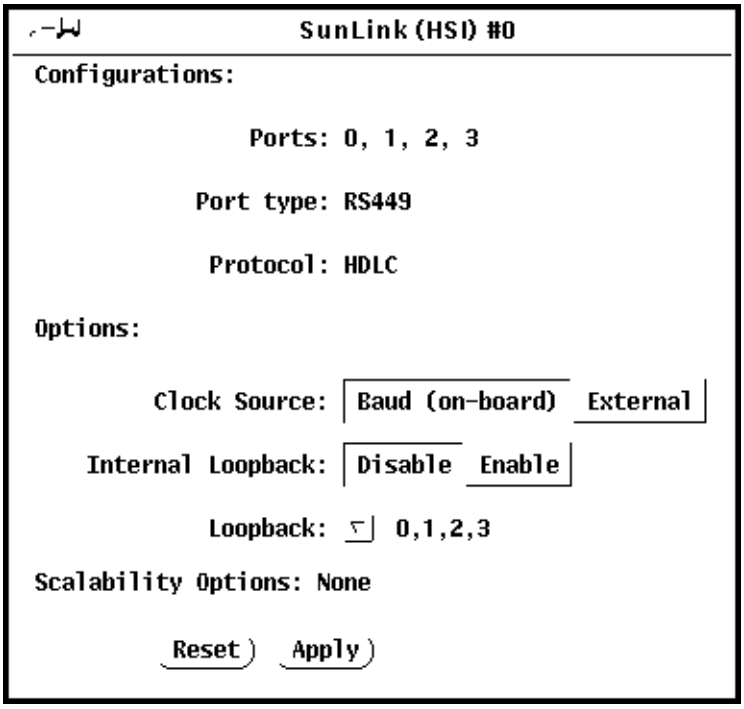


Figure 3-4 `sunlink` Option menu

Clock Source

Clock Source gives you the choice of using either the on-board or external clock for use within `sunlink`. In order to use the external clock option, the transmit, receive, and clock data lines must be physically loopbacked.

Internal Loopback

Click SELECT to enable or disable internal loopback tests. Internal Loopback is only needed when the Loopback setting is not port-to-port, and the clock source is on-board.

Loopback

The Loopback cycle option specifies the loopback type. The choices are: simple single external port loopback, multiple external port loopback, and port-to-port external loopback.

3.3.4 `sunlink` *Command Line Syntax*

`/opt/SUNWdiag/bin/sunlink [`*device_name clock_source port#***`]`** **`p`***data_pattern* **`i`***loopback_type* **`I k st`** *standard_arguments*

Arguments		
device_name	Specifies the device to be tested.	
	`hih`	HDLC and SDLC protocols
clock_source	**`o`**	On Board
	`e`	External
port#	The port number to be tested. For example, **`hiho0. 0`** is the port number.	
`p`*data_pattern*	*data_pattern* is one of the following:	
	`c`	Specifies character (0x55)
	`i`	Incrementing
	`d`	Decrementing
	`r`	Random.
`I`	Enables internal loopback for HSI/S.	
`k`	Load DCP kernel	
`st`	Display sunlink status only	

The following is a typical command line syntax for testing an HSI/S Sbus card:

```
woodberry% /opt/SUNWdiag/bin/sunlink hiho0 I
```

Typing this at the command line will test port 0's internal loopback.

3.3.5 `sunlink` *Quick Test Description*

Running this test in quick mode abbreviates the test procedure. Normally, the internal and external loop counts are set to 100. The internal loop count is set to 1 for the quick test; the external loop count remains at 100. Normally, this test receives time-out checks every 600 seconds; in the quick test, it receives the checks every 4 seconds.

3.3.6 `sunlink` *Loopback Connectors*

When selecting the SCP2 loopback assignments, the “To” port number is connected to the “From” port shown directly above it. For example, in the option menu shown above, ports 0 and 1 are looped, and ports 2 and 3 are looped. If port 3 were not specified, ports 0 and 1 would be linked and port 2 would be looped to itself:

```
From: 0, 2
  To: 1
```

Refer to Appendix A of this manual, and the High Speed Serial Interface hardware manuals for information on loopback connectors.

3.4 *Prestoserve Test (*`pstest`*)*

3.4.1 `pstest` *Test Description*

Prestoserve is an NFS accelerator. It reduces the frequency of disk I/O access by caching the written data blocks in non-volatile memory. Prestoserve then flushes the cached data to disk asynchronously, as necessary. `pstest` verifies Prestoserve's functionality with the following three checks:

Board Battery Check

To ensure proper battery power level, the test runs this check before running the other two checks. If it finds a bad battery, it exits `pstest` immediately with a "fatal error" message.

Board Memory Check

This check maps the entire board memory to a process address space and locks the board to prevent multiple accesses. The test then travels through the mapped address spaces sequentially, doing a "char," "short," and "long" comparison on each. `pstest` executes this check twice.

Board Performance and File I/O Access Check

This check exercises only synchronous read/write access. `pstest` writes data equal to the amount of on-board memory to the memory cache and reads the data back for comparison. The time taken to write that data is measured twice: the first time with Prestoserve disabled, the second time with Prestoserve enabled. The first value is divided by the second to get the performance ratio. If the ration is less than 1.5 on three passes of the test, the Prestoserve board may or may not have a problem.

Since Prestoserve accelerates the `/opt` partition during testing and this partition may be mounted remotely, there may or not be a problem with the Prestoserve board itself; rather, a network performance problem could be the root of the problem. In either case, a warning message is displayed if the performance ratio is less than 1.5.

Caution – To insure that consistent results are obtained, run as many concurrent tests as possible when a Prestoserve product has been installed. Tests are selected from the Option menu. The default is 2; the maximum is 10. However, do not enable `pstest` and `kmem` at the same time. Running these tests together causes SunDiag to incorrectly report errors.

3.4.2 `pstest` *Command Line Syntax*

`/opt/SUNWdiag/bin/pstest l e` *standard_arguments*

Arguments	
`l`	Enable long memory test.
`e`	Enable performance warning, which will display a warning message if the performance ratio is less than 1.5.

3.4.3 `pstest` *Quick Test Description*

Running `pstest` in quick mode changes the test algorithm as follows:

- Board Battery Check - No difference
- Board Memory Check - The check will be executed once instead of twice.
- Board Performance and File I/O Access Check - No difference

3.5 *Serial Parallel Controller Test (*`spiftest`*)*

SunDiag uses the Serial Parallel Controller device driver to access the card components such as the cd-180 and ppc2 chips, and the serial and parallel ports.

3.5.1 `spiftest` *Hardware Requirements*

Before running SunDiag system exerciser, make sure you install the cards to be tested and the device driver. Also, you should reboot your system with the **`boot -r`** command to reconfigure the system and allow the SunDiag kernel to recognize the new driver.

The following minimum hardware configuration is required to successfully run the Internal Test:

- SBus-based SPARC desktop system with an SBus slot.
- Serial Parallel Controller card, installed in one of the SBus slots.

The following hardware is also required to successfully run the other Sundiag Serial Parallel Controller tests:

- Serial Parallel Controller Patch panel (Part No. 540-2007).
- 96-pin loopback plugs (Part No. 370-1366).
- 25-pin serial loopback plugs (Part No. 540-1558).
- RS-232 serial cables (Part No. 530-1685).
- TTY terminal.

Note – `spiftest` must be run in intervention mode.

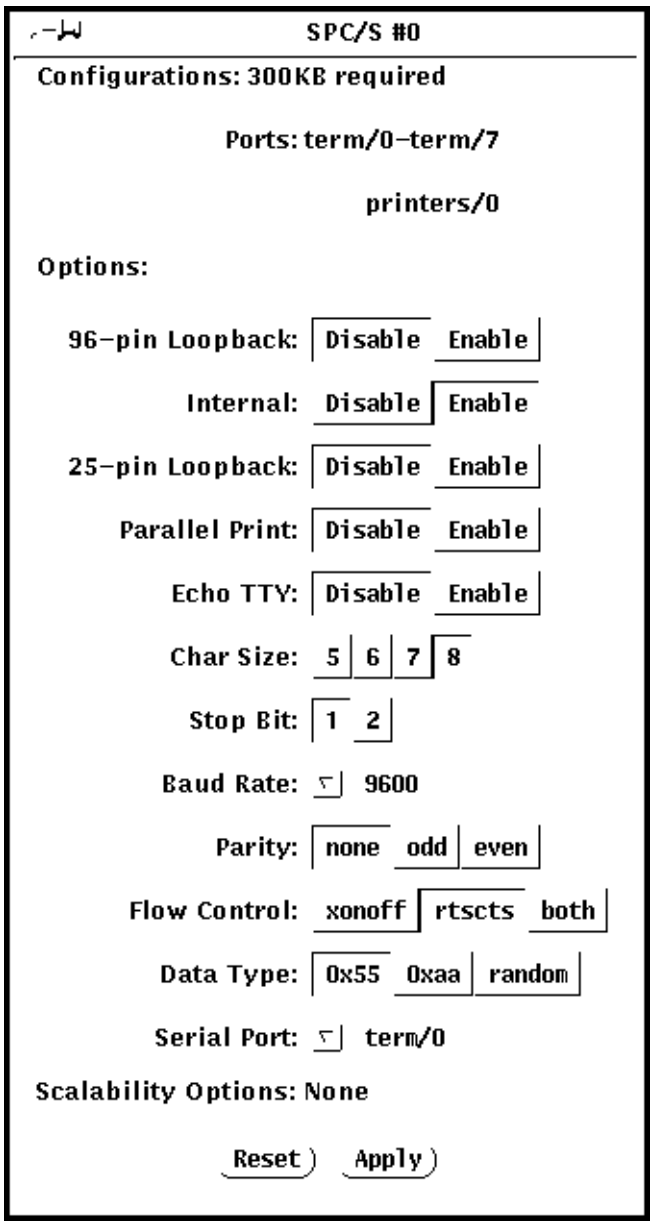


Figure 3-5 `spiftest` Option Window

3.5.2 `spiftest` *Configurations*

The Configurations: section displays the serial ports available for the SPC/S board. The available ports are as follows:

Board Number	Board Device	Serial Ports	Parallel Ports
0	stc0	term/0-7	printers/0
1	stc1	term/8-15	printers/1
2	stc2	term/16-23	printers/2
3	stc3	term/24-31	printers/3
4	stc4	term/32-39	printers/4
5	stc5	term/40-47	printers/5
6	stc6	term/48-55	printers/6
7	stc7	term/56-63	printers/7

3.5.3 `spiftest` *Options*

96-pin Loopback (LB)

This test provides data transmission, full-modem loopback, and parallel port loopback testing. You *must* attach a 96-pin loopback plug to the card under test before running this test (See Appendix A).

Internal Test

This test performs a quick internal check of the Serial Parallel Controller card(s) installed in SBus slots. You do not need to attach anything to the card(s) to perform this test.

25-pin Loopback (LB)

This test provides full-duplex transmission and full-modem loopback testing of the serial port selected in the Serial Port section of this menu. You *must* attach a 25-pin Loopback plug to the serial port on the patch panel that is being tested (See Appendix A). This test cannot be run concurrently with the Echo-TTY option enabled.

Parallel Printer

This test sends the entire ASCII character set to a parallel printer. You must attach a parallel printer to the parallel port on the Serial Parallel Controller patch panel. Observe the printer output to validate the test.

Echo-TTY

This test checks the proper operation of the serial port selected in the Serial Port section of this menu by echoing characters typed on a TTY terminal keyboard to the TTY terminal screen. Type anything on your TTY keyboard, and the characters you type should show up on the TTY screen. If you do not type anything, this test will eventually time-out. This test is terminated by pressing Control-C. After a short delay, the Status Window updates the Pass Count. This test cannot be run concurrently with the 25-pin Loopback subtest.

Char Size

Character length; choose 5, 6, 7, or 8 characters.

Stop Bit

Specifies the number of stop bits; choose 1 or 2 bits.

Baud Rate

Specifies the baud rate; choose 110, 300, 600, 1200, 2400, 4800, 9600, 19200, or 38400 baud.

Note – The baud rate of 38400 can only be used if one port is tested at a time and the Internal Test is disabled.

Parity

Specifies the selectable parity; choose `none`, `odd` or `even`.

Flow Control

Specifies the selectable flow control; can be `XOnOff`, `rtscts`, or `both`.

Data Type

Specifies the selectable data type pattern; can be 0x55555555, 0xaaaaaaaa, or random. 0x55555555 and 0xaaaaaaaa are abbreviated “0x55” and “0xaa” on the option menu.

Serial Port

Specifies the serial port to be tested. The available ports are listed in the Configurations section at the top of the spiftest options menu.

You can also change the test options by modifying the `/opt/SUNWdiag/bin/.usertest` file. See Section 1.8, “Adding Your Own Tests in `.usertest`” in Chapter 1 of the *SunDiag User’s Guide.*

3.5.4 `spiftest` *Command Line Syntax*

`/opt/SUNWdiag/bin/spiftest` **`D=`***device_name* **`T=`***subtest_number* **`B=`***baud_rate* **`C=`***character_length* **`S=`***#of_stop_bits* **`P=`***parity* **`F=`***flow_control* **`I=`***loopback_pattern standard_arguments*

Arguments

`D`=*device_name* — *device_name* is the device to be tested. There is no default; you must type a device name. It can be either a board (`sb1-8`) or an individual port (`term/0 - term/63`):

`sb1`	The 8 serial ports in the first card
`sb2`	The 8 serial ports in the second card
`sb3`	The 8 serial ports in the third card
`sb4`	The 8 serial ports in the fourth card
`sb5`	The 8 serial ports in the fifth card
`sb6`	The 8 serial ports in the sixth card
`sb7`	The 8 serial ports in the seventh card
`sb8`	The 8 serial ports in the eighth card
	or
`/dev/term/`*mm*	

Where *mm* is 0 - 63 (any of the serial ports in SBus card slots)

Arguments		
T=*subtest_number*	*subtest_number* is one of the subtest options below. These are binary representations of the test values. You can enter one value, or the sum of any combination. There is no default; you must type a value:	
	1	Internal loopback test
	2	Printer test Note: Requires parallel printer if run alone.
	4	96-pin loopback test
	8	DB-25 loopback test
	16	Echo-TTY test
	Here are two examples of subtest combinations:	
	10	Printer test with the DB-25 loopback test
	18	Printer test with the Echo-TTY test
B=*baud_rate*	*baud_rate* is one of 110, 300, 600, 1200, 2400, 4800, 9600, 19200, or 38400. The default is 9600. To use the 38400 rate, only one port at a time can be tested, and the Internal Test must be disabled.	
C=*character_length*	*character_length* specifies a character length between 5 and 8. The default is 8 characters.	
S=*#of_stop_bits*	*#of_stop_bits* is the number of stop bits. The default is 1.	
	1	1 stop bit
	2	for two stop bits
P=*parity*	Parity. The default is **none**; the options are:	
	none	
	odd	
	even	
F=*flow_control*	Flow Control Protocol. The default is **rtscts**.	
	xonoff	Xon/Xoff
	rtscts	hardware flow control
	both	xonoff and rtscts

Arguments		
I=*loopback_pattern*	*loopback_pattern* is the loopback test data pattern. The default is **a**.	
	5	an 0x55555555 pattern
	a	an 0xaaaaaaaa pattern
	r	a random pattern

3.5.5 `spiftest` *Quick Test Option*

Running this test in quick mode restricts testing to the Internal Loopback Test. The internal loopback test performs a quick internal check of the Serial Parallel Controller card(s) installed in SBus slots. You do not need to attach anything to the card(s) to perform the quick test.

3.5.6 `spiftest` *Error Messages*

These error messages are generated when the Sundiag Serial Parallel Controller discovers fatal errors. The error descriptions below identify possible causes for the card or test failure, and identify the Field Replaceable Unit (FRU), if possible. The three FRUs affected are: the Serial Parallel Controller card, the 96-pin shielded cable, and the Patch panel.

This section also provides suggestions if an error is not hardware related.

`No SPC/S card found or device driver not installed`

An incorrect slot number was specified, or a non-SPC/S card was found, or the device driver was not installed.

`Ioctl STC_DCONTROL(STC_REGIOW-COR2) error on` <*device name*>, or

`Ioctl STC_DCONTROL(STC_REGIOR-CCR) error on` <*device name*>, or

`Ioctl STC_DCONTROL(STC_REGIOW-CCR) error on` <*device name*>, or

`Ioctl STC_DCONTROL(STC_PPCREGW-PDATA) error on` <*device name*>, or

`Ioctl STC_DCONTROL(STC_PPCREGR-PDATA) error on` <*device name*>, or

`Ioctl TCSETS failed on` <*device name*>, or

`Ioctl TIOCMGET error on` <*device name*>, or

`Ioctl TIOCMSET error on` <*device name*>, or

`Ioctl STC_GPPC error on device` <*device name*>, or

`Ioctl STC_SDEFAULTS error on` <*device name*>, or

`Ioctl STC_DCONTROL error on` <*device name*>

The device driver was not installed correctly, or the card is not working (see system console for messages). You may need to reinstall the device driver.

`Expected DSR set, observed clear`
`Parallel port loopback test failed on` <*device name*>

The modem lines DSR are not stable, or the parallel port is not sending the correct data, or the 96-pin loopback plug either is wired incorrectly or not making proper contact.

`Expected` <*pattern*>
`Observed` <*pattern*>
`Modem loopback test failed on` <*device name*>

The modem lines are not stable. If the modem loopback test passed on the 96-pin loopback test and failed on the 25-pin loopback test, then it is possible that either the patch panel or the cable is not functional. Also, make sure the loopback plug is making proper contact.

`Can't enable SP-96 when DB-25, Print or echo-tty test option is selected!`

The 96-pin loopback test was selected while an incompatible option (25-pin Loopback, Parallel Print, or Echo-TTY) was also chosen.

`Can't enable echo-tty when DB-25 or SP-96 test option is selected!`

Echo-TTY test was selected while an incompatible option (25-pin or 96-pin loopback tests) was also chosen.

`Can't enable Parallel Print when SP-96 test option is selected`

The Parallel Print test was enabled while an incompatible option (25-pin Loopback test) was also chosen.

`Can't open <`*device name*`> with file descriptor <number>`

This is an internal error. Contact Sun for technical assistance.

`Device <`*device name*`> already open`

Two tests are trying to access the same device at the same time or the device is already busy running another process.

`Open error on <`*device name*`>, device driver may not be installed properly`

The device driver is not installed, or not installed properly. When the SPC/S board is removed from a system, the device driver is unloaded at the next boot. If the board is reinstalled, the device driver must then be reloaded with the `/usr/sys/unbundled/stc_config` installation script. Otherwise, Sundiag will not recognize the board, or display `spiftest` as an option, and will return this error message.

`Read error on <`*device name*`>`, or

`Write error on <`*device name*`>`

Another application is currently using this device or there is a board error.

`Paper out error on device <`*device name*`>,`

The parallel printer may be out of paper, or there is no printer connected. Check the printer status.

`Off-line error on device <`*device name*`>`, or
`Busy error on device <`*device name*`>`, or
`Error on device <`*device name*`>`

The parallel printer may be off-line, busy printing data, or some other error condition exists. Check the printer status.

`Expected <number> bytes, observed <number> bytes`
`Internal test failed on <device name>`

Data transmission failed because you're running the internal loopback with another test in `.usertest` mode, or there is a hardware error. The Serial Parallel Controller card must be replaced.

`Expected = <pattern>, observed = <pattern>`
`Internal test failed on <device name>`

Data transmission failed, the card must be replaced.

`Expected = <pattern>, observed = <pattern>`
`Data loopback failed on <device name>`

Data transmission failed because you are running the internal loopback with another test, or the loopback connector is not making proper contact, or the card failed. Re-insert the connector and make sure it makes proper contact. If the test still fails and if you are running the 25-pin loopback test, you can try to run the 96-pin loopback test to isolate the problem. If both tests fail for the same port, then it is possible that the card is not functional. If the 96-pin test passes, and the 25-pin doesn't, try the test with a different cable, loopback plug, or patch panel.

`Timeout error on <device name>`

The system is heavily loaded or has a large amount of swap space configured. Try running `spiftest` alone.

`No loopback plug found on <device name>`

`Timeout error`, or
`tty terminal is not connected to <device name>`

No terminal is connected, or the Echo-TTY test has not been terminated with a Control-C.

3.6 *ZX Graphics Accelerator Test (*`leotest`*)*

SunDiag verifies the Sun Microsystems ZX Graphics Accelerator with a sequence of subtests. All tests are nondestructive and maintain the system integrity during and after the tests are run.

Caution – Do not run any other application that uses the ZX accelerator port while running `leotest`. This combination will cause SunDiag to return incorrect errors.

Note – `leotest` requires approximately 2.0 Mbytes of disk space in the `/tmp` directory to extract its working files. If this space is not available, the diagnostic will fail and report warning and error messages indicating lack of disk space.

By default, SunDiag runs all of the available tests, except the Stereo test. See the Test Descriptions section below.

Note – To avoid excessive test cycle times when testing the ZX Graphics Accelerator, follow these instructions:

1. Enable Single Pass on the SunDiag Options menu.
2. Enable Verbose on the SunDiag Options menu.
3. Do not select any other diagnostic tests.

Following these procedures will ensure that `leotest` will run once, report status as each test routine executes, and then exit.

Note – Disable all screen savers before testing any graphics device. Type **`xset s off`** at a UNIX prompt to disable the Solaris screen saver.

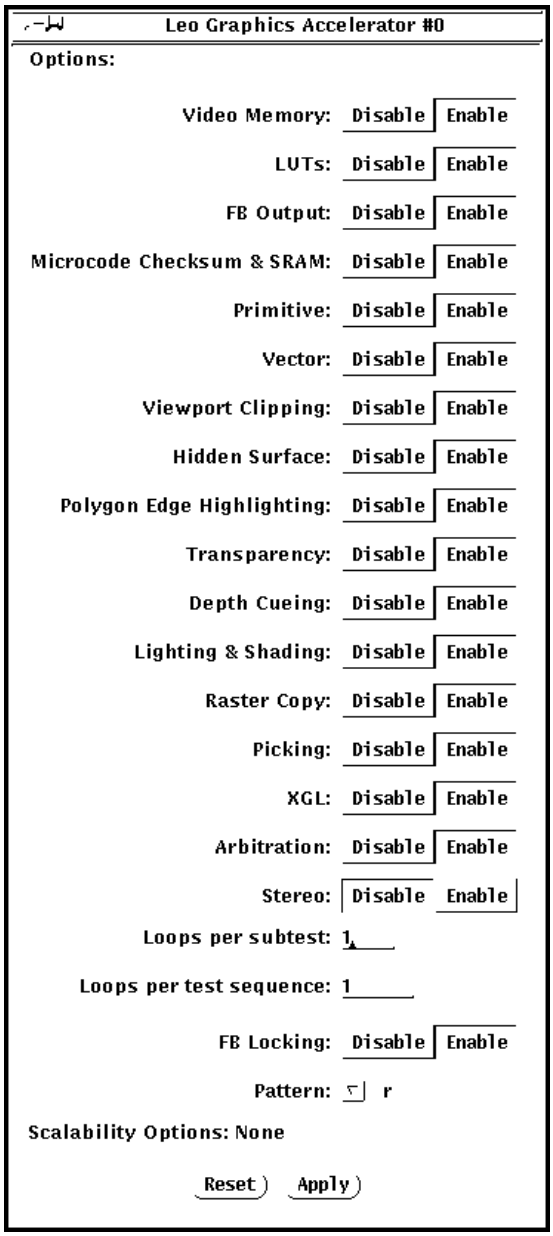


Figure 3-6 `leotest` Option Window

3.6.1 `leotest` *Test Description*

`leotest` is divided into two subtest categories: Direct Port Tests, and Accelerator Port Tests.

3.6.1.1 *Direct Port Tests*

The direct ports tests check the non-accelerated portion of the ZX using the following subtests.

Video Memory

The video memory array subtest selects and tests 64 by 64 pixel regions covering all video memory planes, including the 24-bit double-buffered image plane, 8-bit overlay plane, 24-bit depth (Z buffer) plane, and10-bit WID plane. If the subtest detects an error, SunDiag reports the defective plane and location.

LUTs

This subtest performs a non-destructive read-write test on the frame buffer color look up tables and the window ID look up tables. After the test is done, the previous contents of these tables will be restored. If this subtest detects a failure, SunDiag reports the location of the failure.

At the beginning of this subtest, red, green, and blue stripes display for visual verification of the digital-to-analog converters (DACs).

Frame Buffer Output

The Frame Buffer Output subtest creates various windows in the Window ID plane then sets up the look up tables (LUTs) associated with these windows. This subtest then writes random values to the video memory of these windows. Next, the test verifies the image RGB data of each pixel by triggering the trap registers, reading and generating RGB checksums from these values, and comparing these checksums with known values. This is also a visual cursor test as the cursor is being displayed for each pixel that is under test.

Caution – Moving the mouse cursor during the test will prolong the test, and may cause failures. For best results, remove the mouse from the mouse pad during this test.

3.6.1.2 *Accelerator Port Tests*

The accelerator port test consists of a sequence of subtests which are designed to ensure the ZX Graphics Accelerator integrity at the system level. They take as their inputs accelerator port transaction files. These files contain graphic data which are passed to the ZX Accelerator port in groups of 32-bit words called “packets.” These packets contain dots, vectors, triangles, and pass-through commands, and are generated in either immediate (programmed I/O) or DMA mode. The ZX SunDiag queries the system software for DMA capability, and if applicable, will render objects in DMA mode for some subtests. For more information on ZX hardware, please refer to the Leo Hardware Reference Manual.

For verification, after the image is rendered to the frame buffer, each subtest reads the RGB image data from the frame buffer and compares the data against known good images. To save disk space, the good image data are stored in a reduced size (64 times smaller that the normal size), and are stored in the Sun raster file format. The files are stored in the `leotest.data` file in `/opt/SUNWdiag/bin`.

Note – These subtests verify a frame buffer region of 1152 by 900 pixels, regardless of the size of the monitor attached to the system.

Microcode SRAM Checksum and Read/Write Selftest

This subtest sends a diagnostic package to the microcode of the Floating Point Transform engine to instruct it to reset the accelerator port, run the SRAM selftest, and restart the engine. Then it will verify each SRAM of all four Leo Float chips in the Floating Point Transform engine to make sure they have the same checksum. If not, it will report error using the information passing to it from the microcode. It also performs non-destructive read/write tests on each SRAM.

Primitive

This subtest renders primitives such as dots and lines as well as triangles with different color and shading at each vertex.

Vector

This subtest renders fairly large vector objects with aliased and anti-aliased vectors. This subtest will be rendered in DMA mode, when applicable.

Viewport Clipping

This subtest renders and clips an object around and in front on the screen.

Hidden Surface

This subtest renders objects with the Z-buffer-compare attribute turned on.

Polygon Edge Highlighting

This subtest renders an object with the polygon edge attribute turned on. This subtest will be rendered in DMA mode, when applicable.

Transparency

This subtest renders a scene with two transparency modes (stand-alone and alpha blend) in various degrees. This results in a two-pass transparency of the objects in the scene. This subtest will be rendered in DMA mode, when applicable.

Depth-Cueing

This subtest renders an object with the depth-cueing attribute turned on.

Lighting and Shading

This subtest renders an object with multiple light sources and Gouraud shading for front and back surfaces. This subtest will be rendered in DMA mode, when applicable.

Raster Copy

This subtest renders 32-bit image and various subregions of it, and zoom in on a subregion, using the microcode raster data copy command.

Picking

This subtest has two parts: a pick detect test and a pick echo test.

XGL

The transaction file for this subtest was generated from an XGL program to ensure that the XGL registers are exercised.

Arbitration

This subtest continually renders an object into the accelerator port while a second process performs a read-write test to the WID planes from the direct port on the Frame Buffer. This subtest simulates conditions in the real world, where rendering processes and windows operation run concurrently. This subtest will be rendered in DMA mode, when applicable.

Stereo (Interactive)

This subtest displays text information in stereo mode. The user verifies proper operation by looking at the screen with stereo glasses and following the instructions being displayed.

3.6.1.3 ZX SunDiag Options

Loops per subtest

Type the number of times each test should be run before going on to the next test. The default is 1 loop.

Loops per test sequence

Type the number of times the entire test sequence should be run. Each sequence pass will register one in the SunDiag Pass Count window. The default is 1 loop.

FB Locking

See the “Special Note on Testing Multiple Framebuffers” section in Chapter 1 of the S*unDiag 4.4 User’s Guide* for details.

Pattern

Press MENU to select a data pattern for use in the direct port tests. The default is “r” for random patterns. You may select a pattern of 0s, 3s, 5s, or 9s. For example, if you choose a pattern of 3s, the value 0x33333333 will be written to and read from the frame buffer.

3.6.2 `leotest` *Command Line Syntax*

`/opt/SUNWdiag/bin/leotest` **D=***device_name* **S=***subtest_number* **F=***#_of_subtest_loops* **B=***#_of_test_loops* **L** *standard_arguments*

Arguments	
D=*devicename*	*device_name* is the full path name of the device under test. The default is `/dev/fbs/leo0`.
S=*subtest_number*	*subtest_number* is the test number of the subtest to be run. Select from the subtests below. You can run multiple subtests by adding the subtest numbers. For example, `n=0x3` runs both test 1 and test 2; `n=0x180` runs both test 0x080 and test 0x0100. Note that you do not need the leading zeros. To run all tests, enter **`n=0xFFFF`**. **`0x 000 001`** Direct port—video memory **`0x 000 002`** Direct port—CLUTs and WID LUTs **`0x 000 004`** Direct port--Frame buffer output section **`0x 000 008`** Direct port--SRAM checksum & Read/Write **`0x 000 010`** Accelerator port —primitives **`0x 000 020`** Accelerator port—vectors **`0x 000 040`** Accelerator port—clipping **`0x 000 080`** Accelerator port—z-buffer **`0x 000 100`** Accelerator port—polygon edge **`0x 000 200`** Accelerator port—transparency **`0x 000 400`** Accelerator port—depth cueing **`0x 000 800`** Accelerator port—lighting & shading **`0x 001 000`** Accelerator port—raster copy **`0x 002 000`** Accelerator port—picking **`0x 004 000`** Accelerator port—XGL **`0x 008 000`** Accelerator port—arbitration **`0x 010 000`** Accelerator port—stereo (interactive)
F=*#_of_subtest_loops*	*#_of_subtest_loops* is the number of loops for each subtest. The default is 1 (one loop)

Arguments	*(Continued)*
B=*#_of_test_loops*	*#_of_test_loops* is the number of loops of each test sequence. The default is 1 (one loop).
L	Disables framebuffer locking. See the "Special Note on Testing Multiple Framebuffers" in Chapter 1 of the *SunDiag 4.4 User's Guide* for details.
P=*pattern_number*	Select a pattern number for use with the direct ports tests. The default is **r**, for random patterns. You may also choose **0**, **3**, **5**, or **9**.

3.6.3 `leotest` *Quick Test Description*

Running this test in quick mode does not change the test procedure.

3.6.4 `leotest` *Command Line Examples*

Here are three examples illustrating how to run `leotest` from a command line. Make sure to change the directory to `/opt/SUNWdiag/bin` before running `leotest` from the command line. `leotest` is hard-wired to look for its data file, `leotest.data`, in `/opt/SUNWdiag/bin`.

1. A Simple accelerator port test, "primitive" single pass:

```
machine# cd /opt/SUNWdiag/bin
machine# leotest S=0x10
```

2. All direct port tests, five loops of sequence:

```
machine# cd /opt/SUNWdiag/bin
machine# leotest S=0x7 B=0x5
```

3. All subtests (except the interactive tests), two loops of each subtest, four loops of each test sequence:

```
machine# cd /opt/SUNWdiag/bin
machine# leotest S=0xFFFF F=2 B=4
```

3.6.5 `leotest` *Error Messages*

The ZX SunDiag error messages are described below. The error messages are listed in alphabetical order.

`Arbitration test failed.`

The arbitration test fails, and the cause is given in the following message.

`Background process wouldn't die. System error.`

A software error. You may have to re-boot the SPARCstation.

`Busy wait exceeded %d loops. Error in the Floating Point Transform section. Re-run Leoconfig.`

A timeout error condition. Possibly there is a problem with the Floating Point Transform section, and the leoconfig software (located in `/etc/opt/SUNWleo/bin/leoconfig`) should be executed again. Check the man pages on leoconfig for more details.

`[Plane group name] Byte Access Mode error at x=%d y=%d, bank=#, expected=0x#x, observed=0x#n, XOR=0x#x.`

The direct port video memory test has found an error at pixel (x,y) in the named plane group. The bank # refers to the corresponding VRAM bank number. Byte/Stencil Access Mode applies to all plane groups that access 8 bits of the frame buffer memory (in other words, the 8-bit image and overlay planes). The test expected to find *exp* but observed *obs*, yielding *xor* when the two values are exclusive `or'd` with each other.

Note – The following error messages are software errors. They are grouped together because the cause of the errors is similar. The reason for the errors is described at the end of the list.

`Cannot read Window ID look up table from device [device]. Check device for existence and/or permissions.`

`Cannot write Window ID look up table to device [device]. Check device for existence and/or permissions.`

Cannot read Color LUTs from device [device]. Check device for existence and/or permissions.

Cannot post Color LUTs to device [device]. Check device for existence and/or permissions.

Cannot get monitor mode from device [device]. Check device for existence and/or permissions.

Cannot set diagnostic mode from device [device]. Check device for existence and/or permissions.

Cannot set monitor mode from device [device]. Check device for existence and/or permissions.

Cannot create raster for device [device]. Check device for existence and/or permissions.

Cannot create color map for device [device]. Check device for existence and/or permissions.

Cannot create context for device [device]. Check device for existence and/or permissions.

Cannot create color translation object for device [device]. Check device for existence and/or permissions.

Cannot create path object for device [device]. Check device for existence and/or permissions.

Cannot create child raster for device [device]. Check device for existence and/or permissions.

Cannot create multiple plane group information for device [device]. Check device for existence and/or permissions.

Software error. The device that you specified (the default is `/dev/fbs/leo0`) may not be available to the test, therefore the above operation cannot be performed on this device. Make sure that you are executing the test on a machine with a ZX, and that you have permission to access it, and the device is not being used by another application.

`Cannot start another process. Software error.`

Software error. The process table maybe full and the SPARCstation may have to be rebooted.

`Cannot grab mouse or keyboard because [reason]. May need to bring down other graphic software that is currently running same window server.`

Software error. When the FB locking option is selected, `leotest` also tries to lock the mouse and keyboard but not successful in doing so. The reason can be one of the followings, according the window system software:

```
mouse/keyboard is frozen

grab window is not viewable

grabbed at invalid time

already grabbed by another client
```

`CLUT #n, index #, color [color], expected 0x#, observed 0x#, XOR=0x#`

An error was found in one the three color look up tables tested by SunDiag. The error was found in the *n*th CLUT. The index is out of 256 entries in each CLUT. Each CLUT has eight bits value each for red, green, and blue. The color indicates in which set of eight bits the error was found. The test expected to find *expected* but received *observed*, yielding *XOR* when the two values are exclusive or'd with each other.

`Data file [filename] missing in the current test directory.`

Software error. ZX SunDiag can't find the data file (`leotest.data`) in the current `/opt/SUNWdiag/bin` directory. May have to reinstall the SUNWdiag package in the specified directory.

`Error in [subtest] test.`

The subtest fails and the cause is given in the following message.

`Error in verifying the [Red/Green/Blue] plane at x=#, y=#,bank=#, expected=0x#,observed=0x#, XOR=0x#.`

Failed accelerator port test. The error is in either the Red, Green, or Blue image plane. The x-y coordinate of the pixel should contain *expected* value but *observed* value is received instead, yielding *XOR* value, or the bits in error, when the two values are exclusive or'd with each other.

`Failed to open data file [filename]. Suspect incomplete or incorrect hardware installation. Files may also have been corrupted.`

Indicates a software initialization problem. [*filename*] is the data file that SunDiag can't open.

`Failed to read data file [filename]. Suspect incomplete or incorrect hardware installation. Files may also have been corrupted.`

Indicates a software initialization problem. [*filename*] is the data file that SunDiag can't read.

`Illegal SBus DVMA code = 0x%x addr = 0x%x data= 0x%x. Maybe data file is corrupted.`

`Illegal SBus packet, code = 0x%x addr = 0x%x data = 0x%x. Maybe data file is corrupted.`

This error occurs while the subtest is reading the transaction file. It is likely that the data file is corrupted, so the `SUNWdiag` package may need to be re-installed.

`Out of Memory.`

Out of memory error. Increase swap space and/or kill other processes.

`Pick Detect misses:%d lines and/or triangles inside the pickbox and/or %d lines and triangles outside the pickbox.`

Failed the Picking accelerator port test. Only the failing component (RED, GREEN, or BLUE) appears in the message.

`Pick Echo failed: *** Error(s) found in [RED], [GREEN], [BLUE} components.`

Failed the Picking accelerator port test. Only the failing component (RED, GREEN, or BLUE) appears in the message.

`Picking: *** Error(s) found in [RED], [GREEN], [BLUE} components.`

Failed the picking accelerator port test. Only the failing component (RED, GREEN, or BLUE) appears in the message.

`[Plane group name] Pixel Access Mode error at x=# y=#, bank=#, expected=0x#,observed=0x#, XOR=0x#.`

The direct port video memory test has found an error at pixel (x,y) in the named plane group. Pixel Access Mode applies to all plane groups that access the frame buffer memory four bytes at a time. (In other words, all planes except eight bit planes). The memory for the pixel resides in the given VRAM bank. The test expected to find *expected* but observed *observed*, yielding *XOR* when the two values are exclusive or'd with each other.

`SRAM checksum mismatch. Float #1 = 0x#, Float #2 = 0x#, Float #3 = 0x#, Float #4 = 0x#.`

All four checksums of the SRAMs in the Floating Point Transform section are not identical. The subtest displays the mismatch checksum from each SRAM of each LeoFloat chip.

`SRAM of the LeoFloat [0/1/2/3], number of failures = #, first SRAM location = #, expected = 0x#, observed = 0x#, XOR = 0x#`

An error is found in the SRAM test of the LeoFloat chip number 0, 1, 2, or 3. If the number of failures is more than 1, the subtest display the first SRAM location that fails, the expected and observed values in this location, and the bits in error (values of expected and observed are xor'd together).

`'tar' never finished. System software problem.`

Software error. Make sure that the tar program is installed correctly on your system. Also, use `df` to see if you have enough disk space left in your `/tmp` directory.

`tar: [error]`

Software error. Make sure that the tar program is installed correctly on your system. Also, use `df` to see if you have enough disk space left in your `/tmp` directory.

`TAR failed. Note: A space of approximately 2MB in /tmp is required for the test to run correctly.`

Software error. The tar program cannot unpack data file to the `/tmp` directory because of not enough disk space.

`The checksums for [red/green/blue] image data of all pixels don't match, expected = 0x#, observed = 0x#, XOR = 0x#`

The FB output section subtest prints out this message when the checksums of either red, green, or blue image data *observed* from the trap registers in the Video Output test are not the same as *exp*ected values. It also prints out the error messages below if the upper four bits in the trap registers are not set/reset as expected,

`The Even Field bit expected to be 0, observed 1.`

`The Composite Sync bit expected to be 1, observed 0.`

`The Composite Blank field expected to be 1, observed 0.`

`The Stereo bit expected to be 1, observed 0.`

`The Stereo bit expected to be 1, observed 0.`

Either these bits are not set correctly or the read is from wrong location, which indicates there is an error in the Video Output section.

`Unable to map (access) [device]. Not enough memory.`

Software error. Cannot map the addresses for the ZX device (default /dev/fbs/leo0) because of not enough memory. May have to increase swap space or add more memory.

`Unable to open device [device]. Check device for existence and/or permission.`

Software error. SunDiag is unable to open the ZX device. Make sure that `/dev/fbs/leo0` exists and that the permissions are correct. There may be a software installation problem in which the ZX software packages need to be re-installed.

`Uncompression of data file failed. Note: A space of approximately 2MB in /tmp is required for the test to run correctly`

Software error. Not enough disk space in /tmp, about 2 Megabytes to uncompress data file. You may have to remove unneeded files or link `/tmp` to a bigger disk partition.

`Unknown data file magic number = 0x#.`

Software error. The data file was generated by an older version of software tools. Report this error by filing a bug report or calling the Sun 800 number.

`Unsupported 24-bit data length. Maybe data file is corrupted.`

This error occurs while the subtest is reading the transaction file. It is likely that the data file is corrupted, so the `SUNWdiag` package may need to be re-installed.

`Unable to open display. Window server not running.`

Warning message only. This message is displayed when the ZX SunDiag is executed from the command line remotely or if SunDiag is run in TTY mode.

`vfork: [error]`

Software error. An error has occurred while trying to fork a child process. Increase swap space, or close other running processes.

`[PWID/QWID] WLUT: Look up table error at index #, expected 0x#, observed 0x#, XOR 0x#.`

An error was found in either PWID (Hardware Window ID) or QWID (Software Window ID) look up tables tested by SunDiag. The error was found in the *n*th WLUT. The index is out of 64 entries for PWID or 15 for

QWID. The test expected to find *expected* but received *observed*, yielding *XOR* when the two values are exclusive or'd with each other, which indicates the bits in error. This error message indicates there is an error in the Video Output section.

3.7 *NeWSprinter Test (*`spdtest`*)*

3.7.1 `spdtest` *Description*

The `spdtest` is a two part test that checks the printer support hardware. The first part is a register test that checks the NeWSprinter™ 20 SBus printer card's internal functions. The second part is a printing test that checks the interaction between the printer and the print server, as well as the printer's own capabilities.

3.7.2 `spdtest` *Options*

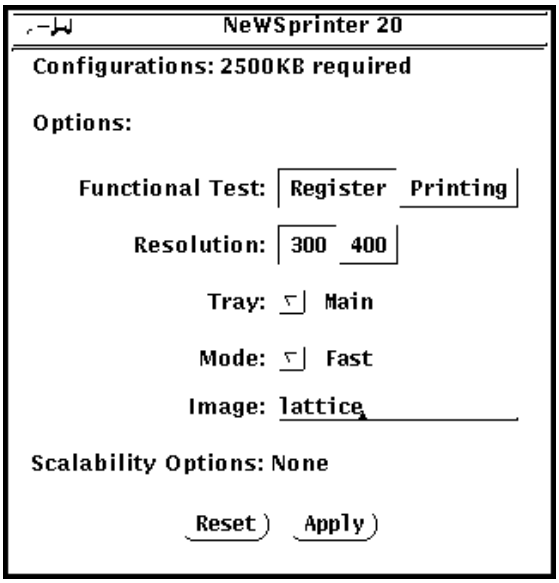


Figure 3-7 NeWSprinter Option Window

Functional Test

The Functional test is composed of two subtests. The first subtest is the Register test that checks the NeWSprinter 20 SBus printer card. The second subtest is the Printing subtest, which tests the printer's capabilities.

The default setting is Register.

Resolution

This exclusive setting defines the printer resolution of the printer test pattern. You can choose either 300 or 400 dots per inch.

The default setting is 300 dots per inch.

Tray

From the Tray menu, you can choose from five different types of paper cassettes:

- Main Tray (or Tray 1) is the top cassette that feeds paper to the printer.
- Tray 2 is the bottom cassette that feeds paper to the printer.
- Tray 3 is an optional tray.
- Manual Tray is a manual feed tray connected to the rear of the printer. This tray is used to hand-feed print media which cannot be accommodated by the paper cassettes.
- Auxiliary Feeder is optionally installed in place of the manual feed tray. The Feeder stores and automatically feeds print media that would otherwise require the manual feed.

The default setting is the Main tray.

Mode

From the Mode menu, you can set the interval between image printings. The choices are:

- Fast prints an image every 10 seconds.
- Medium prints an image every 12 minutes.
- Extended prints an image every 30 minutes.

The default setting is Fast.

Image

This option allows you to enter the filename of the test image to print.

3.7.3 `spdtest` *Command Line Syntax*

`/opt/SUNWdiag/bin/spdtest F=`*function* **`D=`***devicename* **`L=`***loop* **`R=`***resolution* **`I=`***image* **`T=`***tray* **`M=`***mode*

Arguments	
`F=`*function*	The *function* value represents one of the following functional tests: 0 = Register test 1 = Printing test
`D=`*devicename*	The full pathname of the device must be substituted for *devicename*.
`L=`*loop*	Substitute *Loop* for the number of times you want to run the test.
`R=`*resolution*	Replace *resolution* with a number representing the printer resolution: 0 = 300 dpi 1 = 400 dpi
`I=`*image*	Replace *image* with the name of a rasterfile.
`T=`*tray*	The *tray* number represents the type of printer cassette the printer uses: 0 = Main tray (Tray 1) 1 = Tray 2 2 = Tray 3 (optional) 3 = Manual Tray 4 = Auxiliary Feeder (optional)
`M=`*mode*	The *mode* number represents how often to print the test image: 0 = Fast (Prints an image every 10 seconds) 1 = Medium (Prints an image every 12 minutes) 2 = Extended (Prints an image every 30 minutes)

3.8 SunVideo Test (`rtvctest`)

The `rtvctest` verifies the functionality of the SunVideo™ SBus card. SunVideo technology captures and compresses video input in real-time, making it possible to have real-time video conferencing over standard Ethernet networks.

3.8.1 `rtvctest` Test Description

The `rtvctest` is divided into four sub-tests: PROMCheck, memory, Jalapeno, and CL4000. The PROMCheck sub-test verifies the SunVideo card's programmable read only memory. The memory test verifies all of the memory on the card, including the 2 Mbytes of memory on the CL4000 compression engine and the memory on the Jalapeno application-specific integrated circuit (ASIC). The Jalapeno sub-test verifies the interface logic between the SBus, A/D conversion chips, and the CL4000 compression engine. The CL400 sub-test verifies that the compression engine ASIC is able to compress digitized video data from the A/D chips and send this data to the SBus, through the Jalapeno ASIC.

The `rtvctest` is composed of 49 verification test modules. Table 3-3 lists these modules and their associated test sequence numbers.

Table 3-3 `rtvctest` Verification Modules

SunVideo Verification Module Name	Test Sequence Number
RTVC SUNDIAG Start	0
RTVC Checksum	1
RTVC Jalapeno SMEM	2
RTVC CL4000 DMEM	3
RTVC Jalapeno SBus Interrupt Mask	4
RTVC CL4000 Interrupt Mask	5
RTVC DVMA Control Register	6
RTVC DVMA Transfer Size Counter	7
RTVC DVMA Memory Address Counter	8
RTVC DVMA Virtual Memory Address Counter	9
RTVC DVMA Slave SBus Rerun Register	10

Table 3-3 `rtvctest` Verification Modules *(Continued)*

SunVideo Verification Module Name	Test Sequence Number
RTVC IIC Control Register	11
RTVC IIC Data Register	12
RTVC Video DMA Control Register	13
RTVC Video DMA Transfer Size Counter	14
RTVC Video DMA Memory Address Counter	15
RTVC User Interrupt 0	16
RTVC User Interrupt 1	17
RTVC User Interrupt 2	18
RTVC User Interrupt 3	19
RTVC Video Control and Status Register	20
RTVC Video Control Field Line Interrupt 1	21
RTVC Video Control Field Line Interrupt 2	22
RTVC Video Scan Line Mask Registers	23
RTVC Video Input Format Type	25
RTVC Video Horizontal Lock	26
RTVC Video Even Odd Field	27
RTVC CL4000 Host Control	28
RTVC CL4000 Host Lock	29
RTVC CL4000 Video Port A Control	30
RTVC CL4000 Video Port B Control	31
RTVC CL4000 Video Port A FIFO	32
RTVC CL4000 Video Port B FIFO	33
RTVC CL4000 Address Memory Registers	34
RTVC CL4000 Instruction Memory Access Registers	35
RTVC Time Stamp Register	36
RTVC CL4000 Register Memory	37
RTVC CL4000 Scratch Memory	38

Table 3-3 `rtvctest` Verification Modules *(Continued)*

SunVideo Verification Module Name	Test Sequence Number
RTVC CL4000 DMA Mode	39
RTVC CL4000 Motion Estimation Registers	40
RTVC CL4000 PSW	41
RTVC CL4000 Variable Length Coder Registers	42
RTVC CL4000 Channel Memory Registers	43
RTVC CL4000 CPU Control	44
RTVC CL4000 Multiply Control	45
RTVC CL4000 DMA Interrupt Control	46
RTVC CL4000 Block Transfer Mode	47
RTVC CL4000 Accumulator MSB	48
RTVC CL4000 JPC Field	49
RTVC SUNDIAG Finish	50

3.8.2 `rtvctest` *Options*

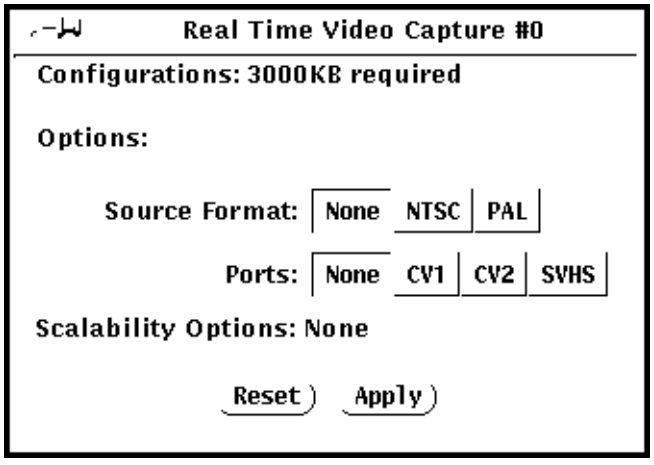


Figure 3-8 `rtvctest` Option Window

You may test the SunVideo card without any video device (camera, video disc player, or VCR) connected to a input port. However, if you connect a video device to the card, then you must state the format of the video source and the input port which the device is connected.

Note – If you do not state both the video source and the input port, the `rtvctest` will print an error and terminate testing.

Source Format

Select the format of the video source used for testing. You may select from the following sources:

Video Source	Definition
None	No video source.
NTSC	A National Television System Committee (NTSC) video source, which is the video standard in the United States and Japan.
PAL	A phase alternate line video source, which is the video standard in Europe.

Ports

If you have selected a video source for testing, then you must state which input port you have used to connect the source to the SunVideo card. You may select from the following ports:

Input Port	Definition
None	No input port used.
CV1	The Composite Video Input Port 1 (RCA type input).
CV2	The Composite Video Input Port 2 (RCA type input).
SVHS	The Super VHS input port.

3.8.3 `rtvctest` *Command Line Syntax*

`/opt/SUNWdiag/bin/rtvctest [ntsc|pal] [cv1|cv2|svhs] D=`*rtvc_device*

Arguments	
`[ntsc|pal]`	Select the format of the video source used for testing. Select either **`ntsc`** (video standard in the United States and Japan) or **`pal`** (video standard in Europe).
`[cv1|cv2|svhs]`	If you have selected a video source for testing, then you must state which input port you have used to connect the source to the SunVideo card. You may select from the following ports: **`cv1`** Composite video input port 1 (RCA type) **`cv2`** Composite video input port 2 (RCA type) **`svhs`** Super VHS port
`D=`*rtvc_device*	You are required to state the SunVideo device when running the rtvctest from the command line. Replace *rtvc_device* with **`rtvc`***n*, where *n* is the device number (from 0 to 31) of the SunVideo card under test.

3.8.4 `rtvctest` *Quick Test Description*

Running this test in quick mode abbreviates the test procedure. The quick mode test performs a subset of the available test modules, which should provide a quicker check of the SunVideo hardware.

3.8.5 `rtvctest` *Error Messages*

SunVideo Error Message #10

%s

Where:

%s is one of the following message:

```
Missing REQUIRED argument D=rtvc[0..31]
```

SunVideo Error Message #20

%s `Error #` *%d*

Where:

%s is one of the following message:

```
ioctl RTVC_CMD_RESET fault
ioctl RTVC_CMD_SET_VIDEO fault
ioctl RTVC_CMD_GET_VIDEO fault
open /dev/rtvc fault
close /dev/rtvc fault
open /dev/rtvcctl fault
close /dev/rtvcctl fault
Unknown Jalapeno Version
```

%d is one of the following number:

```
ioctl error
module version
```

SunVideo Error Message #30

%s1 %s2 `Error # %d`

Where:

%s1 is one of the following message:

```
memory map fault for
memory unmap fault for
selected /dev/rtvc and /dev/rtvcc1 not available
```

%s2 is one of the following message:

```
prom
sram
buses
reset
dmem
rtvc
```

%d is the error number

SunVideo Error Message #40

```
RTVC Fault Detected via module %s1
Physical Address %X Expected Value %X Actual Value %X
RTVC Sundiag Module number %d
Location %s2
Message: %s3
```

Where:

%s1 is the name of the module under test

%X are the values associated with physical address, expected value, actual value

%d is the Sundiag Module under test

%s2 is the U location of the faulty component

%s3 are any additional messages

SunVideo Error Message #41

`RTVC Fault Detected via module` *%s*

Where:

%s is the name of the module under test

SunVideo Error Message #42

%s1 %X1 %s2 %X2

Where:

%s1 is one of the following message:

```
Video Format Expected
Video Port Expected
```

%X1 is the expected value

%s2 is one of the following message:

```
Actual
```

%X2 is the actual value

SunVideo Error Message #50

%s %d

Where:

%s is one of the following message:

```
Unknown RTVC CL4000 test module
Unknown RTVC Jalapeno test module
Unknown RTVC test module
Unknown RTVC memory test module
Unknown Start Bit
```

%d is the module number or start bit location

RTVC Verbose Information #10

%s

Where:

%s is one of the following message:

```
Video Format OR Port NOT Selected - Test Skipped
Probe module : NOT XQT under SUNDIAG
Probe module : XQT under SUNDIAG
```

RTVC Verbose Information #20

%s1 %s2 `Status` *%X*

Where:

%s1 is one of the following message:

```
Finished
Started
```

*%s*2 is one of the following message:

```
Module Name
Sundiag
Walking One
Walking Zero
Data Line
Address Line
Stuck At Fault
Modulus 3
Read Write Collision
```

%X is fault detected flag

RTVC Verbose Information #30

`Executing module` *%d %s*

Where:

%d is the test module number

%s is the test module name

3.9 *PCMCIA Modem Card Test (*`pcmciatest`*)*

This test verifies the functionality of the PCMCIA Modem Card.

3.9.1 `pcmciatest` *Test Description*

The `pcmciatest` issues a series of commands to the modem which instructs the modem's firmware to run an internal analog loopback diagnostic test. Upon completion, the firmware sends back a three digit status message indicating whether the test passed or failed.

As an option, the `pcmciatest` will test socket I/O cards. This test will write an 8 Kbyte incrementing data pattern to the I/O card which is then looped back, read, and verified.

Note – When testing socket I/O cards, a 9-pin loopback connector is required. However, no loopback connector is required when testing the default modem card. See Appendix A, "Loopback Connectors," for loopback connector wiring instructions.

3.9.2 `pcmciatest` *Options*

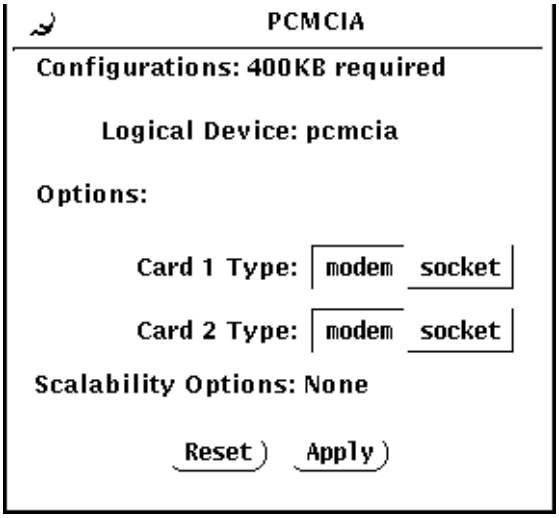


Figure 3-9 `pmciatest` Option Window

The default card type for each PCMCIA slot is a modem card. If only one modem card is plugged in, the empty slot will be ignored.

From the `pcmciatest` Option Window, you can choose to test socket I/O cards. To test a socket card, choose socket on the Card Type switch. If you choose to test a socket I/O card in an empty slot, the test will fail.

Any combination of modem and socket I/O cards may be placed in the PCMCIA slots. However, you must select the correct type of card in the Option Window. If you select an incorrect card type, the test will fail.

3.9.3 `pcmciatest` *Command Line Syntax*

`/opt/SUNWdiag/bin/pcmciatest o=`*card_type* **`t=`***card_type*

Arguments	
`o=`*card_type*	Specify the card type for slot 1 (**`o`** is short for "one"). *card_type* is the type of card in slot 1. Type **`o=s`** if there is a socket I/O card in slot 1. If there is a modem card in slot 1, then you do not need to specify this argument.
`t=`*card_type*	Specify the card type for slot 2 (**`t`** is short for "two"). *card_type* is the type of card in slot 1. Type **`t=s`** if there is a socket I/O card in slot 2. If there is a modem card in slot 2, then this argument is not needed.

3.10 *Infrared Interface Test (`irtest`)*

The `irtest` tests the SPARCstation Voyager™ system's infrared (IR) interface.

3.10.1 `irtest` *Test Description*

The `irtest` is a two part test: a loopback test and an IR transmit test. In the loopback test, the `irtest` will send and receive an 8 Kbyte incrementing data pattern through an internal loopback within the Multi-Interface Chip's (MIC) Serial Communication Controller (SCC).

In the second part of the test, the `irtest` will enable the IR transmitter and receiver. The IR mode will be set to PULSE and the baud rate will be set to 115200. The test will then send the same 8 Kbyte data stream (broken up into 64 byte blocks) out the transmitter, allowing each byte to be detected and read by the receiver. The test will be repeated with the receiver disabled to make sure that the receiver does not pick up any data coming out of the transmitter

Note – The 8 Kbyte data is broken up into 64 byte blocks because this block size is the maximum amount of data that can be transmitted through the IR interface without errors.

Currently, only the PULSE mode is tested by the irtest. The error rate for HIGH mode is too high to permit this type of IR loopback testing.

3.10.2 `irtest` *Options*

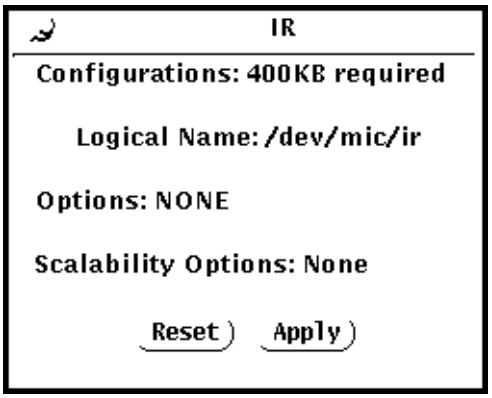


Figure 3-10 `irtest` Option Window

The `irtest` currently has no options.

3.10.3 `irtest` *Command Line Syntax*

```
/opt/SUNWdiag/bin/irtest
```

3.11 *SPARCstorage Array Controller Test (*`plntest`*)*

The `plntest` test checks the controller board on the SPARCstorage™ Array.

3.11.1 `plntest` *Test Description*

The SPARCstorage Array (SSA) is a large disk storage I/O subsystem capable of housing up to 30 SCSI hard drives. The SSA communicates with a host system over a fiber optic link provided by an SBus-based host adapter card in the host system and the corresponding SSA Controller board hardware.

The SSA Controller card is an intelligent, CPU-based board with its own memory and ROM-resident software. In addition to providing a communications link to the disk drives, it also buffers data, between the host system and disk drives, in its non-volatile RAM (NVRAM). In order for data to go from the host to a particular disk, it must first be successfully transferred to this NVRAM space.

To perform this data transfer operation, the host machine, SBus host adapter card, fiber-channel connection, and the SSA controller board must be working properly. It is precisely this operation that `plntest` tries to stress and verify. By stressing this operation, `plntest` can isolate failures on the SSA disk drives from failures on the SSA Controller board.

Note – `fstest` and `rawtest` will transfer data on the SSA disk drives over the same path mentioned above. However, they will not be issuing these data transfers as quickly as `plntest` can.

`plntest` ***Stress Test***

The `plntest` attempts to stress the SPARCstorage Array hardware and software by issuing a large number of transfers between the host system and the SSA controller board.

The `plntest` exercises the hardware and software by invoking many SCSI read buffer and write buffer commands, of various sizes and data patterns, to the NVRAM. These operations exercise the host fiber channel hardware, the SSA fiber channel hardware, the SSA resident management software (PMF), and the hardware component interaction on the SSA controller card (all components except the SCSI ones).

3.11.2 `plntest` *Options*

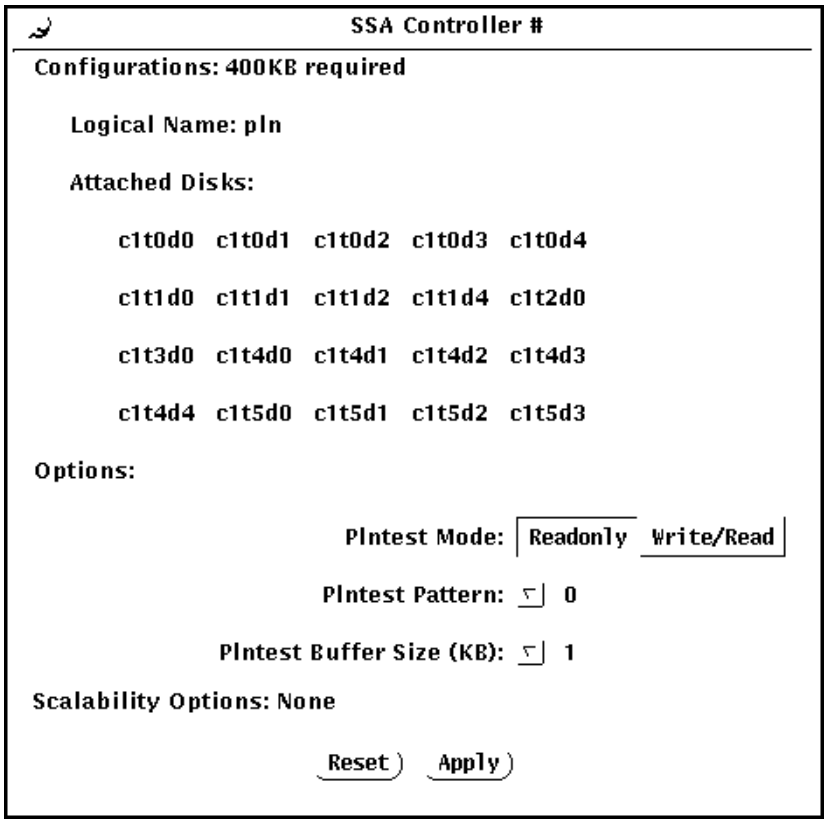


Figure 3-11 `plntest` Option Window

Configurations

The Configuration area lists the names of all the logical disk drives (both single and grouped) that are attached to the SSA controller board.

Note – If no disks are present, the window will display “none” under the Attached Disks heading.

Plntest Mode

By default, `plntest` is run in read-only mode, in which the test simply reads blocks of memory from the SSA controller.

Click on the Write/Read choice to enable the Write/Read test mode. In Write/Read mode, `plntest` writes data (as specified by the Pattern option below) to the SSA controller memory. This data is then read back and compared.

Plntest Pattern

The Pattern option allows you to specify the exact data pattern that will be written to the SSA memory. The data pattern choices are: 0's. 1's, 5's, a's, and Random. Random is the default data pattern.

Plntest Buffer Size

The Buffer size determines the size of the data transfers between the SSA controller card and the host system. The buffer size choices are: 1, 2, 5, 10, 50, 100, 200, 500, or Random Kbytes. The default Random size selects buffer sizes between 1 and 500 Kbytes per transfer.

Note – `plntest` is not a scalable test.

3.11.3 `plntest` *Command Arguments*

`/opt/SUNWdiag/bin/plntest ? D=`*device_name* **`x W P=`***pattern* **`B=`***buffer_size* **`Kb`** *standard_arguments*

Arguments	
`?`	Probes the system for valid SSA Controller devices and prints them to screen (see Section 3.11.3.1, "Probing for SSA Controller Devices.")
`D=`*device_name*	The physical pathname of the SSA controller card to be tested. This argument *must* be included when running `plntest` from the command line, unless the **`?`** argument is used.

Arguments	*(Continued)*
x	Probes the specified SSA controller card for the single and grouped disks attached to the controller card, and prints their logical names to the screen. **Note:** The **D=***device_name* must be specified for this option to work.
W	Invoke the test in Write/Read mode. The default mode is read-only. **Note:** The **D=***device_name* must be specified for this option to work.
P=*pattern*	Choose the data pattern used in Write/Read mode. Choose one of the following data patterns: a all A's (hexadecimal) 0 all 0's 1 all 1's 5 all 5's r Random **Note:** The **D=***device_name* and **W** arguments must be specified to make this option work.
B=*buffer_size* **Kb**	Size of the buffer, in Kbytes, that will be transferred to and from the host per test iteration. The buffer size can range from 1 to 500 Kbytes, or **r** for random (default). **Note:** The **D=***device_name* must be specified for this option to work.

3.11.3.1 *Probing for SSA Controller Devices*

Unlike most other hardware devices, the SSA controller card has not yet been assigned a logical device name (one you would find in the `/dev` directory). Therefore, the SSA controller card is identified by its longer, physical device name.

When running pltest from the command line, the physical device name of the SSA controller card must be specified. To avoid typing the long physical device name, use the ? option. The ? option will probe the system and print out the physical paths of all the attached SSA controller devices.

If you are going to invoke `plntest` from the command line often, it may become bothersome to type the physical device name each time. However, you can create your own logical name and link it to the physical name. You will then be able to invoke plntest with the shorter logical name.

```
machine# ./plntest "?"

1: /devices/io-unit@f,e3200000/sbi@0,0/SUNW,soc@1,0/SUNW,pln@0c0d,0e0f0102:ctlr
2: /devices/io-unit@f,e0200000/sbi@0,0/SUNW,soc@3,0/SUNW,pln@0c0d,0e0f0102:ctlr

machine# ln -s \
/devices/io-unit@f,e3200000/sbi@0,0/SUNW,soc@1,0/SUNW,pln@0c0d,0e0f0102:ctlr \
/dev/ssa1
machine# /opt/SUNWdiag/bin/plntest D=/dev/ssa1
```

3.11.4 `plntest` *Quick Test Description*

In quick test mode, the `plntest` will limit itself to 500 data transfers between the host and the SSA.

3.11.5 `plntest .usertest` *File Command Line*

The following is a sample `.usertest` command:

```
pln0, plntest, D=/dev/pluto0 W P=r B=r
```

3.11.6 `plntest` *Error Messages*

`Unable to determine physical pathname for` *%s*

`Error in Closing device:` *%d*

`Error. Data Mismatch in Controller Memory`

`Error in validating device name`

`Can't open /dev/dsk directory`

```
Invalid device name

No Pluto Controller device specified.  Use ‘D=’ option

No Pluto Controller devices found

No disks found on this Pluto Controller

Error in Opening device: %d

Error trying to probe device: %d

USCSI Read Buffer Error. USCSI ioctl() return %d

SCSI command to pln:ctlr failed. status = 0x%x

USCSI ioctl error. ioctl() return %d

USCSI Write Buffer Error. USCSI ioctl() return %d
```

User Test Descriptions 4

This chapter describes SunDiag tests for hardware devices that SunDiag will not automatically detect during the device probe at start-up. Therefore, these tests do not appear on the SunDiag control or status panels, even though they are included in the `/opt/SUNWdiag/bin` directory.

There are two ways to run these tests: you can either create a `.usertest` file, or run each test individually from a command line. See “Adding Your Own Tests in `.usertest`” in Chapter 1 of the *SunDiag User’s Guide* for details on creating a `.usertest` file.

4

4.1 *SunDials Test (*`sundials`*)*

4.1.1 `sundials` *Test Description*

From a `.usertest` ***File***

This test verifies that the SunDials™ graphics manipulation device controls are working properly. `sundials` also verifies the connection between the dialbox and serial port.

Here is an example of a `.usertest` entry for `sundials`:

```
bd,sundials,s
```

From a Command Line

Running `sundials` from a command line starts an interactive test that displays a screen representation of the dialbox. You can move each of the dials and see the corresponding dial's display change. To run the interactive test, select the Diagnostics Button on the top of the window representation. The dialbox dials on the screen will not move while the diagnostics test is running

Be certain that the dialbox is connected to one of the serial ports, and that the dialbox is complete with a power transformer to power it.

There is no option menu for this test.

4.1.2 `sundials` *Command Line Syntax*

`/opt/SUNWdiag/bin/sundials diag` *standard_arguments*

4.1.3 `sundials` *Quick Test Description*

Running this test in quick mode does not change the test procedure.

The `sundials` test window looks like this:

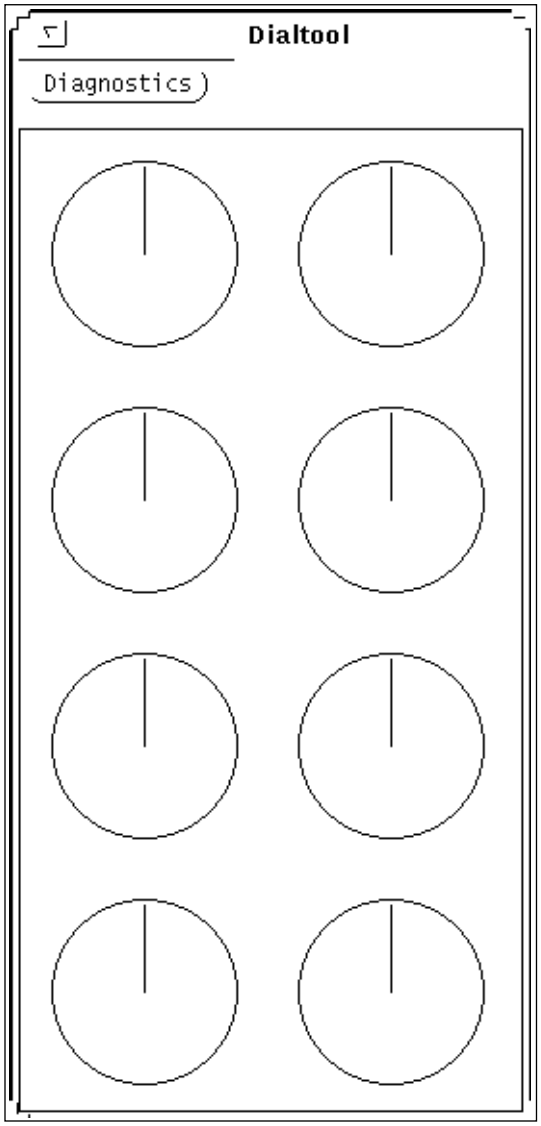


Figure 4-1 `sundials` Test Window

4.1.4 `sundials` *Error Messages*

`Sundials O.K.`

The test successfully completed

`Cannot open device.`

The device was probably not connected to the serial port.

`ioctl(VUIDSFORMAT, VUID_NATIVE) -- you may need to run dbconfig.`

`dbconfig` was probably not run and the test is getting an `ioctl` error when trying to access the device.

`No Response from Dialbox`

The dialbox is connected properly, but it may not be getting power from the transformer.

`Selftest Failed`

The dialbox selftest failed.

4.2 *SunButtons Test (*sunbuttons*)*

4.2.1 sunbuttons *Test Description*

This test verifies that the SunButtons™ graphics manipulation device is working correctly.

From a .usertest *File*

In the non-interactive mode (from a .usertest file) this test verifies that each button functions. You will see each button light up in a round-robin fashion.

From a Shell Command Line

This is an interactive test. The test will display a screen representation of the buttonbox where you can press each of the buttons and see the corresponding button's display change. To run the diagnostic test, select the Diagnostics Button on the top of the window representation. The buttonbox buttons on the screen will not change while the diagnostics test is running.

To run the test, select the Diagnostics button. As the test runs, you will see each button light in a "round-robin" fashion.

Be certain that the buttonbox is connected to one of the serial ports, and that the buttonbox is complete with a power transformer to power it.

There is no option menu for this test.

4.2.2 sunbuttons *Command Line Syntax*

/opt/SUNWdiag/bin/sunbuttons diag *standard_arguments*

4.2.3 sunbuttons *Quick Test Description*

Running this test in quick mode does not change the test procedure.

The `sunbuttons` test window looks like this:

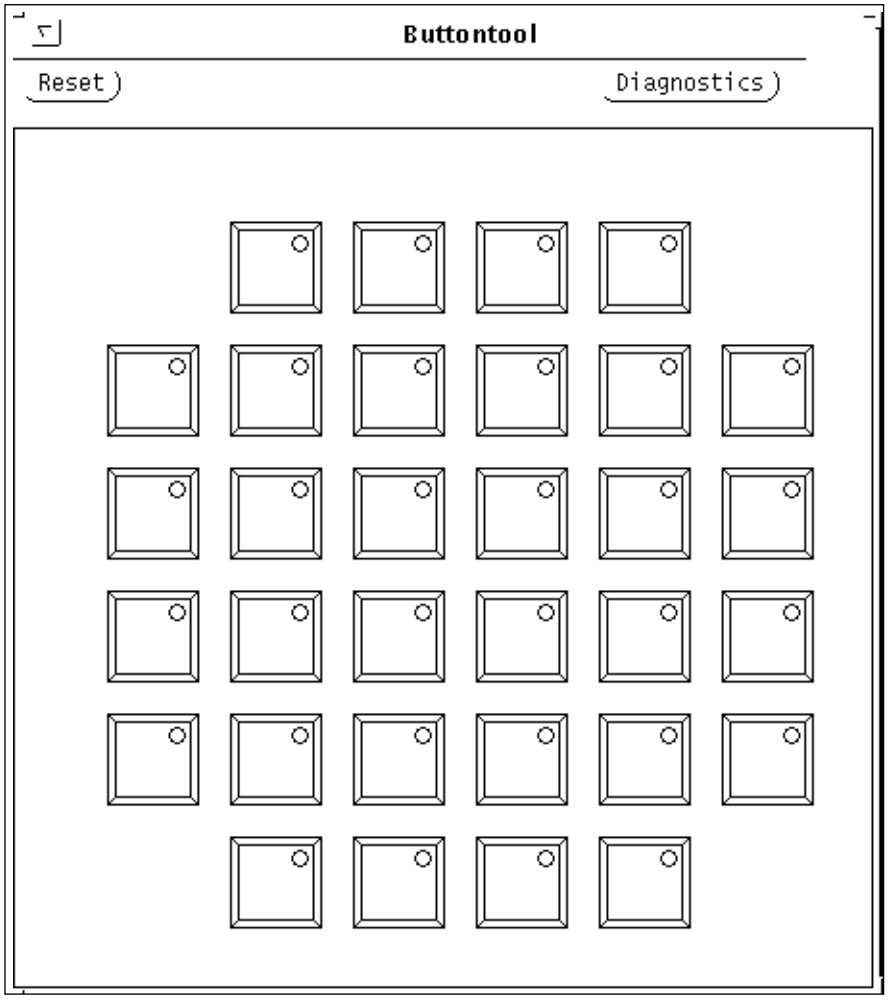


Figure 4-2 `sunbuttons` Test Window

4.2.4 `sunbuttons` *Error Messages*

`Sunbuttons O.K.`

The test successfully completed

`Cannot open device`

The device was probably not connected to the serial port.

`ioctl(VUIDSFORMAT, VUID_NATIVE You may need to run dbconfig.`

`dbconfig` was probably not run and the test is getting an `ioctl` error when trying to access the device.

`No Response from buttonbox`

The buttonbox is connected however it may not be getting power from the transformer.

`Selftest Failed`

The Buttonbox selftest failed.

Loopback Connectors A

These loopback connectors are designed to be used with the tests described in this Addendum. See Appendix B of the *SunDiag User's Guide* for more loopback connectors.

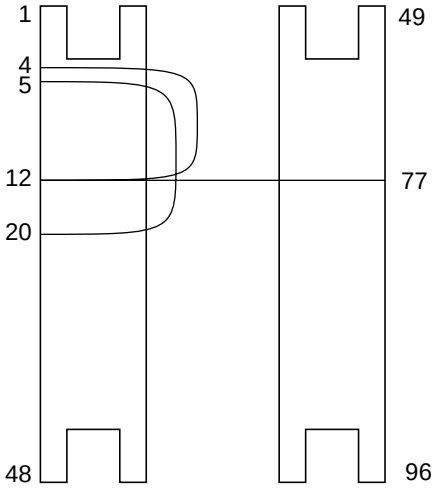
Most of the loopback connectors described in this appendix are available from Sun; the part numbers are given where applicable. To obtain a loopback connector kit, contact Sun Customer Support.

A

A.1 96-Pin Loopback Connector

This 96-pin connector can be ordered from Sun Microsystems (Part Number 370-1366).

Female



Materials:

PCR-E96FA	(1)
PCS-E96LKPA	(1)
3751 Metal Plug (9563K42)	(1)
AWG28 Madison Cable UL/CSA Approved	(8” long)

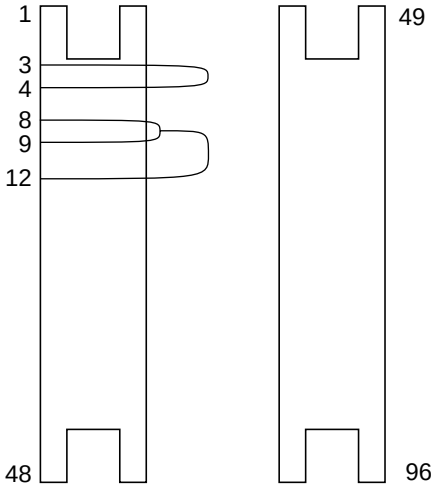
Connect:

pins 4 and 12	to	pin 77
pin 5	to	pin 20
pin 6	to	pin 36
pin 7	to	pin 44
pin 10	to	pin 11
pin 13	to	pin 16
pin 14	to	pin 15
pin 18	to	pin 19
pin 21	to	pin 24
pin 28	to	pin 60
pin 29	to	pin 68
pin 30	to	pin 34
pin 37	to	pin 40
pin 38	to	pin 39
pin 42	to	pin 43
pin 45	to	pin 48
pin 46	to	pin 47
pin 52	to	pin 78
pin 53	to	pin 55
pin 54	to	pin 75
pin 58	to	pin 59
pin 61	to	pin 64
pin 62	to	pin 63
pin 66	to	pin 67
pin 69	to	pin 72
pin 76	to	pin 79
pin 82	to	pin 83
pin 85	to	pin 88
pin 86	to	pin 87
pin 90	to	pin 91
pin 93	to	pin 96
pin 94	to	pin 95

A.2 96-Pin Loopback Connector

This 96-pin connector can be ordered from Sun Microsystems (Part Number 370-1381).

Female Special



Connect:

pin 3	to	pin 4
pin 5	to	pin 7
pins 8 and 9	to	pin 12
pin 10	to	pin 11
pin 13	to	pin 14
pin 15	to	pin 17
pins 18 and 19	to	pin 22
pin 20	to	pin 21
pin 27	to	pin 28
pin 29	to	pin 31
pins 32 and 33	to	pin 36
pin 34	to	pin 35
pin 37	to	pin 38
pin 39	to	pin 41
pins 42 and 43	to	pin 46
pin 44	to	pin 45
pin 51	to	pin 52
pin 53	to	pin 55
pins 56 and 57	to	pin 60
pin 58	to	pin 59
pin 61	to	pin 62
pin 63	to	pin 65
pins 66 and 67	to	pin 70
pin 68	to	pin 69
pin 75	to	pin 76
pin 77	to	pin 79
pins 80 and 81	to	pin 84
pin 82	to	pin 83
pin 85	to	pin 86
pin 87	to	pin 89
pins 90 and 91	to	pin 94
pin 92	to	pin 93

Materials:

PCR-E96FA	(1)
PCS-E96LKPA	(1)
9563K999 Cap	(1)
Madison Cable	(6” long)
(#28 SCSI UL/CSA Approved)	

Open Contacts:

1, 2, 6,16, 23, 24, 25, 26, 30, 40, 47, 48, 49, 50, 54, 64, 71, 72, 73, 74, 78, 88, 95, 96

A.3 37-Pin RS-449 Loopback Cable

Use these wiring instructions for a loopback cable for two 37-pin RS-449 synchronous ports.

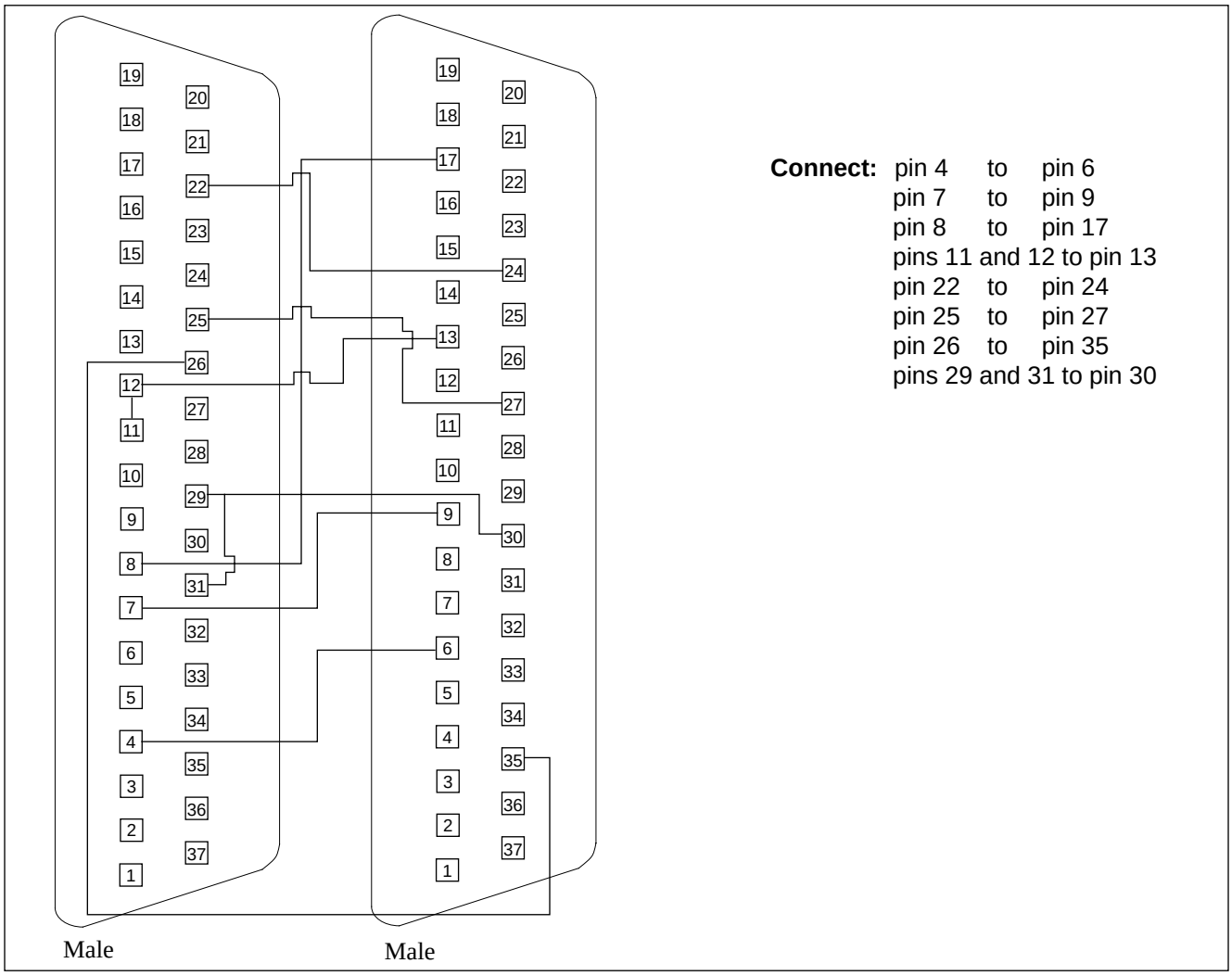


Figure A-1 37-Pin RS-449 Loopback Cable

A.4 37-Pin RS-449 Loopback Plug

Use these wiring instructions for making a male 37-pin RS-449 loopback plug. This connector is also available from Sun (Part Number 530-1430).

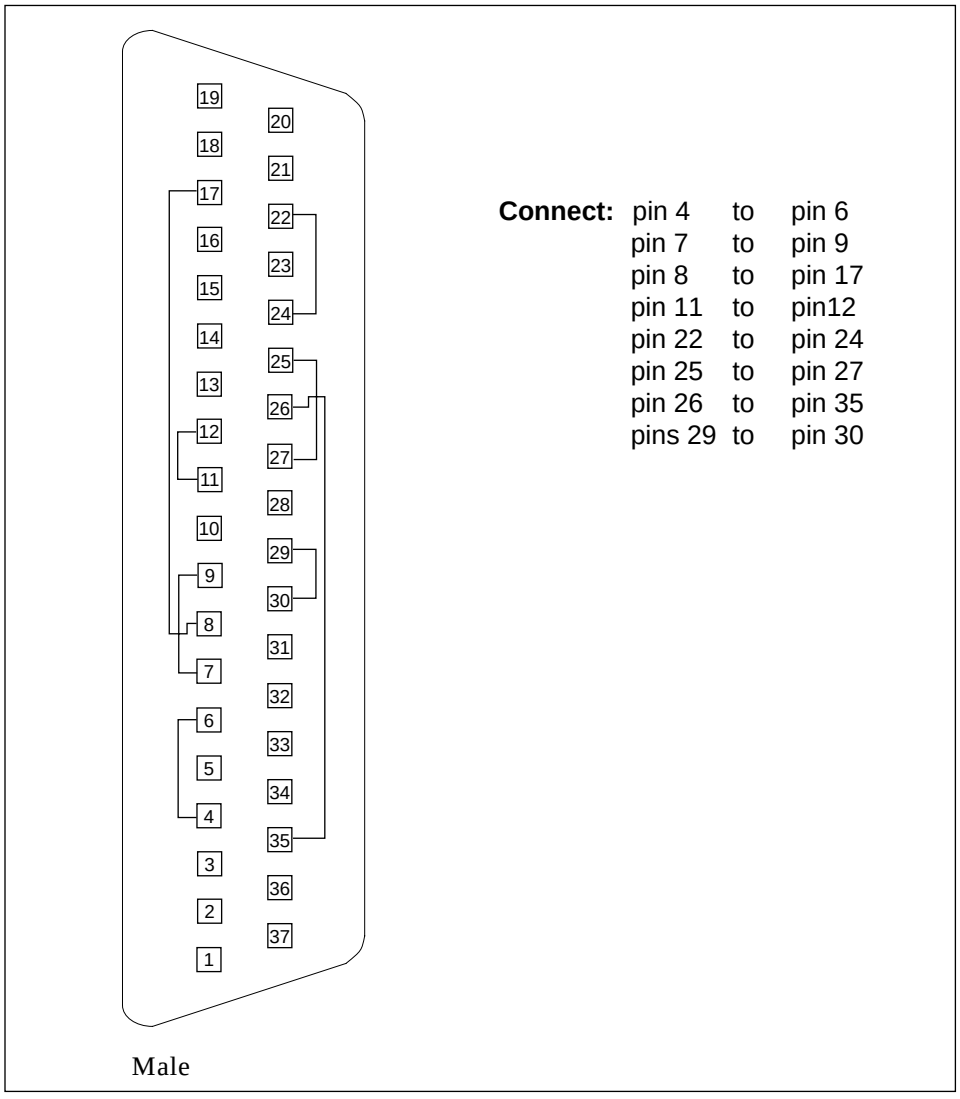


Figure A-2 37-Pin RS-449 Loopback Plug

A.5 9-pin Single-port Loopback Plug

Use these wiring directions for male 9-pin RS-232 and RS-423 single-port loopback plugs:

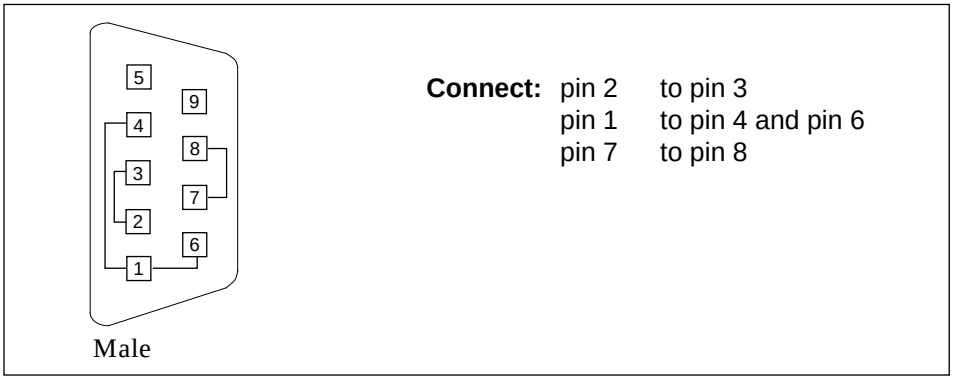


Figure A-3 9-Pin Single-port Loopback Plug

A.6 9-pin Single-port Loopback Plug

Use these wiring directions for female 9-pin RS-232 and RS-423 single-port loopback plugs:

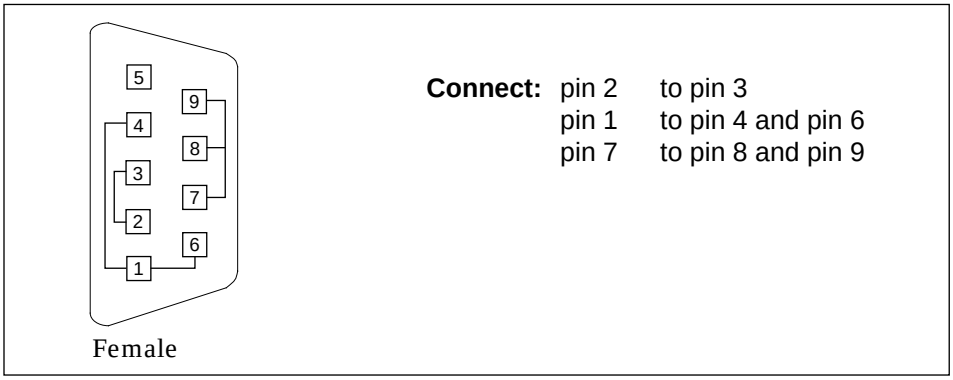
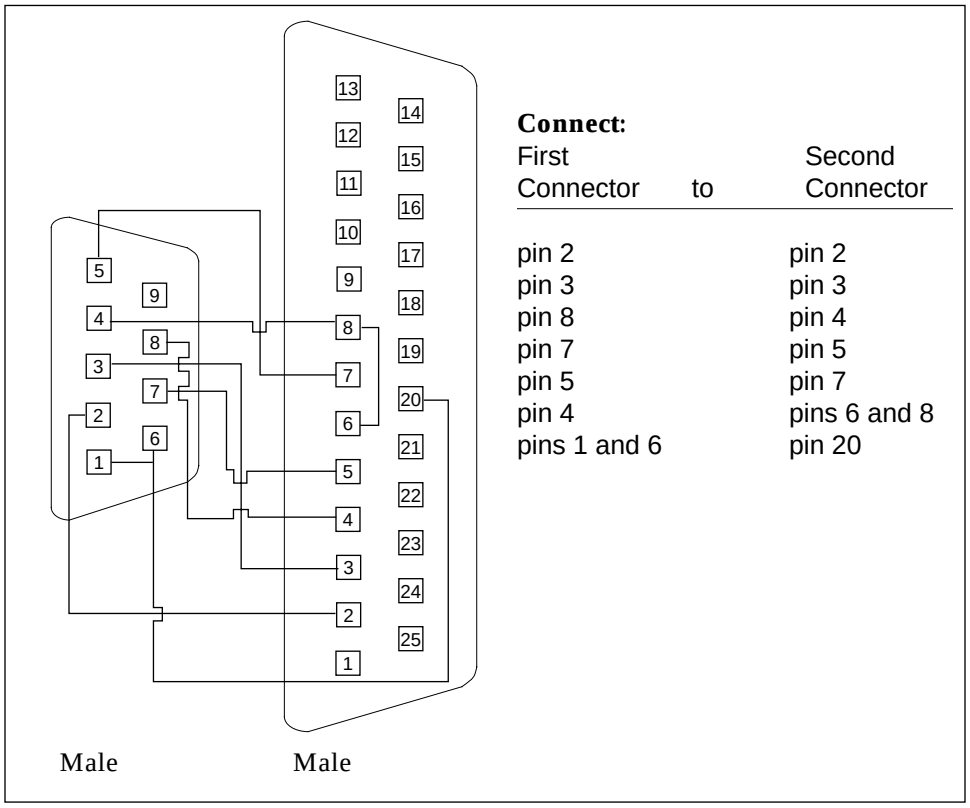


Figure A-4 9-Pin Single-port Loopback Plug

Note – Use this loopback plug with the `pcmciatest`.

A.7 9-Pin to 25-Pin Port-to-Port Loopback Cable

Use these wiring directions for a 9-pin RS-232 and RS-423 port to 25-pin RS-232 and RS 423 port loopback cables. Both connectors are male.



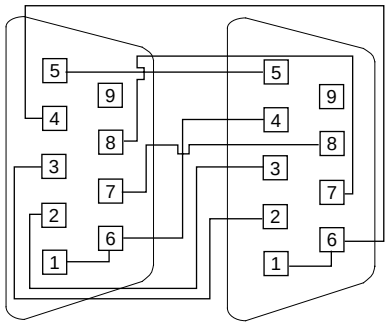
Connect:

First Connector	to	Second Connector
pin 2		pin 2
pin 3		pin 3
pin 8		pin 4
pin 7		pin 5
pin 5		pin 7
pin 4		pins 6 and 8
pins 1 and 6		pin 20

Figure A-5 9-Pin to 25-Pin Port-to-Port Loopback Cable

A.8 9-Pin to 9-Pin Port-to-Port Loopback Cable

Use these wiring directions for 9-pin RS-232 and RS 423 port to 9-pin RS-232 and RS-423 port loopback cables. Both connectors are male.



Connect:

First Connector	to	Second Connector
pins 1 and 6		pin 4
pin 2		pin 3
pin 3		pin 2
pin 4		pins 1 and 6
pin 5		pin 5
pin 7		pin 8
pin 8		pin 7

Figure A-6 9-Pin to 9-Pin Port-to-Port Loopback Cable

Please note that this cable has no Sun part number assigned to it.

A.9 NT to TE Loopback Cable

Using two standard RJ45 connectors, and connect pin1 to pin1, pin 2 to pin 2, etc... for all pins. This loopback is a “straight-through” connection.

A

Index