SunDiag User's Guide

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

SunSoft
A Sun Microsystems, Inc. Business

Contents

Figures

Tables

Preface

About This Book

The SunDiag™ on-line system exerciser runs diagnostic hardware tests. These diagnostic tests can be run from the SunDiag OPEN LOOK® window interface, through serial ports, or individually from shell command lines.

As part of the Solaris® 2.4 for System Administrators AnswerBook®, this manual explains how to set up and run the SunDiag diagnostic tests, as well as how to run the tests individually from command lines. This manual also gives you tips on creating your own diagnostic tests.

The primary audience of this manual is hardware testing and verification organizations. However, SunDiag is simple enough that anyone with Solaris operating environment experience can run it.

How This Book Is Organized

Chapter 1, "Introducing the SunDiag System Exerciser," provides general information about the SunDiag software: how to start the program and what special hardware and software is required for its use.

Chapter 2, "The SunDiag OPEN LOOK Interface," describes startup and exit procedures for the OPEN LOOK user interface. This chapter also explains how to set up test options, which vary according to the hardware installed in the system under test. Instructions for customizing the test sequence and running tests are also provided.

Chapter 3, "The SunDiag TTY Interface," describes the software's complementary TTY interface.

Chapter 4, "Scaling SunDiag Hardware Tests," explains how to "scale" testing for use in multiprocessing systems.

Chapter 5, "Running Individual SunDiag Tests from the Command Line," explains the syntax for running individual SunDiag tests from the command line of a shell, and includes an overview of the procedure for running the SunDiag system exerciser.

Chapter 6, "SunDiag Test Descriptions," contains descriptions of individual SunDiag tests that will work on machines with SPARC® architectures. These descriptions include specific test options, procedures, and error messages.

Appendix A, "Developing Your Own Tests," explains how to use the libraries provided in `/opt/SUNWdiag/lib` and `/opt/SUNWdiag/bin` to develop and create your own tests to use within the SunDiag environment.

Appendix B, "Loopback Connectors," provides information about the serial and parallel port loopback connectors required by some of the SunDiag tests.

Appendix C, "The what_rev Utility," describes `what_rev`, a version control utility designed to determine which files in the SunDiag directory are different from those officially released.

Typographic Conventions

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

Table P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and on-screen computer output (for example, commands, pathnames, test names, and system messages)	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.

Italics also 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 an Ethernet connection you wanted to test was designated by the device named `/dev/le0`, you would replace **`D=`***device_name* with **`D=/dev/le0`**.

Part 1— Running SunDiag

Introducing the SunDiag System Exerciser 1

1.1 Overview

The SunDiag on-line system exerciser runs multiple diagnostic hardware tests from a single interface. It is used primarily with the OpenWindows™ user interface that enables you to set test parameters quickly and easily when running the diagnostic tests.

Support for SunDiag 4.4

This manual describes revision 4.4 of the SunDiag software package, released with the Solaris 2.4 operating environment. The SunDiag 4.4 exerciser is an optional, relocatable software package on the Solaris 2.4 System disk. The default installation directory for SunDiag is `/opt/SUNWdiag/bin`. When you choose to install the SunDiag software, you can change this directory.

Window Interface

SunDiag allows you to select tests and test options with a click of the mouse. See Chapter 2, "The SunDiag OPEN LOOK Interface" for detailed information on the OPEN LOOK interface. You can create your own test environment and save it for future use. SunDiag also features an extensive on-line Magnify Help™ facility to quickly answer questions about the interface. If you have a question about a specific part of the interface (a button, field, or setting), try the on-line help before consulting this manual. You can usually find the answer by pointing to a specific part of the SunDiag window and pressing the Help key.

TTY Interface

Using the TTY interface, you can run the SunDiag software from a terminal or modem attached to a serial port. This feature requires you to type commands instead of using the mouse, and it displays one screen of information at a time. However, it emulates the window system whenever possible; choices are "toggled" by entering a single letter instead of clicking the mouse button.

See Chapter 3, "The SunDiag TTY Interface," for detailed information.

Command Line Interface

You can also run each of the SunDiag tests individually from a shell command line using the command line syntax. Each test description contains the command line syntax to use. See Chapter 5, "Running Individual SunDiag Tests from the Command Line" for a list of standard arguments common to all SunDiag tests.

Hardware Verification

The SunDiag exerciser automatically probes the system kernel for installed hardware devices. Those devices are then displayed on the SunDiag control panel with the appropriate tests and test options. This provides a quick check of your hardware setup. However, you may need to create other tests to verify specific hardware devices (see the "Adding Your Own Tests in .usertest" on page 1-17 and Appendix A, "Developing Your Own Tests.").

The SunDiag exerciser verifies the configuration, functionality, and reliability of most hardware controllers and devices.

1.2 Hardware and Software Requirements

The SunDiag 4.4 software will run on any system installed with the Solaris 2.4 operating environment. The operating system kernel must be configured to support all peripherals that are to be tested.

1.2.1 OpenWindows Software Requirements

You must meet the following three requirements to run SunDiag with the OpenWindows™ software.

1. **OpenWindows Version 3.0**
   You must be running the OpenWindows software, version 3.0, or later.

2. **Display Permission**
   You must have permission to display the SunDiag window on your screen. Typing the following command at a shell prompt will give you the display permission (server access) you need. Substitute the *machinename* variable with the actual name of your workstation.

```
/usr/openwin/bin/xhost + machinename
```

Note – Use the **`/usr/openwin/bin/xhost`** command *before* obtaining root privileges (become superuser). `xhost` will not work in superuser mode.

   This command must be used for every workstation that you intend to display the SunDiag window on.

3. **Correct Library Path**
   You may also have to set the LD_LIBRARY_PATH variable, depending on the location of the OpenWindows directory in your system. If you have installed or mounted OpenWindows files in `/usr/openwin` (the default location), you can ignore this step.

If you have installed the OpenWindows software in a different location, then you must specify where the OpenWindows libraries reside. Use the following command and substitute the *pathname* variable for the actual path where you installed the OpenWindows software:

```
setenv LD_LIBRARY_PATH pathname
```

You can check the existing LD_LIBRARY_PATH by typing **`setenv`**.

1.2.2 Special Note on Testing Multiple Framebuffers

These rules apply when testing multiple framebuffers (displays) simultaneously:

- You can test multiple framebuffers on a system simultaneously, but only one framebuffer can be running OpenWindows software.
- The framebuffer running OpenWindows software must have window locking enabled to avoid incorrect test failures. Other framebuffers must have window locking disabled.

Caution – If window locking is disabled (unlocked) on framebuffers that are running OpenWindows software, the SunDiag tests will return spurious error messages if you move the mouse during testing. Even slight mouse movement can cause a test to fail.

- By default, SunDiag enables window locking on framebuffers that have unit number `0` (for example, `cgsix0`).
  If your system has more than one framebuffer with unit number 0, you must disable window locking on all of them except the one running OpenWindows software.

  If you are running a framebuffer test from a command line, you can disable window locking by specifying the **`L`** argument.

TTY Mode and Framebuffer Window Locking

You can run the SunDiag exerciser in TTY mode in one of three ways: from a terminal attached to a serial port, from a Shell Tool window using the **`-t`** option (see Table 1-3), or on a monitor that is *not* running OpenWindows software.

Table 1-1 Framebuffer Window Locking and TTY Mode

TTY Mode	**Window Locking Issues**
Terminal attached to a serial port	Window locking not necessary because the terminal cannot run OpenWindows software
Shell tool using -t option	Enables window locking on Framebuffer with unit number 0
Monitor *not* running OpenWindows software	The SunDiag software will probe and find framebuffer devices (`cgsix`, for example), but it will not enable window locking on those devices because the OpenWindows software is not running

Warning – Do not attempt to run the TTY mode on the console monitor and framebuffer tests concurrently in this way; doing so causes the framebuffer tests to fail.

1.2.3 Volume Management

Volume Management is a layer of removable media support software that has been added to the Solaris operating environment. This layer manages interactions between users and their removable media and provides transparent access to the media by automatically mounting media with labels.

`rawtest`, `fstest`, and `cdtest` test the diskette and CD-ROM drives regardless of whether the Volume Management software is installed and running. If the Volume Management software is installed, these tests will test the diskette and CD-ROM drives with device names listed in the second column of Table 1-2. If the Volume Management software is not installed, the device names in columns 3 are used.

Table 1-2 Device names used for diskette and CD-ROM drives

Device	**With Volume Manager**	**Without Volume Management**
diskette	`D=/vol/dev/aliases/floppy0`	`D=/dev/diskette`
CD-ROM	`D=/vol/dev/aliases/cdrom0`	`D=/dev/dsk/c0t6d0`

Warning – Do not edit the `/etc/vold.conf` file to change the logical name of the diskette drive (`floppy0`) and CD-ROM drive (`cdrom0`). Currently, SunDiag is hard-wired to use these pathnames as the default logical name.

1.2.4 Booting and New Device Drivers

When adding a new device driver in the Solaris operating environment, you must reboot the machine with the **`boot -r`** command to reconfigure the system and allow the SunDiag exerciser to recognize the new driver.

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.

Starting with Solaris 2.1, boot messages, SCSI error messages and some other debugging messages are no longer sent to the `/var/adm/messages` file. To continue to see these types of messages, make sure to boot your system using the **`boot -v`** (verbose) option.

1.2.5 Swap Space Requirements

The amount of swap space the SunDiag exerciser requires varies widely with individual hardware and software configurations. Most systems have enough swap space already configured to satisfy the SunDiag testing requirements.

When you start testing with the SunDiag system exerciser (by clicking on the Start button), the program will calculate the amount of swap space it will need for testing. The program does this calculation by first determining the amount of swap space available on the system under test, and then calculating how much of this swap space is needed to run the program itself, the various tests, and the virtual memory test.

If there is not enough swap space available on your machine, a pop-up window will display, preventing you from testing. This pop-up window will display how much swap space you will need to add to your system in order to run all of the SunDiag diagnostic tests.

If you do not have enough swap space to run all of the SunDiag tests, you can either deselect some of tests or you can add more swap space to your system. (Refer to the *Solaris 2.4 Administering File Systems* manual for information on increasing swap space.)

Look at the test's option menu to find out how much swap space an individual test will use while testing. The amount of swap space used by the test will be shown in Kilobytes next to the word Configuration. Knowing these swap space amounts will help you decide which SunDiag tests to de-select.

Caution – If your system does not have enough swap space configured, some of the SunDiag tests may run very slowly or they may freeze your screen. In these instances, the SunDiag kernel will usually return error messages indicating that the problem is due to insufficient swap space.

1.2.5.1 Starting SunDiag With the Virtual Memory Test

The SunDiag virtual memory test (`vmem`) is designed to stress test the virtual memory of the system; therefore it uses all of the system's remaining swap space.

Because `vmem` uses all of the system's remaining swap space, you must start all non-SunDiag processes (for example, OpenWindows applications) before you begin to test the system. Starting a process after you have begun testing your system may cause the SunDiag exerciser to run slowly or freeze your screen.

If you plan on starting other processes after you have started the virtual memory test, you must use the `vmem` reserve option to leave space for these processes before you start testing.

Reserve Option to `vmem`

The `vmem` Reserve option allows you to reserve additional amounts of swap space for other processes or tests. This option can be exercised from either the `vmem` test options menu (within the SunDiag control panel) or from the command line.

1.2.6 Setting the Maximum Number of Processes

The SunDiag system exerciser runs under the Solaris 2.4 operating environment. The SunDiag kernel controls hardware specific tests from one interface. Each test displayed on the Status Panel is a separate UNIX® process, independent from the SunDiag kernel and other processes.

As a system's configuration grows, the number of processes needed for SunDiag testing also grows (especially on multi-processor systems). If the number of processes needed by SunDiag tests exceeds the maximum number of processes allowed by your system, the SunDiag application will fail.

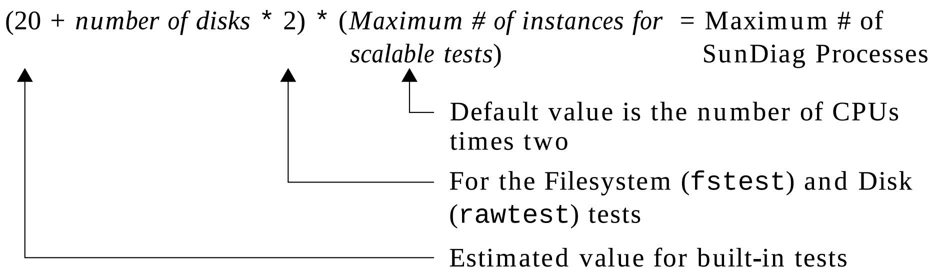
However, you can change the maximum number of processes on your system by setting the `maxusers` parameter in your `/etc/system` file. The following section describes how to set this parameter to allow for all SunDiag processes.

1.2.6.1 *Setting the* `maxusers` *Parameter in the* `/etc/system` *file*

Beginning with the Solaris 2.3 operating environment, the default value of `maxusers` will be set automatically based on the amount of RAM installed on the system. For example, if your system has 32 Mbytes of RAM, the kernel will set the `maxusers` to 32.

The maximum number of UNIX processes allowed on a system is 16 times the value of `maxusers` (16*`maxusers`). Therefore, the maximum number or processes allowed on a system with 32 Mbytes of RAM is 512 (16*32).

However, the default number of UNIX processes set by a system may be insufficient to run all of the SunDiag tests. The number of test processes that the SunDiag exerciser may create while testing is approximately:


(20 + *number of disks* * 2) * (*Maximum # of instances for scalable tests*) = Maximum # of SunDiag Processes
Default value is the number of CPUs times two
For the Filesystem (`fstest`) and Disk (`rawtest`) tests
Estimated value for built-in tests

For example, the SunDiag exerciser will generate 1008 processes on a SPARCcenter™ 2000 system with 6 CPUs and 32 disks [(20 + 32*2)*(6*2)]. Therefore, this system will conservatively need 1200 processes (including system, network, and window system processes) to run without problems, especially if you have set the Concurrent Tests # to a very high value (see "Concurrent Tests #" on page 2-21). With 1200 processes needed, the `maxusers` parameter should be set to 75 (`maxusers` = (1200 / 16) = 75).

If this machine has more than 75 MBytes of RAM, then there is no need to adjust the default value of `maxusers`. Otherwise, in this example, the following line must be added to the `/etc/system` file:

```
set maxusers=75
```

Refer to the *Solaris 2.4 Transition Guide* and the *SunOS 5.4 Administering Security, Performance, and Accounting* manual for more information on setting system configuration parameters.

1.2.7 Loopback Connectors

Certain SunDiag tests require loopback plugs or cables to run successfully. See the individual test descriptions to find out which tests need loopback cables or plugs. Also see Appendix B, "Loopback Connectors" for directions on how to obtain loopback plugs or cables.

1.2.8 Scratch CDs, Tapes, Diskettes

SunDiag requires that "scratch" tapes, diskettes, or optical discs (CDs) be installed in Sun's tape and disk drives for tests to run correctly. Scratch media are spare (usually blank) tapes and diskettes that can be overwritten as well as optical discs. These scratch media devices must be inserted before the kernel is probed by SunDiag, or else SunDiag will report an error message.

For CD-ROM tests, a test CD with a well-known table of contents must be loaded in the CD drive. It is recommended you use a demonstration CD shipped with the drive. Do not use an operating system distribution disc.

For tape tests, you need 4mm, 8mm, $^1/_2$", or $^1/_4$" scratch tapes (depending on the type of drive being tested). Make sure the tape heads have been properly cleaned.

For hard disk and diskette tests, be sure there is enough space on your disk partition. Double or triple density diskettes (1.4 Mbyte) are required, depending on the diskette drive in your system. An additional megabyte of swap space is needed to run `fstest`.

Note – Beware of using old or scratched tapes and diskettes; they could cause spurious errors in specific tests.

1.2.9 TTY Terminals

SunDiag is designed to run in a window environment, but may be used from a terminal attached to a serial port. Chapter 3, "The SunDiag TTY Interface" describes the user interface when running in TTY mode.

1.3 Preparing to Start the SunDiag Exerciser

You must have server access (display) permission on the system under test to display the SunDiag window. Typing **`/usr/openwin/bin/xhost +`** *displayhost* will give you display permission to run SunDiag on *displayhost*. Remember that you must run this command *before* you become superuser.

You must be logged in as superuser (root) to run the program. The SunDiag software needs to write log and error files to the `/var/adm/sundiaglog` directory, which should be owned by root.

Note – While the SunDiag software is running, you should not run other programs or software that use the same hardware devices that the program is testing. In particular, when the virtual memory test is running, there may not be enough free memory to run *any* other programs. However, the Reserve option in the virtual memory (`vmem`) menu lets you specify the amount of memory to reserve from being tested. This option allows you to free up memory to run another application while running the SunDiag software.

If you log in remotely (`rlogin`) or log in from a serial port, the SunDiag application will automatically run in TTY mode. See Chapter 3, “The SunDiag TTY Interface.”

The syntax for using the `sundiag` command is shown below. You will find it helpful to read Chapter 3, “The SunDiag TTY Interface” before actually starting the program in TTY mode.

Note – The SunDiag exerciser enables all available tests when it is invoked (except those that require intervention mode). Starting all the enabled tests will slow your system down drastically. Be sure to read the “Hardware and Software Requirements” section in this chapter before running all available tests.

1.4 Starting the SunDiag Exerciser

Read through the argument descriptions listed in Table 1-3 before actually starting the SunDiag exerciser. If you don't need any of those options, just type these basic commands:

```
example% xhost + machinename
 machinename being added to access control list
example% su
Password: (enter your root password)
example# /opt/SUNWdiag/bin/sundiag
SunDiag: Starting probing routine, please wait...
```

(Next, you may see some error messages if you haven't inserted scratch tapes, disks, or CDs in your system. The SunDiag window then displays on your screen.)

If you are running OpenWindows software, the SunDiag window will display by default; otherwise, SunDiag will display in TTY mode (See Chapter 3, "The SunDiag TTY Interface" for more details). Table 1-3 shows the full syntax for starting the SunDiag exerciser.

Table 1-3 SunDiag Syntax

/opt/SUNWdiag/bin/sundiag [-Cpqtw] [-i *number***] [-o** *options_file***] [-b** *batch_file***] [-k** *kernel_name***]**

Argument	Description
-C	Redirects the console output from any existing console window to the SunDiag console window. If you are using the TTY interface, the console message is displayed in the message line of the status screen.
-p	Tells SunDiag to ignore the kernel probe for devices. Use this when running user-defined tests found in `.usertest`.
-q	Automatically quits the SunDiag program when testing stops. This option can only be issued from a command line.
-t	Instructs SunDiag to run in TTY mode. See Chapter 3, "The SunDiag TTY Interface" for information on running SunDiag in this mode.

Table 1-3 SunDiag Syntax *(Continued)*

/opt/SUNWdiag/bin/sundiag [-Cpqtw] [-i *number***] [-o** *options_file***]**
[-b *batch_file***] [-k** *kernel_name***]**

Argument	Description
-w	Writes the system hardware configuration to the `/var/adm/sundiaglog/sundiag.conf` file.
-i *number*	Specifies the maximum number of instances for scalable tests. This setting overrides the default setting of two times the number of processors on the system under test.
-o *options_file*	Directs the SunDiag software to use a specific “saved option” file. See the “Option Files Window Button” section in Chapter 2 for directions on creating an options file. If you do not use the **-o** argument, SunDiag uses the default option file `/var/adm/sundiaglog/options/.sundiag`, if it exists.
-b *batch_file*	Enables you to use a *batch_file* (collection of option files) to specify testing parameters when running the SunDiag software. See the “Using Batch Files” section in this chapter for more details.
-k *kernel_name*	Specifies a customized kernel name. The default kernel name is `/kernel/unix`. Since the `rstatd` that the performance monitor requires is hard-wired to use `/vmunix` as kernel name, the performance monitor is disabled when the **-k** option is specified.

1.5 Stopping the SunDiag Exerciser

To exit SunDiag from the OpenWindows interface, click the Stop button at the top of the control panel to stop any tests that are running. Some of the tests, such as the tape tests, may delay before actually stopping, because these tests require time to rewind the tapes.

To exit SunDiag from a terminal or remote login, type **q** for `quit`.

1.6 Using Batch Files

Batch files are lists of option files which specify SunDiag tests and their options. To use the SunDiag **-b** option, you must first create a `batch_file` in `/var/adm/sundiaglog/configs` before invoking SunDiag. The `batch_file` must use the following format:

```
#option file runtime delay_before_loading_next_option_file (min.)
#----------- ------- ------------------------------------
optfile1     60      3
optfile2     1020    5
optfile1     60      16
optfile3     0       0
#
```

In the example above, `optfile1` and `optfile2` are created using the Option Files menu button on the SunDiag control panel (See Section 2.1.4, “Control Panel”). These files list the SunDiag tests to be run. They run for the times specified (in minutes) in the `runtime` column.

Files with a `runtime` of `0` display the final status of tests that have already run. This feature can be used to give the status of some or all of the option files in the batch file. For instance, the example file above first runs `optfile1` and `optfile2`. Assume that `optfile3` is a concatenation of `optfile1` and `optfile2`. Running `optfile3` with a runtime of `0` will return the final status of `optfile1` and `optfile2`.

The `delay_before_loading_next_option_file` field ensures that all tests have been stopped before the next option file is loaded. SunDiag reserves enough time to ensure a smooth transition between tests, even if the delay specified in this field is not long enough.

The settings in the SunDiag Options menu override those in batch files. For example, if the value for `Max # of passes` is reached before the runtime set in the batch file is over, the test will stop. For this reason, you should set large values on the Set Options menu when using batch files and avoid using the `Single Pass` values.

Note – The SunDiag schedule option does not work with the batch option.

1.7 Running the SunDiag Exerciser on a Remote System

Use the following commands to run the SunDiag exerciser remotely with the OpenWindows interface. In the cases below, you must be logged in as superuser (root) to run the program. The SunDiag exerciser must be able to write the log and error files to the `/var/adm/sundiaglog` directory, which should be owned by root.

Note – To get the OpenWindows display on a remote machine, be certain to open the permissions of that window server using the **`/usr/openwin/bin/xhost +`** *hostname* command.

On a Local System

Use one of the following commands to run the SunDiag exerciser with the OpenWindows interface from a local system:

`#` **`./sundiag`**	If OpenWindows software is running, the Sundiag OpenWindows interface displays. Otherwise, SunDiag starts in TTY mode.
`#` **`./sundiag -display`** *remotehost*:**0**	If OpenWindows software is running on the remote system, the Sundiag OpenWindows interface will display on the remote system. Otherwise an error message is returned: `XView Error: Cannot open display on` *window_server*:*remotehost*:`0`

From a Remote System

Use one of the following commands to run the SunDiag exerciser with the OpenWindows interface from a remote machine. These cases assume that you have used `rlogin` to log in as superuser on *remotehost*, and that you want the SunDiag OpenWindows interface to appear on your local system:

`# ./sundiag`	Sundiag TTY mode will display on your local window/console. See Chapter 3; this is the default mode.
`# ./sundiag -display` *localhost*`:0`	If OpenWindows software is running on the local system, the Sundiag OpenWindows interface displays on the localhost. Otherwise, an error message is returned: `XView Error: Cannot Open display on` *window_server*: *localhost*`:0`
`# ./sundiag -display` *remotehost*`:0`	If OpenWindows is running on the remote system, the Sundiag OpenWindows interface displays on the remote server. Otherwise, an error message is returned: `XView Error: Cannot open display on` *window_server*: *remotehost*`:0`

1.8 Running the SunDiag Exerciser on a Stand-alone System

Some SunDiag applications require a network connection to run successfully. However, you can run the SunDiag exerciser “stand-alone” (without a network connection) by following these steps.

1. **Edit the `/etc/rc2.d/S72inetsvc` script file, and comment out the following line:**

```
# /usr/sbin/ifconfig -au netmask + broadcast +
```

2. **Comment out all remote mount file systems from `/etc/vfstab` (since there will be no network connection).**

3. **Make sure `ypbind` is not running in the stand-alone system.**
   This step will make sure that `ypbind` is not started by any of the `rc` scripts.

4. **Reboot your system.**
   The SunDiag software will now run properly on stand-alone machines.

1.9 Adding Your Own Tests in `.usertest`

You can add your own tests and have them appear as options on the SunDiag interface. Any file that can be run from a UNIX® shell can be used as a test file (including shell scripts).

To run your test through the SunDiag interface, you need to put the test file in the `/opt/SUNWdiag/bin` directory, and create a new `/opt/SUNWdiag/bin/.usertest` file using the syntax below.

1.9.1 Setting up a `.usertest` file

Use the following syntax to set up a `.usertest` file:

```
device_name_label, testname, test_specific_arguments, SCA
```

Argument	Description
device_name_label	*device_name_label* is the device name to be displayed for your test on the control panel.
testname	*testname* is the actual name of the file that contains your test (for example, `vmem` for the virtual memory test). It is used as the test name on the status panel.
test_specific_arguments	These are the optional test-specific command-line arguments you use to execute your test. Since commas are used to delimit arguments in a `.usertest` line, use spaces to delimit the test-specific arguments instead. See the individual test descriptions in “SunDiag Tests” for the test-specific arguments.
`SCA`	This optional flag specifies that the test will be scalable. See “Scalable Tests” on page 4-1 for a list of scalable tests. Not all are tests are designed to be scalable.

The options you set in the `.usertest` file will become the default options for each test. You can change these options using the pop-up option menus on the SunDiag window interface, but the Reset button will return the options to the default settings specified in the `.usertest` file.

The following is an example of a `.usertest` file. SunDiag will ignore lines that are commented out (lines that begin with #).

```
# @(#).usertest Rev   MM/DD/YY

SPC/S Internal Test,newtest,D=any T=1
SPC/S 25-pin LB on ttyz00,newtest,D=/dev/ttyz00 T=8
#SPC/S 25-pin LB on ttyz01,newtest,D=/dev/ttyz01 T=8
#SPC/S 25-pin LB on ttyz02,newtest,D=/dev/ttyz02 T=8
#SPC/S 25-pin LB on ttyz03,newtest,D=/dev/ttyz03 T=8
#SPC/S 25-pin LB on ttyz04,newtest,D=/dev/ttyz04 T=8
#SPC/S 25-pin LB on ttyz05,newtest,D=/dev/ttyz05 T=8
#SPC/S 25-pin LB on ttyz06,newtest,D=/dev/ttyz06 T=8
#SPC/S 25-pin LB on ttyz07,newtest,D=/dev/ttyz07 T=8
SPC/S Echo TTY on ttyz00,newtest,D=/dev/ttyz00 T=16
SPC/S 96-pin LB on board 1,newtest,D=sb1 T=4
SPC/S 96-pin LB on board 2,newtest,D=sb2 T=4
SPC/S 96-pin LB on board 3,newtest,D=sb3 T=4
```

Figure 1-1 Example of a `.usertest` File

1.9.2 Test Writing Precautions

When writing tests for the SunDiag program, you should note the following precautions: (For more thorough instructions on developing your own tests, see Appendix A, "Developing Your Own Tests.")

- The file descriptor, `stdin`, will be closed, and `stdout` and `stderr` will be redirected to `/dev/null`.

  Therefore, there will be no direct interaction between the user and the test. Tests should be written to use other methods of logging messages (for example, you can program your tests to write to output files).

- When the test exits with a status code of `0`, SunDiag will count this exit as a pass.

  Status counts of 1 through 90 are indications of errors and the error count will increment. The other status codes are reserved by SunDiag and should not be used.

- If the test requires a large amount of memory to run (greater than 1 MByte), you should use the `reserve` option of the `vmem` test to set aside extra memory for the test.

See Section 6.2, “Virtual Memory Test (vmem)” for instructions.

1.10 SunDiag Exit Status Codes

SunDiag will exit with the following status codes:

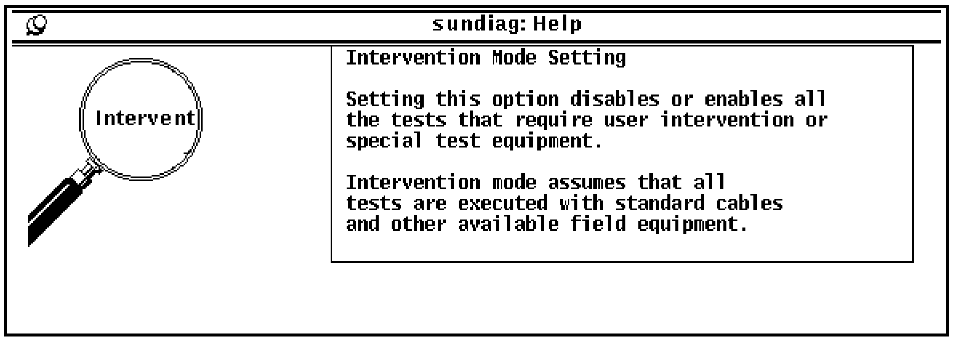
Table 1-4 SunDiag Exit Status Codes

Code	Description
`0`	The SunDiag kernel exited normally and the tests that were run (if any) all completed successfully.
`-1`	The SunDiag kernel exited due to an abnormal condition in the execution of its kernel.
`1`	The SunDiag kernel exited normally but one or more of the tests that were run, failed.

The SunDiag OPEN LOOK Interface 2

This chapter explains how to run the SunDiag system exerciser using the OPEN LOOK Graphical User Interface. This is the interface that appears if the SunDiag software is started from within the OpenWindows environment (a shell tool, for example).

Remember, Magnify Help™ is available for all items (buttons, settings, panels, etc.) in the SunDiag window. You can view the Magnify Help window by pointing to an item, and then pressing the Help key. A pop-up window with a magnifying glass will display, giving more information about the item.



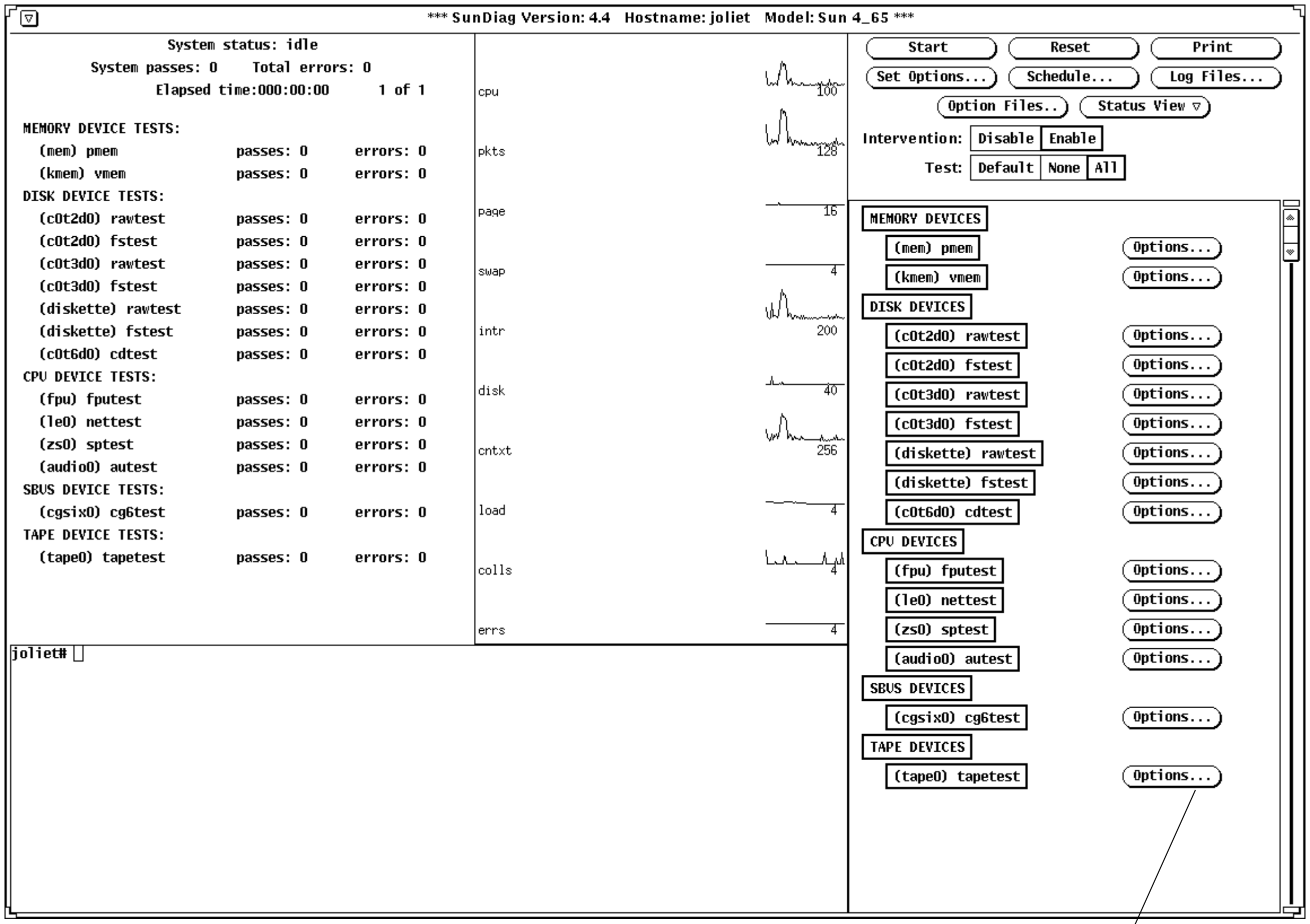
```
Intervention Mode Setting

Setting this option disables or enables all
the tests that require user intervention or
special test equipment.

Intervention mode assumes that all
tests are executed with standard cables
and other available field equipment.
```

Figure 2-1 Magnify Help Window

System Status Panel
This panel shows testing status. Asterisks appear next to tests that are currently in progress.

Performance Monitor Panel
These graphs show the same statistics as the Solaris Performance Meter program.

Control Panel
This is the main control panel for the SunDiag interface.



```
                  System status: idle
          System passes: 0    Total errors: 0
                 Elapsed time:000:00:00      1 of 1

 MEMORY DEVICE TESTS:
   (mem) pmem              passes: 0      errors: 0
   (kmem) vmem             passes: 0      errors: 0
 DISK DEVICE TESTS:
   (c0t2d0) rawtest        passes: 0      errors: 0
   (c0t2d0) fstest         passes: 0      errors: 0
   (c0t3d0) rawtest        passes: 0      errors: 0
   (c0t3d0) fstest         passes: 0      errors: 0
   (diskette) rawtest      passes: 0      errors: 0
   (diskette) fstest       passes: 0      errors: 0
   (c0t6d0) cdtest         passes: 0      errors: 0
 CPU DEVICE TESTS:
   (fpu) fputest           passes: 0      errors: 0
   (le0) nettest           passes: 0      errors: 0
   (zs0) sptest            passes: 0      errors: 0
   (audio0) autest         passes: 0      errors: 0
 SBUS DEVICE TESTS:
   (cgsix0) cg6test        passes: 0      errors: 0
 TAPE DEVICE TESTS:
   (tape0) tapetest        passes: 0      errors: 0
```

```
joliet#
```

Console Window
Messages appear in the console window. You can also execute Solaris Operating Environment commands as superuser from this window.

Option Menu Buttons
Each individual test has an option menu button for setting various options.

Figure 2-2 SunDiag OPEN LOOK Interface Main Window

2.1 SunDiag Main Window

When you start the SunDiag system exerciser, an OPEN LOOK window is displayed, as shown in Figure 2-2. The window is divided into four smaller sections:

- A *system status panel* that shows testing status.
- A *performance monitor panel* that displays performance statistics for the system under test.
- A *control panel* with buttons, settings, and window buttons for changing test parameters and options.
- A *console window* that displays operating system messages and test message. As superuser, you can execute operating system commands from this window.

The Options menu buttons are also noted in Figure 2-2; they are explained in the “Options Menu Buttons” section later in this chapter.

2.1.1 System Status Panel

The System Status panel is located in the upper left of the SunDiag main window, as shown in Figure 2-3. This window shows you know how the tests are progressing. The top of the system status panel shows:

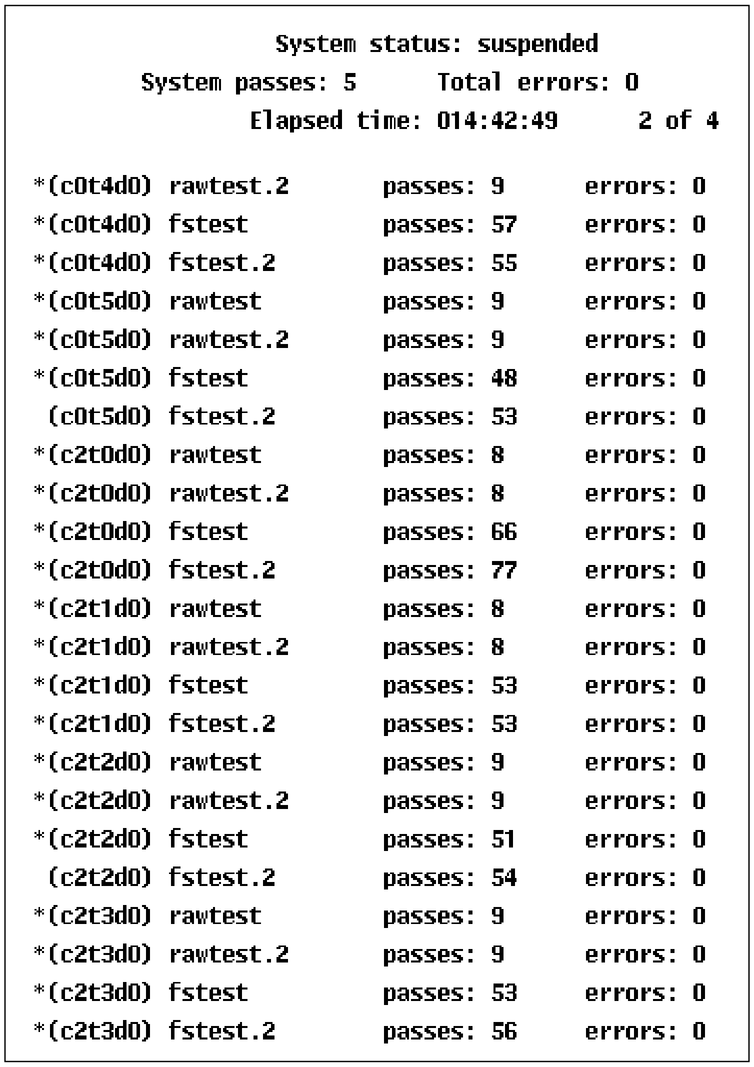
- Current system status (idle, testing, or stopped)
- Number of successful system passes
- Total number of errors
- Elapsed time that testing has taken
- Page number in the current view (for example, 1 of 4)

You can reset all the values in these fields to zero by clicking the Reset button on the control panel.

The lower portion of the panel shows the number of successful passes and errors for each of the tests being run. The tests are grouped by device type, and an active test is marked with an asterisk. Use the Status View button to manipulate this list. (See Section 2.2.4.4, “Status View Button.”)

The results displayed in this panel are stored in special log files. See the “Log Files Window Button” section in this chapter for details.

The “System passes:” field always shows the least number of passes that the longest test has completed. For example, if the pmem test has completed two passes while all other tests have completed four passes, the system status panel shows two system passes.



```
                   System status: suspended
          System passes: 5      Total errors: 0
                 Elapsed time: 014:42:49      2 of 4

 *(c0t4d0) rawtest.2       passes: 9      errors: 0
 *(c0t4d0) fstest          passes: 57     errors: 0
 *(c0t4d0) fstest.2        passes: 55     errors: 0
 *(c0t5d0) rawtest         passes: 9      errors: 0
 *(c0t5d0) rawtest.2       passes: 9      errors: 0
 *(c0t5d0) fstest          passes: 48     errors: 0
  (c0t5d0) fstest.2        passes: 53     errors: 0
 *(c2t0d0) rawtest         passes: 8      errors: 0
 *(c2t0d0) rawtest.2       passes: 8      errors: 0
 *(c2t0d0) fstest          passes: 66     errors: 0
 *(c2t0d0) fstest.2        passes: 77     errors: 0
 *(c2t1d0) rawtest         passes: 8      errors: 0
 *(c2t1d0) rawtest.2       passes: 8      errors: 0
 *(c2t1d0) fstest          passes: 53     errors: 0
 *(c2t1d0) fstest.2        passes: 53     errors: 0
 *(c2t2d0) rawtest         passes: 9      errors: 0
 *(c2t2d0) rawtest.2       passes: 9      errors: 0
 *(c2t2d0) fstest          passes: 51     errors: 0
  (c2t2d0) fstest.2        passes: 54     errors: 0
 *(c2t3d0) rawtest         passes: 9      errors: 0
 *(c2t3d0) rawtest.2       passes: 9      errors: 0
 *(c2t3d0) fstest          passes: 53     errors: 0
 *(c2t3d0) fstest.2        passes: 56     errors: 0
```

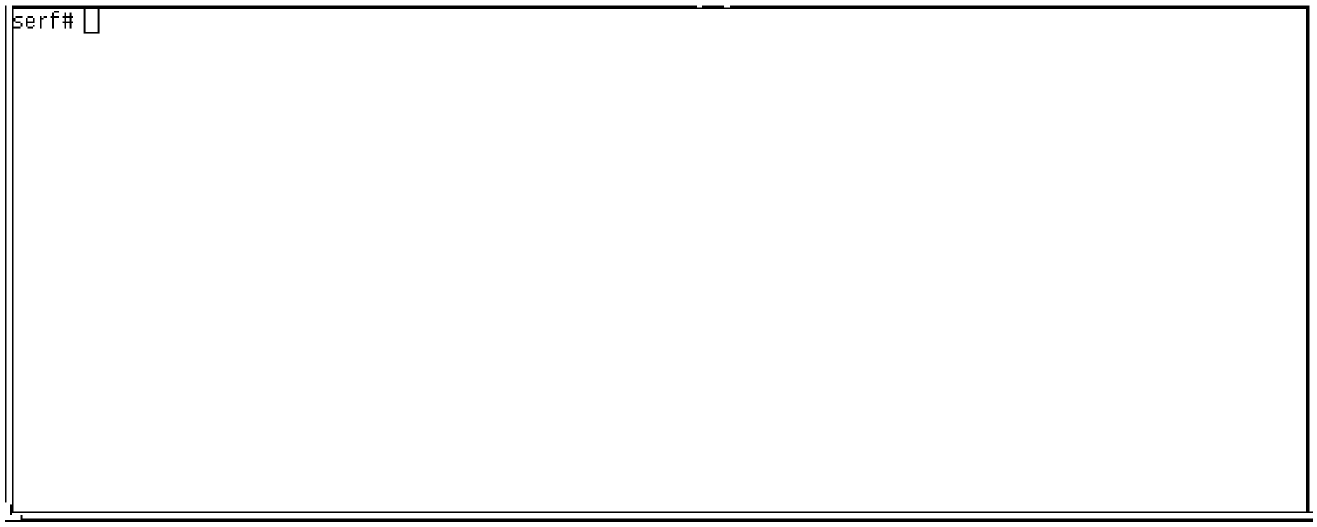
Figure 2-3 Sample System Status Panel

2.1.2 Console Window

The console window is displayed in the lower left corner of the SunDiag main window. This console window is a standard Solaris console window; you can enter commands from this window as superuser and display the results.

Using the `-c` option when starting the SunDiag program redirects system messages from your system console window to the SunDiag console window.

Note – Do not start another Solaris console window after starting the SunDiag program. If you do, the SunDiag window will stop receiving system console messages. The SunDiag console will continue to receive SunDiag messages, but not system console messages. Even if the new console window is closed later, the SunDiag console window will not receive console messages.



```
serf#
```

Figure 2-4 Sample Console window

2.1.3 Performance Monitor Panel

The performance monitor panel is the same as that provided by the OpenWindows Performance Meter Deskset program; it provides a graphic display of system performance statistics.

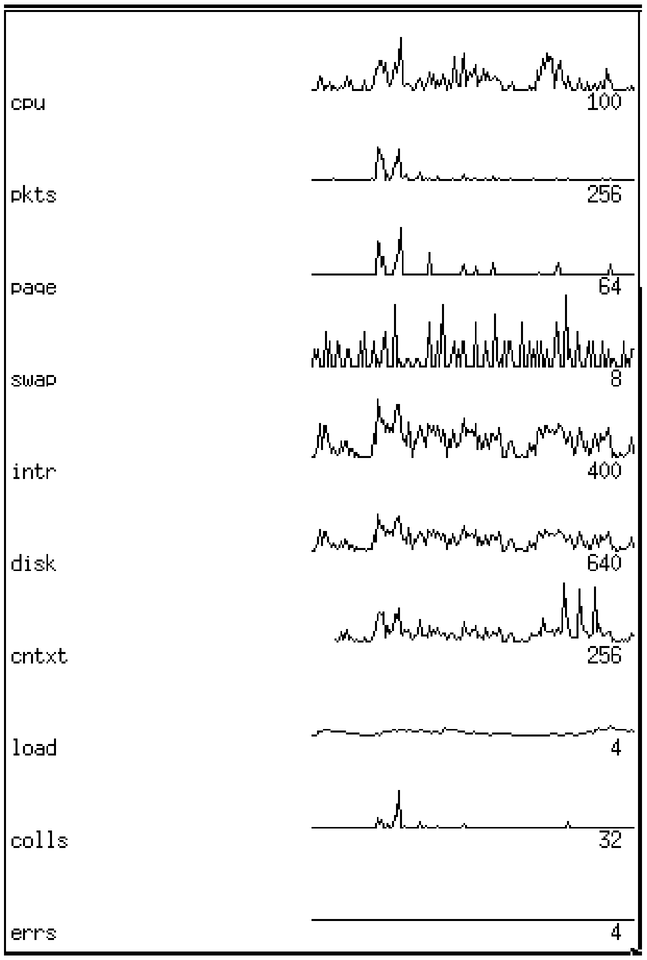


Figure 2-5 Performance Monitor Panel

The following table describes what each graph on the performance monitor panel represents:

Graph	Description
`cpu`	Percent of CPU being utilized
`pkts`	Ethernet packets per second
`page`	Paging activity in pages per second
`swap`	Jobs swapped per second
`intr`	Number of device interrupts per second
`disk`	Disk traffic in transfers per second
`cntxt`	Number of context switches per second
`load`	Average number of runnable processes over the last minute
`colls`	Collisions per second detected on the Ethernet
`errs`	Errors per second on receiving packets

2.1.4 Control Panel

The contents of the lower part of the control panel are determined by your system configuration. The SunDiag software automatically probes for available devices and displays them in this part of the control panel. Figure 2-6 on page 2-9 shows the control panel for a generic SPARCstation™ system that contains:

- Physical and virtual memory
- Two SCSI disk drives
- Floppy disk drive
- Floating point accelerator
- Ethernet connection
- Serial ports
- Audio port
- Color graphics accelerator
- Special parallel port

When this window is displayed, check the devices listed in it against the devices you know are installed on your system. If there are any discrepancies, check the boot-up information in the `/var/adm/messages` file to make sure that the operating environment sees all of your hardware devices.

If the operating system does not recognize a hardware device that you know is attached to your system, check the hardware and the Solaris kernel configuration.

Note – When adding a new device driver in the Solaris operating environment, you must reboot the machine with the `boot -r` command before the SunDiag software will recognize the new driver.

When the software configuration agrees with the hardware configuration, you can begin to select settings for the devices you want to test.

In Figure 2-6, the tests shown with bold frames are enabled for testing.

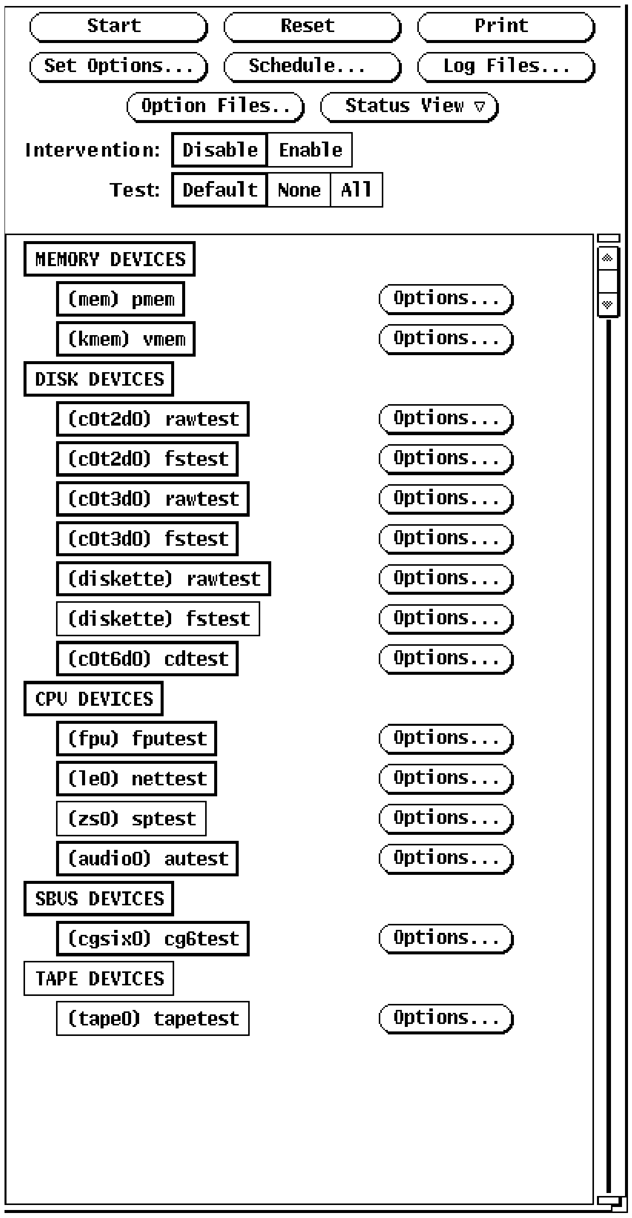
2.1.4.1 Selecting Options and Clicking Buttons

Move the pointer to a button and click SELECT. Click buttons to perform actions such as Start / Stop and Reset / Suspend / Resume. Click window buttons (such as the Set Options and Log Files window buttons) to display a pop-up menu that contains additional controls.

Clicking SELECT on a setting chooses that setting. Selected settings are displayed with bold frames on monochrome monitors. On color monitors, selected settings appear shaded, as if they are "pushed in."

2.1.4.2 Dimmed Buttons

Some of the window buttons appear to be dimmed once testing has started. These buttons are not available until testing has stopped.



Intervention: [✓ Disable] Enable

Test: [✓ Default] None All

MEMORY DEVICES
(mem) pmem Options...
(kmem) vmem Options...
DISK DEVICES
(c0t2d0) rawtest Options...
(c0t2d0) fstest Options...
(c0t3d0) rawtest Options...
(c0t3d0) fstest Options...
(diskette) rawtest Options...
(diskette) fstest Options...
(c0t6d0) cdtest Options...
CPU DEVICES
(fpu) fputest Options...
(le0) nettest Options...
(zs0) sptest Options...
(audio0) autest Options...
SBUS DEVICES
(cgsix0) cg6test Options...
TAPE DEVICES
(tape0) tapetest Options...

Figure 2-6 SunDiag Control Panel

2.2 Setting SunDiag Options

The following sections describe the various buttons and settings that appear in the top part of the control panel.

2.2.1 Start/Stop Button

When you click the Start button, the SunDiag exerciser starts all enabled tests. Once testing has started, the wording on the Start button changes to Stop. Clicking Stop will stop testing.

Note – Click Stop only once. This button does not change back to Start until all tests have stopped, and some tests do not stop immediately.

The System status item on the system status panel changes from “idle” to “testing” once testing has started. When you stop testing, the system status changes to “stopping,” then “idle.”

2.2.2 Reset/Suspend/Resume Button

The Reset button resets the passes and error count for each test in the system status panel to zero. It also resets system passes, total errors, and elapsed time to zero. This button works only when testing has stopped.

Once tests have started, this button becomes Suspend. If you click Suspend, System status in the system status panel changes to “Suspended,” and the Suspend button changes to Resume. When you are ready to resume testing, click the Resume button. After testing has stopped, the button changes to Reset.

2.2.3 Print Button

Click the Print button to take a snapshot of the current screen and print the file to the printer specified in the Set Options menu.

2.2.4 Log Files Window Button

The SunDiag system exerciser saves the status of its progress in three log files. These files contain Error, Information, and UNIX Messages. You can access these files by clicking the Log Files window button.

Warning – Do not attempt to edit message files while the SunDiag exerciser is running. You may only display, print, or remove these files. Editing these files while SunDiag exerciser is running causes the software to crash.

With the resulting pop-up menu you can to display, remove, or print each of those log files.

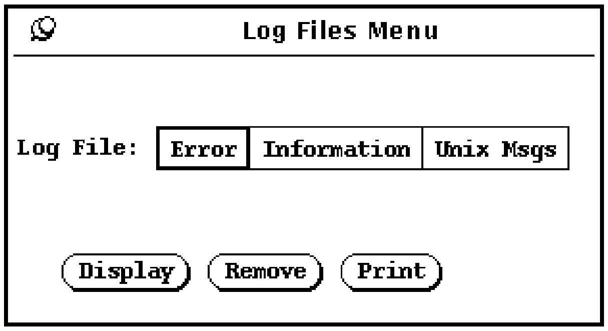


Figure 2-7 Log Files Window

The three log files in this menu are:

`/var/adm/sundiaglog/sundiag.err`	SunDiag's Error Status Log
`/var/adm/sundiaglog/sundiag.info`	SunDiag's Information Log
`/tmp/unix.msg`	Solaris System Message Log

The `sundiag.err` file contains SunDiag test error messages and start/stop times. The status log file `sundiag.info` contains informative SunDiag and probe messages generated while starting and stopping the SunDiag program. The `/tmp/unix.msg` file is a concatenation of general UNIX messages in the `/var/adm/message*` files.

Note – Starting with the Solaris 2.1 operating environment, boot messages, SCSI error messages and some other debugging messages are no longer sent to the `/var/adm/messages` file. To continue to see these types of messages, make sure to boot your system using the **`boot -v`** (verbose) option.

Displaying Logs

You can display any of the three log files by selecting the log file and then clicking the display button.

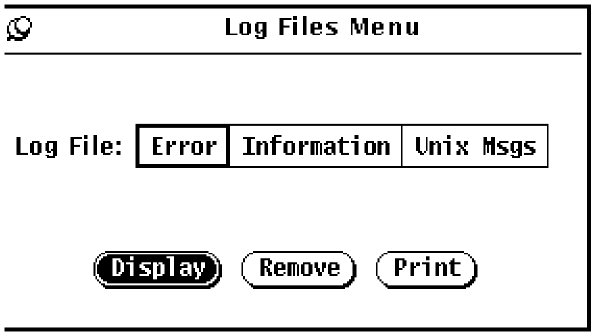
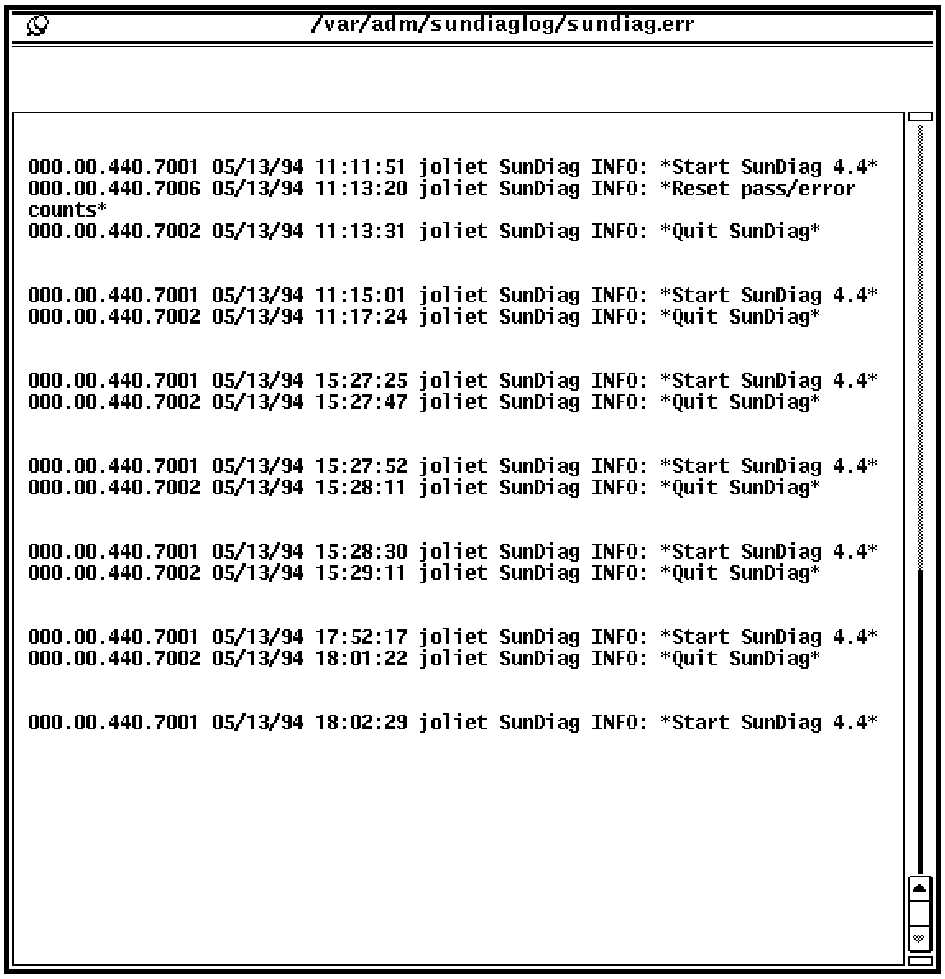


Figure 2-8 Displaying the Error Log File

A pop-up test window will display the selected file (see Figure 2-9). Note that the log file name is printed at the top of the window.



```
000.00.440.7001 05/13/94 11:11:51 joliet SunDiag INFO: *Start SunDiag 4.4*
000.00.440.7006 05/13/94 11:13:20 joliet SunDiag INFO: *Reset pass/error
counts*
000.00.440.7002 05/13/94 11:13:31 joliet SunDiag INFO: *Quit SunDiag*


000.00.440.7001 05/13/94 11:15:01 joliet SunDiag INFO: *Start SunDiag 4.4*
000.00.440.7002 05/13/94 11:17:24 joliet SunDiag INFO: *Quit SunDiag*


000.00.440.7001 05/13/94 15:27:25 joliet SunDiag INFO: *Start SunDiag 4.4*
000.00.440.7002 05/13/94 15:27:47 joliet SunDiag INFO: *Quit SunDiag*


000.00.440.7001 05/13/94 15:27:52 joliet SunDiag INFO: *Start SunDiag 4.4*
000.00.440.7002 05/13/94 15:28:11 joliet SunDiag INFO: *Quit SunDiag*


000.00.440.7001 05/13/94 15:28:30 joliet SunDiag INFO: *Start SunDiag 4.4*
000.00.440.7002 05/13/94 15:29:11 joliet SunDiag INFO: *Quit SunDiag*


000.00.440.7001 05/13/94 17:52:17 joliet SunDiag INFO: *Start SunDiag 4.4*
000.00.440.7002 05/13/94 18:01:22 joliet SunDiag INFO: *Quit SunDiag*


000.00.440.7001 05/13/94 18:02:29 joliet SunDiag INFO: *Start SunDiag 4.4*
```

Figure 2-9 Error Log File

Log File Test Message Syntax

`TTT.VV.SSS.EEEE` *date time devicename testname* `[ERROR|FATAL|INFO|WARNING]` *message*

Table 2-1 Log File Test Message Syntax

Arguments	Description
`TTT`	The test identifier.
`VV`	The two digit test version identifier.
`SSS`	The subtest identifier.
`EEEE`	The error identifier; the first digit is the priority identifier (message type) and the last three digits are the message identifier. The priority identifier number is one of the following:
	`9: ERROR`
	`7: INFO`
	`5: WARNING`
	`3: FATAL`
date time	Tells when the error occurred.
devicename	The device under test where the error occurred.
testname	The name of the test reporting the error.
message	This argument varies considerably, depending on the type of test running when the error occurred.

A typical error message generated by a test looks like this:

```
009.33.999.9025 12/13/89 18:54:49 xy0 fstest ERROR:
not enough trace blocks on xy0
```

Refer to Table 5-2 on page 5-3 for a list of tests and their corresponding identification numbers.

The format of SunDiag kernel messages is:

```
TTT.VV.SSS.EEEE date time hostname sundiag [ERROR|FATAL|INFO|WARNING]:
message
```

When the SunDiag kernel probes for devices at power-up, a message is displayed if any problems are found. The format is:

```
TTT.VV.SSS.EEEE date time hostname probe [ERROR|FATAL|INFO|WARNING]
```

Removing Log Files

Click Remove on the Log Files window to truncate the selected log file (remove all but the most recent few entries of the log file).

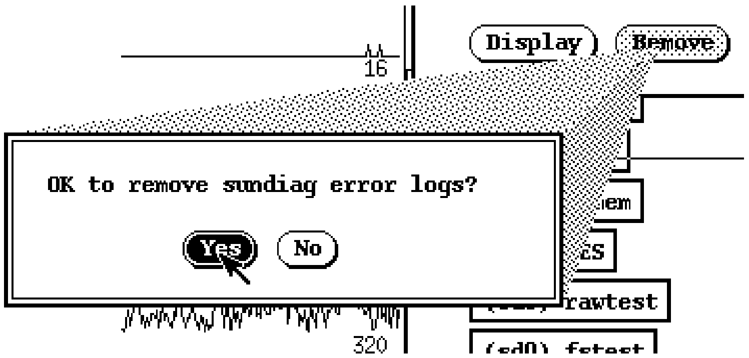

OK to remove sundiag error logs?

Figure 2-10 Confirming Error Log Removal

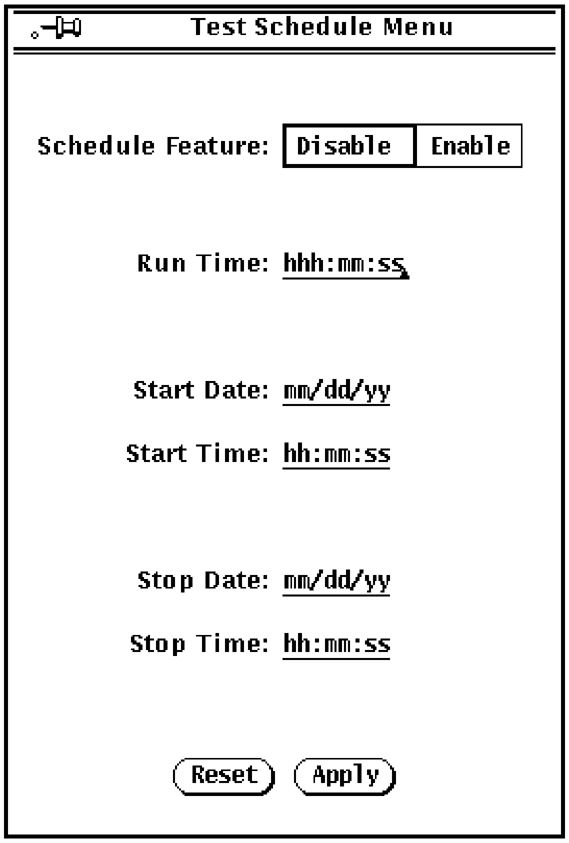
Printing Error Logs

Click Print to print the entire log file to your default printer.

Note – These logs can be very long. Make sure you want the entire file before printing it.

2.2.4.1 Schedule Window Button

Click the Schedule window button to display a pop-up menu like the one shown in Figure 2-11. The test schedule menu allows you to set start and stop dates and times to automatically run the SunDiag exerciser.


Test Schedule Menu

Schedule Feature: Disable Enable

Run Time: hhh:mm:ss

Start Date: mm/dd/yy

Start Time: hh:mm:ss

Stop Date: mm/dd/yy

Stop Time: hh:mm:ss

Reset Apply

Figure 2-11 Test Schedule Menu

▼ How to Use SunDiag's Schedule Feature

1. **Click Enable.**
   This setting needs to be enabled for the schedule feature to work.

2. **Fill in the necessary times and dates for automatic testing.**
   Type times in the form of `hours:minutes:seconds`.

   For example: **`2:30:00`** runs SunDiag for two-and-a-half hours.

   Type dates in the form of `month/day/year`.

   For example: **`1/15/93`** represents January 15, 1993.

   Include the colons and slashes.

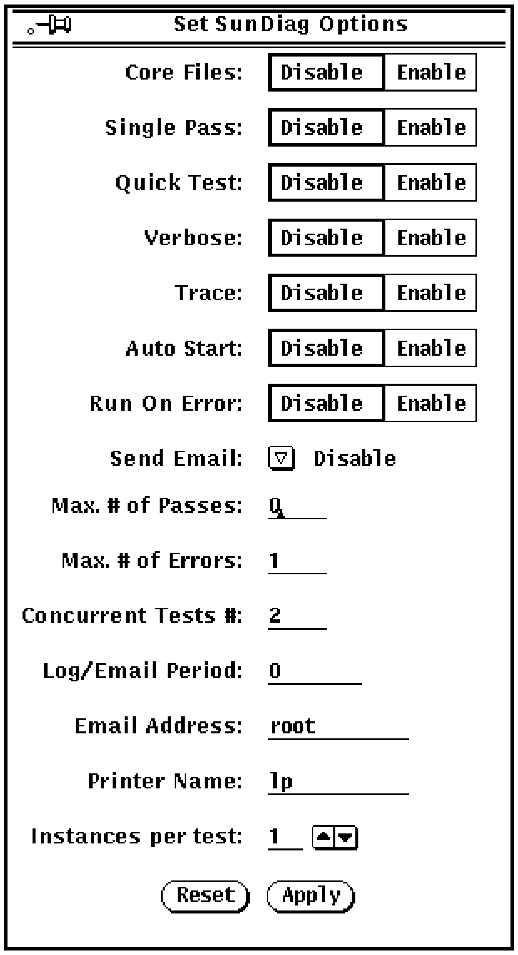
Note – You don't have to fill in all of the fields for the schedule feature to work. Stop Date, Stop Time, and Run Time override each other. If you set these fields with conflicting times, the SunDiag software will stop testing at the earliest specified time.

---

3. **Click Apply.**
   The SunDiag exerciser will begin automatic testing at the start time and continue testing until either the Run Time or Stop Time parameter is met.

2.2.4.2 Set Options Window Button

The Set Options menu sets various global test options, such as whether core files are enabled or disabled, and whether the tests should run on an error condition. Click SELECT on the Set Options window button to display the Set Options menu, like the one shown in Figure 2-12.



Set SunDiag Options

Core Files:	[✓ Disable] Enable
Single Pass:	[✓ Disable] Enable
Quick Test:	[✓ Disable] Enable
Verbose:	[✓ Disable] Enable
Trace:	[✓ Disable] Enable
Auto Start:	[✓ Disable] Enable
Run On Error:	[✓ Disable] Enable
Send Email:	Disable
Max. # of Passes:	0
Max. # of Errors:	1
Concurrent Tests #:	2
Log/Email Period:	0
Email Address:	root
Printer Name:	lp
Instances per test:	1

Figure 2-12 Set Options Menu

You can only change items on this menu when testing is stopped. The settings are enabled or disabled by clicking SELECT. Selected items appear in bold frames on monochrome monitors, and “pushed-in” on color monitors.

Core Files

When you click Disable, the tests try to capture the signals that cause core dumps. This means that you receive informative messages instead of a core file. These messages will appear on the SunDiag console window, and they will be stored in the information log file `/var/adm/sundiaglog/sundiag.info` and in the error logs file `/var/adm/sundiaglog/sundiag.err`. See “Log Files Window Button” on page 2-11 for more details.

Single Pass

When enabled, all SunDiag tests will only execute one time.

Quick Test

When enabled, abbreviated versions of the enabled tests are run. See the individual test descriptions in Part 2 of this book for details; some tests perform normally when this option is specified.

Verbose

When enabled, more diagnostic messages are printed in the console window while tests are running.

Trace

When enabled, the test programs are traced down by function names. Messages are printed in the console window while the tests are running.

Caution – Enabling trace is not recommended for the normal operation of the SunDiag system exerciser. Trace should be used for debugging purposes only, and it should be started from the command line.

Auto Start

To eliminate the need to click the Start button to start SunDiag testing:

1. **Click the Auto Start setting.**
2. **Save the global options and test options (using the Option Files menu described in the next section) to a specified option file.**
3. **Specify the saved option file (using the `-o` argument) when you type the command line to invoke the SunDiag system exerciser. The program will start testing after the window is displayed.**

Run On Error

When enabled, SunDiag runs failing tests continuously until the number specified for Max. # of Errors is reached.

Send Email

This feature is designed to keep you apprised of testing status by sending the most recent 60 lines of the SunDiag information file, `/var/adm/sundiag.info`. Email is sent to the host machine.

To select the automatic send mail feature, press MENU on the Send Email abbreviated menu button. The choices are:

• Disable	- no email is sent.
• On_Error	- email is sent when the SunDiag software finds an error.
• Periodically	- used in conjunction with the Log/Email Period option. When running, the SunDiag software sends email once every *x* minutes, where *x* is the period set in the Log/Email field.
• Both	- SunDiag sends email both on error and periodically.

Max. # of Passes

If you enable this option, the SunDiag exerciser runs passing tests continuously until the number specified for Max. # of Passes is reached. Setting this option to 0 runs the tests indefinitely until interrupted.

Max. # of Errors

See “Run On Error” above. A setting of 0 will continue testing no matter how many errors are encountered.

Concurrent Tests #

Type the number of tests you want to run concurrently.

Note – If you set this parameter to a high value, then you may need to set the maxusers parameter to accommodate the increased number of UNIX processes. See Section 1.2.6, “Setting the Maximum Number of Processes,” on page 1-8, for more information.

Log/Email Period

This option sets the time period you want the email error message to be sent (if enabled) and enables the periodic status log. To set the Log/Email Period, type the number of minutes. A zero entry disables this function.

Email Address

This option tells SunDiag where to send the email message. The default is root. To change the address, remove the root entry and type a new address.

Printer Name

Type the name of the printer you want to use. The name should match an entry in your `/etc/printcap` file, such as **`lp1`**, or the name of the host to which the printer is attached. If you do not specify a printer, SunDiag uses the printer specified with the PRINTER environment variable. If there is no PRINTER variable, SunDiag prints on the printer specified by the `lp` entry in your `/etc/printcap` file. Any printer you specify must exist in the `printcap` file.

Instances Per Test

See Section 4.1, “Scalability Options” for instructions.

PAM (Scalable)

See Section 4.1, “Scalability Options” for instructions.

PAM (Non-scalable)

See Section 4.1, “Scalability Options” for instructions.

The Reset and Apply buttons

Click the Reset button to reset the options to the default values (shown in Figure 2-12) and close the window. The Apply button applies the changed options to the SunDiag tests when they are started. You must click Apply for the changes to take effect.

Note – “Unpinning” the Set Options window before clicking Reset or Apply negates any changes you make.

2.2.4.3 Option Files Window Button

Clicking the Option Files window button displays a pop-up window menu for the option files that exist in the `/var/adm/sundiaglog/options` directory. The options stored in these files are those you have chosen from the Set Options menu and from the control panel for specific tests. The Option Files button works only when tests are stopped.

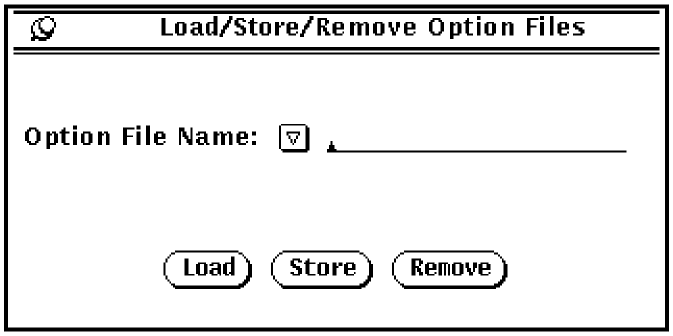


Figure 2-13 Option Files Menu

▼ How to Save Options to an Option File

1. **Type the name of your option file in the test field.**
   The SunDiag default option file is named `.sundiag`. If you start the software without the `-o` option, the `.sundiag` file is used, if it exists.

2. **Click Store.**
   The SunDiag exerciser saves *all* options — including those from the Set Options file and from individual test option menus — to the specified file name.

▼ How to Load an Option File

1. **Press MENU on the abbreviated menu button to select an option file.**
   The options saved in the available option files listed here are the ones you set in the Set Options menu, in addition to the options chosen for each individual test.

2. **Click Load.**
   This loads your current settings from the named file. To write a new file, click Store, and then Click Load. The next time you start testing, the options you saved will be used, and the control panel will reflect these choices.

   Click Remove to remove the named option file. The SunDiag program will prompt you for confirmation before removing the file.

Note – In option files, the Processor Affinity filed is a binary value (or “bit position”) that simulates check boxes in the OpenWindows software. See Section 3.7, “Setting Processor Affinity Masks in TTY Mode,” on page 3-20 for more information about setting Processor Affinity Masks in ascii format.

2.2.4.4 Status View Button

In some large systems, such as the SPARCcenter™ 2000 system, the SunDiag system exerciser may have as many as 100 disk tests running at one time. Each test is listed in the system status panel. You to quickly scroll the system status panel to a specific device using the Status View button.

The Errors Only option changes the system status panel to only list tests that have failed. Errors Only is the default option; clicking SELECT toggles between viewing Errors Only and All Tests. The first, last, previous, and next page items scroll the list to the desired page.

Note – The status view menu button is available if the system status panel is more than one page long, but certain choices (Next Page, Previous Page, First Page, and Last Page) are unavailable. Errors Only lists only tests that have failed. Go To Device has no effect.

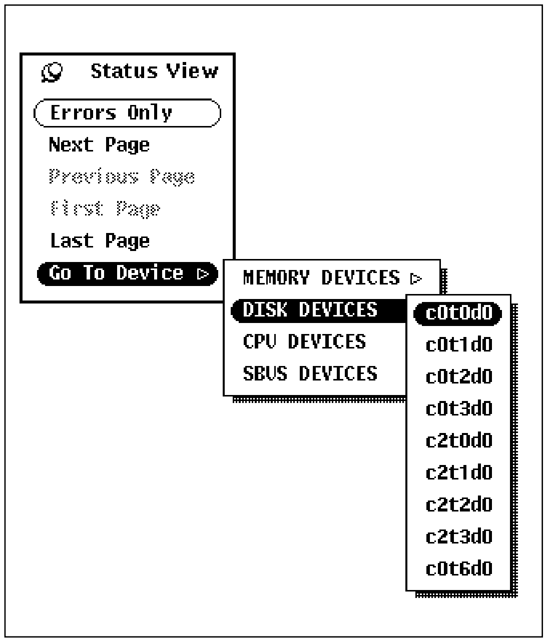


Figure 2-14 Using the Status View Button

2.2.4.5 *Intervention Mode*

This setting serves as a reminder that you must intervene before a test or tests can be run successfully. This setting affects two categories of tests:

- Tests of drives that require scratch media (tapes, discs, or diskettes)
- Tests that require loopback connectors

The SunDiag software will not run these tests until you have enabled intervention mode by clicking Enable for the “Intervention Mode” setting on the Control Panel. This setting does not change the program’s behavior; it just serves as a reminder that you must intervene.

2.2.4.6 *Test Setting*

The test setting below the intervention setting is a quick way for you to select groups of tests. Click Default to enable the default group of tests. The lower part of the control panel will enable those tests. Click None to enable none of the individual tests. Click All to select all of the tests.

Note – Clicking SELECT on an individual test name (“disabling”) while tests are running stops that particular test. Likewise, clicking None while tests are running is the same as clicking the Stop button.

2.3 *Changing Individual Test Options*

The lower part of the control panel lists the devices and corresponding tests available for testing. For example, `fstest` is a test that checks the file system structure for a disk, and so the SunDiag program provides an `fstest` for each installed disk drive. Options menu buttons for each particular test are also shown. Click the Options menu buttons to display menus to change individual test parameters. See “Options Menu Buttons” on page 2-26 for more details.

Devices are arranged into the following categories:

- MEMORY DEVICES
- DISK DEVICES
- CPU DEVICES
- SBUS DEVICES
- TAPE DEVICES

When started, the SunDiag software probes the kernel for available hardware devices and displays them in this part of the control panel.

When the SunDiag main window appears, the control panel shows all tests that have been enabled. This list includes all the devices found during the kernel probe, except those which require intervention or additional equipment. Clicking on the Start button will start all enabled tests.

Enable or disable tests by clicking the test names. Test names that are framed in bold or appear to be “pushed in” are selected. You can also enable or disable groups of tests by selecting the device category name.

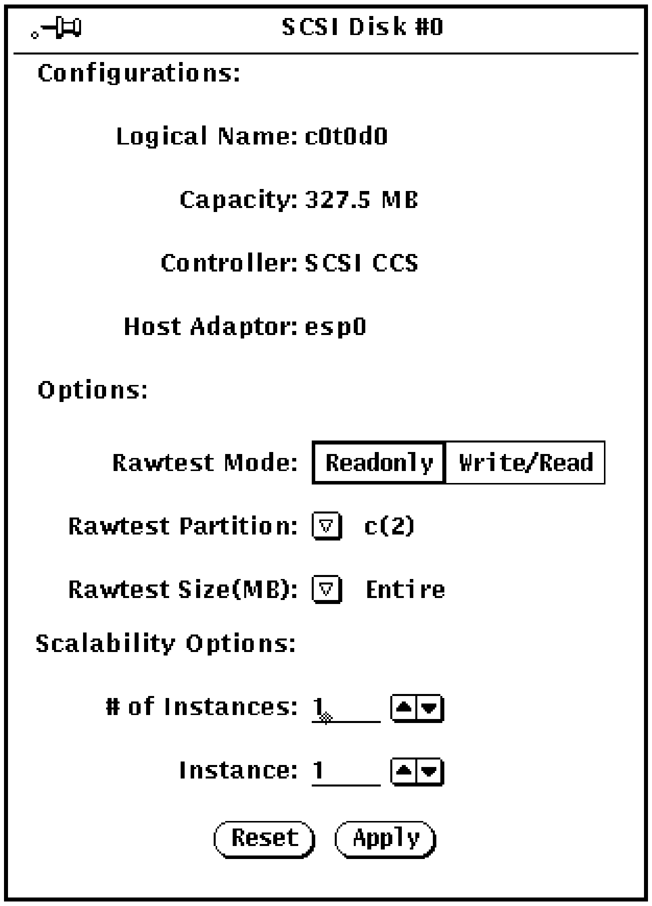
If a device is enabled during testing, the test(s) are started as soon as possible. If an enabled device is disabled during testing, the test(s) stop immediately.

Note – `tapetest` may not stop until the tape has fully rewound.

2.3.1 Options Menu Buttons

All tests have options menus that allow you to specify test parameters. Options menus are displayed by clicking SELECT on the options window buttons to the right of each test listed on the control panel. (See Figure 2-2) All test options menus have configurations and options fields.

Figure 2-15 shows the `rawtest` option menu for SCSI disk #0. It shows that the disk has a 327.5 MB capacity, and that its controller ID is an SCSI CCS. This information comes from the operating system kernel. This test offers an exclusive setting control that can be “toggled” between read-only and read / write testing.



SCSI Disk #0

Configurations:

Logical Name: c0t0d0

Capacity: 327.5 MB

Controller: SCSI CCS

Host Adaptor: esp0

Options:

Rawtest Mode: [✓ Readonly] Write/Read

Rawtest Partition: c(2)

Rawtest Size(MB): Entire

Scalability Options:

# of Instances: 1

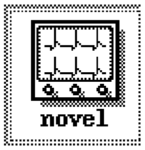
Instance: 1

Figure 2-15 Options Menu for `rawtest`

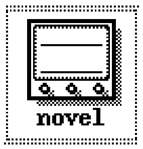
Refer to the individual test descriptions in Part 2 — “SunDiag Tests” for test-specific options. When you are ready to view another option menu, double-click on the pushpin to dismiss the option menu, or Reset or Apply any changes you have made. You can only view or make changes to one option menu at a time.

2.4 Running the SunDiag Program from an Icon

The SunDiag program continues to run if you close the SunDiag window to an icon. Close the SunDiag window as you would any other OPEN LOOK window by clicking SELECT on the window menu button on the upper left of the window frame. (You can also use the OPEN key keyboard shortcut). Even in icon mode, you can monitor the SunDiag program for test failures.



This icon shows that SunDiag is running.



This icon shows that an error has occurred, and SunDiag may have stopped running. SunDiag continues to run if the "Run On Error" option is enabled, or if the "Maximum # of Errors" has not been reached.

Figure 2-16 Icons for SunDiag

The SunDiag TTY Interface 3

This chapter describes how to set testing parameters using the SunDiag TTY interface. This interface makes it possible to run the SunDiag system exerciser from a terminal attached to a TTY (or serial) port. This interface is also used by default when you remotely log into a system.

To use this interface, a terminal can be attached to a serial port of the system under test or connected with a modem. All the options available in the OPEN LOOK version of the SunDiag interface have command equivalents in TTY mode. These commands are described in more detail in Chapter 2, “The SunDiag OPEN LOOK Interface.”

Note – If you are remotely logged into a Sun system, you may still run the SunDiag software with the OpenWindows interface. Refer to Section 1.7, “Running the SunDiag Exerciser on a Remote System,” on page 1-15 for more information.

3.1 Using the SunDiag Program from a Terminal

The workstation under test must have the SunDiag software already loaded. A terminal must be remotely logged into the system, or attached to a serial port.

Starting SunDiag from a TTY port (terminal) is almost identical to starting it in OPEN LOOK. Type in the command line described in Section 1.4, "Starting the SunDiag Exerciser." The program senses that you are logging in from a terminal interface and automatically brings up the TTY version of the control panel.

Note – You can run the SunDiag software in TTY mode from a monitor. However, some framebuffer tests will fail. See Section , "TTY Mode and Framebuffer Window Locking," on page 1-5 for more information.

3.1.1 Starting TTY Mode from a Shell Tool

To run the SunDiag software in TTY mode from an OpenWindows shell tool window, become superuser, and type:

```
# /opt/SUNWdiag/bin/sundiag -t
```

A brief initialization message is displayed, followed by the main display, as shown in Figure 3-1.

Multiprocessing systems

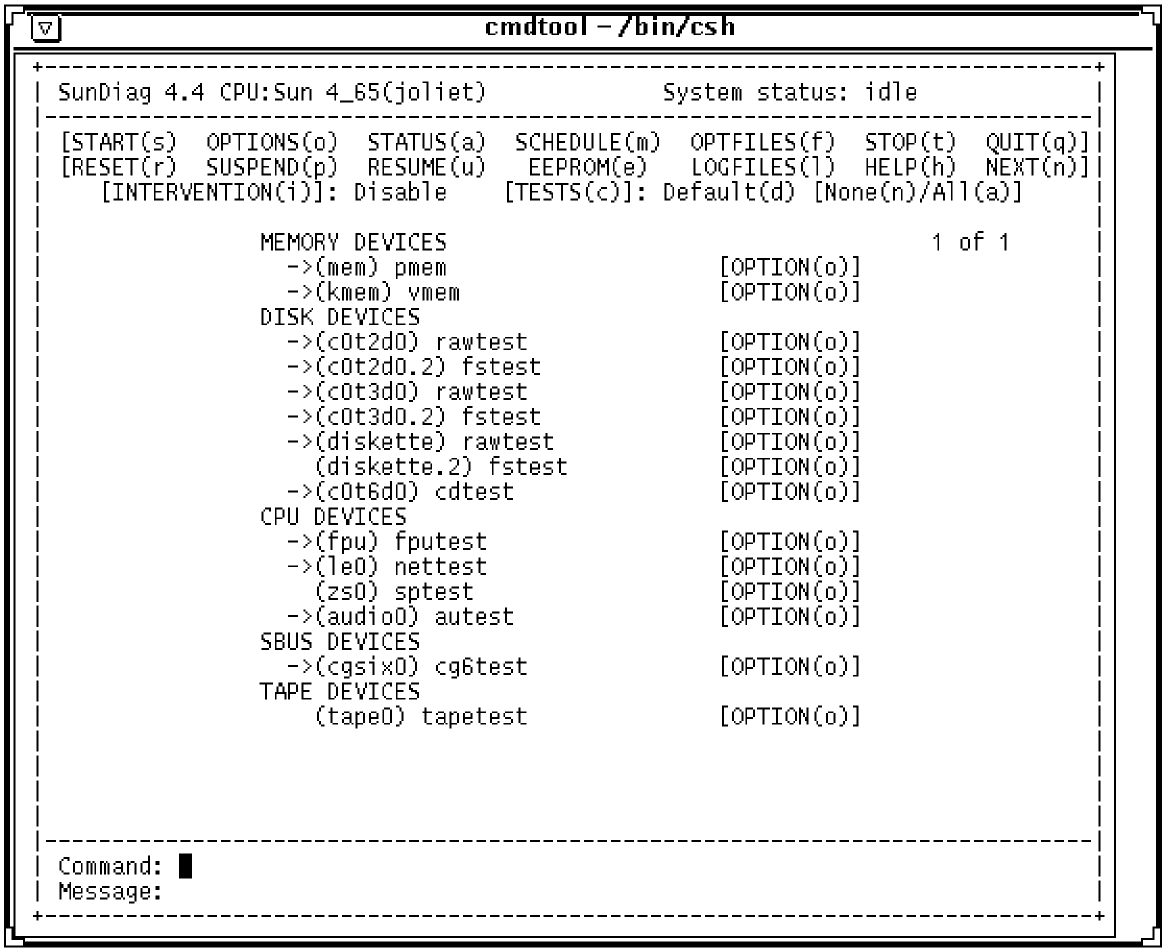
If your system under test is a multiprocessing system, the SunDiag program displays the number of processors at the top of each TTY screen after the hostname.

3.1.2 Executing Commands in TTY Mode — Use the Abbreviations

You can execute commands, set options, or change screens by typing at the `Command:` prompt at the bottom of each TTY screen. Each of the TTY commands has a one-letter abbreviation shown in parenthesis. You can type the full command name, but you may find that it's easier to type the abbreviation.

3.1.3 Navigating the Screens

You can view other screens and menus using the commands displayed at the top of each screen. To go back to a previous screen, type **d** for `done` at the command prompt.



```
+------------------------------------------------------------------------------+
| SunDiag 4.4 CPU:Sun 4_65(joliet)             System status: idle             |
|------------------------------------------------------------------------------|
| [START(s)  OPTIONS(o)  STATUS(a)  SCHEDULE(m)  OPTFILES(f)  STOP(t)  QUIT(q)]|
| [RESET(r)  SUSPEND(p)  RESUME(u)   EEPROM(e)   LOGFILES(l)  HELP(h)  NEXT(n)]|
|    [INTERVENTION(i)]: Disable    [TESTS(c)]: Default(d) [None(n)/All(a)]     |
|                                                                              |
|               MEMORY DEVICES                                  1 of 1        |
|                 ->(mem) pmem                [OPTION(o)]                     |
|                 ->(kmem) vmem               [OPTION(o)]                     |
|               DISK DEVICES                                                   |
|                 ->(c0t2d0) rawtest          [OPTION(o)]                     |
|                 ->(c0t2d0.2) fstest         [OPTION(o)]                     |
|                 ->(c0t3d0) rawtest          [OPTION(o)]                     |
|                 ->(c0t3d0.2) fstest         [OPTION(o)]                     |
|                 ->(diskette) rawtest        [OPTION(o)]                     |
|                   (diskette.2) fstest       [OPTION(o)]                     |
|                 ->(c0t6d0) cdtest           [OPTION(o)]                     |
|               CPU DEVICES                                                    |
|                 ->(fpu) fputest             [OPTION(o)]                     |
|                 ->(le0) nettest             [OPTION(o)]                     |
|                   (zs0) sptest              [OPTION(o)]                     |
|                 ->(audio0) autest           [OPTION(o)]                     |
|               SBUS DEVICES                                                   |
|                 ->(cgsix0) cg6test          [OPTION(o)]                     |
|               TAPE DEVICES                                                   |
|                   (tape0) tapetest          [OPTION(o)]                     |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
|------------------------------------------------------------------------------|
| Command: █                                                                   |
| Message:                                                                     |
+------------------------------------------------------------------------------+
```

Figure 3-1 SunDiag TTY Main Display

3.1.4 Main Display Commands

This section describes what to expect when you enter the commands shown in brackets at the top of the TTY main display screen. The commands are described in the order they appear on the screen, from left to right.

Note – Options are typed at the Command prompt at the bottom of the screen. Press Return after you type any of these commands.

[START]
Typing **`start`** or **`s`** begins testing. After you have selected the tests and entered the system level and test level options of your choice, type **`start`** to begin testing. The `Status` window at the top of the screen will display "testing."

[OPTIONS]
Type **`options`** or **`o`** to display a menu of global system level options that affect every test. See Section 3.2, "Options Menu," on page 3-8 for an example of the display and a description of each option.

[STATUS]
Type **`status`** or **`a`** to display information about the tests in progress.

[SCHEDULE]
Type **`schedule`** or **`m`** to display the scheduling options (See "How to Use SunDiag's Schedule Feature" on page 2-17.)

[OPTFILES]
Type **`optfiles`** or **`f`** to display global options you can save (see "Option Files Window Button" on page 2-22.)

[STOP]
Type **`stop`** or **`t`** to stop all testing. While the SunDiag exerciser is stopping the tests in progress, the top right of the screen displays:

```
System Status: stopping
```

When all tests are stopped, the system status is `idle`.

To stop an individual test, type the test name:

```
Command: mem
Message:
```

If a test has already stopped, type the test name to restart it.

[QUIT]
Type **`quit`** or **`q`** to exit the SunDiag TTY interface. If you have remotely logged in, change directories to your remote entry point and exit the remote log-in.

[RESET]
Type **`reset`** or **`r`** to reset the passes to zero. This also resets the error count for each test, the system passes, and the total errors. This option is valid only while tests are stopped.

[SUSPEND]
Type **`suspend`** or **`p`** to suspend all testing. When you type this command, the system status window at the top of the screen displays "suspended."

[RESUME]
Type **`resume`** or **`u`** to resume testing after you've suspended testing. When you type this command, the system status window displays "testing."

[EEPROM]
The SunDiag software and the Solaris environment only support SPARC architectures. Typing **`eeprom`** or **`e`** will report the following error:

```
The EEPROM feature is not available!
```

[LOGFILES]
Type **`logfiles`** or **`l`** to display the operations you can perform on the log files stored in the `/var/adm/sundiaglog` directory. See "The Log Files Window Button" in Chapter 2 for more details.

[HELP]
This option is not available at this time.

[NEXT]
Type **`next`** or **`n`** to display the next screen of information when a second screen is needed. When you reach the end of the display, type **`n`** again to page back to the beginning, or type a new command from those in brackets at the top of the screen.

[INTERVENTION]
When you type **`intervention`** or **`i`** in the control panel display, the `[INTERVENTION]:` text item toggles from `Disable` to `Enable` or from `Enable` to `Disable.` When you enable intervention mode, you can now enable tests that require special loopback connectors and scratch media. If you try to enable a test that requires intervention mode and it is not enabled, a message at the bottom of the screen reminds you to enable it.

[TESTS]
Type **`tests default`** or **`c d`** to run all default tests when you type start. A `->` symbol will appear before each default test. All default tests run when you type **`start`**.

If you type **`tests none`** or **`c n`**, no tests are enabled, and the `->` symbols disappear from the test names.

If you type **`tests all`** or **`c a`** and intervention is disabled, all tests that do not require user intervention are enabled, and an arrow is displayed in front of each test. If intervention mode was enabled before you entered **`tests all`**, all tests displayed are enabled.

3.1.5 Scrolling

With the `next` command you can view a display that is too long to fit on one terminal screen. When you reach the end of the display, the `next` command takes you back to the beginning.

When you type the first character of a command, the screen clears any messages at the bottom of the screen.

3.1.6 Redrawing the Screen

To redraw the screen, type Control-L.

When tests are actively running, an asterisk is displayed before the test name, as in the OPEN LOOK mode.

Some commands bring up a secondary display that replaces the existing one. You can type **`d`** for `done` from most displays and return to the control panel display.

The SunDiag TTY interface attempts to simulate all the windows you see when you use the OPEN LOOK interface. In some cases, features are combined for your convenience. For example, the status panel shows whether a test is in progress or has failed. A secondary display either shows all available control panel commands besides those available from the current level, or provides a `done` command that returns you to the original display.

Note – When you choose to `Display` a log file, the SunDiag TTY interface uses vi to display the file. You must then use the vi command **`:q`** to exit from that display.

3.1.7 Running the SunDiag Exerciser in the Background

To run the SunDiag exerciser with the TTY interface in the background, press Control-X. This frees your screen for other functions. You can also log out of the remote system after placing the program in background mode. Customer Support personnel often run the SunDiag software in the background to free up the system for other diagnostic applications.

To bring the SunDiag exerciser back into view, log back into the remote system, if necessary, and type **`/opt/SUNWdiag/bin/sundiagup`**. The SunDiag display will reappear.

3.1.8 Enabling and Disabling Individual Tests

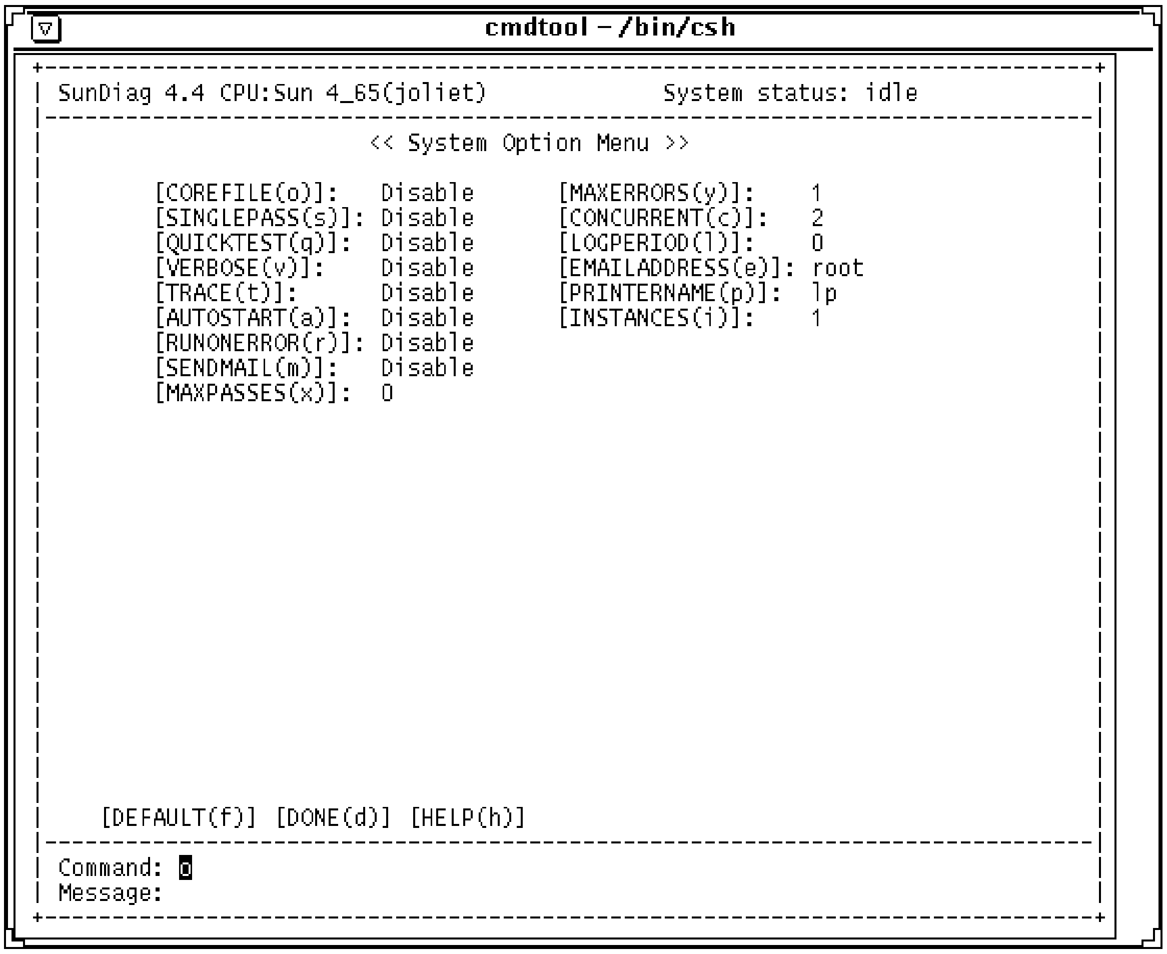
If you do not want to run the preselected default tests or all the tests, you can enable or disable any test by entering its name as shown in parentheses. For example, if there is no `->` symbol in front of the display

```
(mem) Physical                [OPTION]
```

you can type **`mem`** to enable the physical memory test. An arrow is displayed before the test name. Typing **`mem`** again disables the test and removes the arrow. Tests that do not have an arrow in front of the test name are not run when you type **`start`**. As mentioned earlier, you can also type a test name to stop an individual test in progress, or start a new test.

3.2 Options Menu

Typing **`options`** or **`o`** in the control panel displays the system option menu as shown in Figure 3-2.



```
+---------------------------------------------------------------------------+
| SunDiag 4.4 CPU:Sun 4_65(joliet)           System status: idle           |
|---------------------------------------------------------------------------|
|                          << System Option Menu >>                         |
|                                                                           |
|        [COREFILE(o)]:   Disable      [MAXERRORS(y)]:    1                 |
|        [SINGLEPASS(s)]: Disable      [CONCURRENT(c)]:   2                 |
|        [QUICKTEST(q)]:  Disable      [LOGPERIOD(l)]:    0                 |
|        [VERBOSE(v)]:    Disable      [EMAILADDRESS(e)]: root              |
|        [TRACE(t)]:      Disable      [PRINTERNAME(p)]:  lp                |
|        [AUTOSTART(a)]:  Disable      [INSTANCES(i)]:    1                 |
|        [RUNONERROR(r)]: Disable                                           |
|        [SENDMAIL(m)]:   Disable                                           |
|        [MAXPASSES(x)]:  0                                                 |
|                                                                           |
|    [DEFAULT(f)] [DONE(d)] [HELP(h)]                                       |
|---------------------------------------------------------------------------|
| Command: o                                                                |
| Message:                                                                  |
+---------------------------------------------------------------------------+
```

Figure 3-2 System Option Menu

Commands can be entered as complete words or with the abbreviations displayed in parentheses after each command word.

These commands correspond to the Set Options menu items described in detail in Chapter 2, "The SunDiag OPEN LOOK Interface."

When you type **`default`** or **`f`**, the preselected default system options are enabled. The display should display `enabled` after each default option.

To exit from this display, type **done** or **d**.

The corefile (`o`), single pass (`s`), quick test (`q`), verbose (`v`), auto start (`a`) and run on error (`r`) options toggle from `disable` to `enable` when you type their commands. For example, if the display shows:

```
  [COREFILE] disable
```

and you type **`corefile`** or **`o`**, the option of having a core dump saved in a file is enabled, and the display changes to `Enable`. When you type the command again, the display changes back to `Disable`, and the tests attempt to capture the signals that cause core dumps. Messages will be displayed instead of a core file.

To change the maximum number of errors the SunDiag exerciser allows before stopping tests, type:

```
maxerrors number
```

or:

```
y number
```

When you press Return, the number you have typed is displayed after `maxerrors`.

To specify the maximum number of tests you want to execute concurrently, type:

```
concurrent number
          or
c number
```

To specify where you want your log files to be printed, type the printer name as it appears when you type `lpstat -v`. For example:

```
printer lp1
```

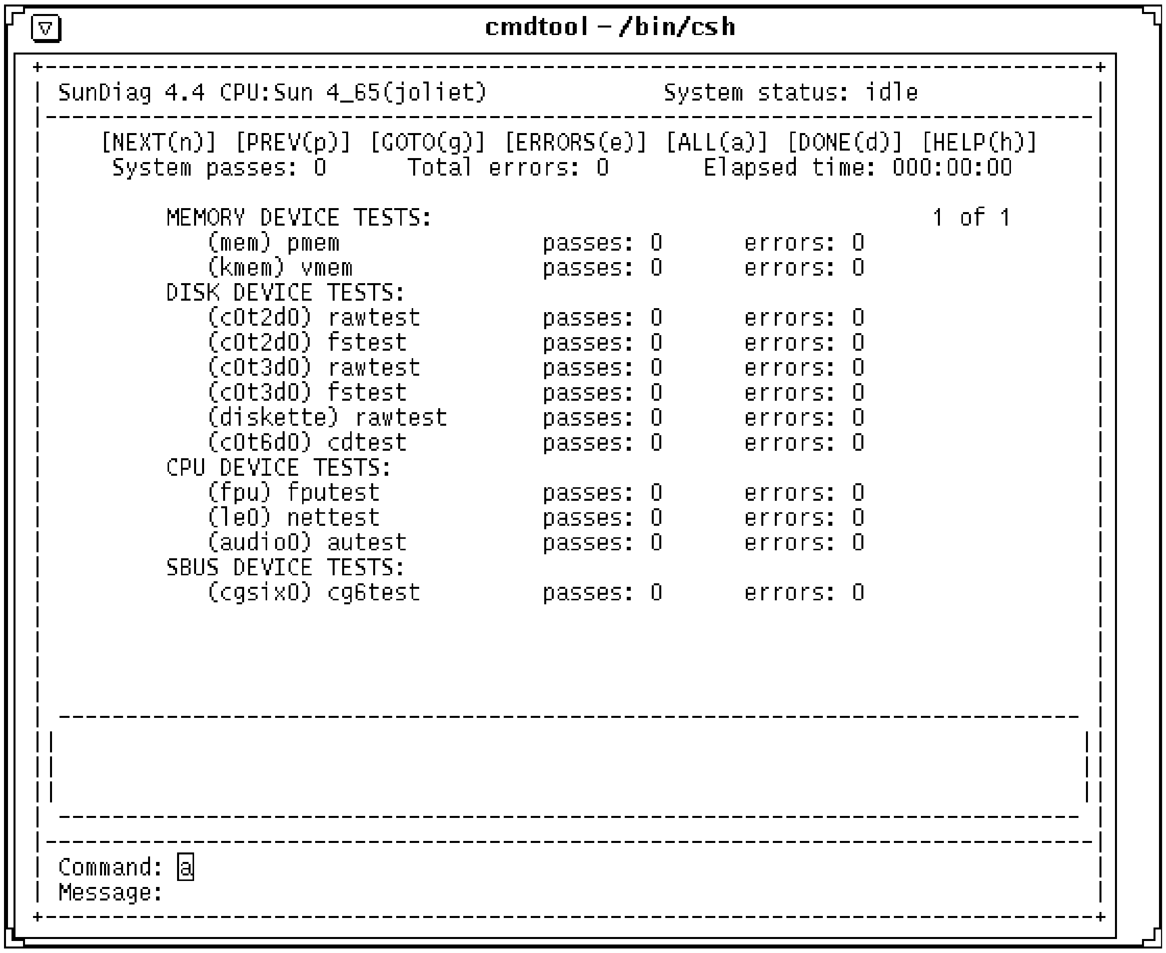
You can also type:

```
p printer_name
```

Note – You cannot print a screendump in TTY mode.

3.3 System Status Display

Type **status** or **a** in the control panel to display the screen shown in Figure 3-3.



```
+------------------------------------------------------------------------------+
| SunDiag 4.4 CPU:Sun 4_65(joliet)            System status: idle              |
|------------------------------------------------------------------------------|
|    [NEXT(n)] [PREV(p)] [GOTO(g)] [ERRORS(e)] [ALL(a)] [DONE(d)] [HELP(h)]    |
|     System passes: 0      Total errors: 0       Elapsed time: 000:00:00      |
|                                                                              |
|        MEMORY DEVICE TESTS:                                    1 of 1        |
|           (mem) pmem               passes: 0      errors: 0                  |
|           (kmem) vmem              passes: 0      errors: 0                  |
|        DISK DEVICE TESTS:                                                    |
|           (c0t2d0) rawtest         passes: 0      errors: 0                  |
|           (c0t2d0) fstest          passes: 0      errors: 0                  |
|           (c0t3d0) rawtest         passes: 0      errors: 0                  |
|           (c0t3d0) fstest          passes: 0      errors: 0                  |
|           (diskette) rawtest       passes: 0      errors: 0                  |
|           (c0t6d0) cdtest          passes: 0      errors: 0                  |
|        CPU DEVICE TESTS:                                                     |
|           (fpu) fputest            passes: 0      errors: 0                  |
|           (le0) nettest            passes: 0      errors: 0                  |
|           (audio0) autest          passes: 0      errors: 0                  |
|        SBUS DEVICE TESTS:                                                    |
|           (cgsix0) cg6test         passes: 0      errors: 0                  |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
| ---------------------------------------------------------------------------- |
||                                                                            ||
||                                                                            ||
||                                                                            ||
| ---------------------------------------------------------------------------- |
|------------------------------------------------------------------------------|
| Command: a                                                                   |
| Message:                                                                     |
+------------------------------------------------------------------------------+
```

Figure 3-3 System Status Display Screen

Test messages display in the message line at the bottom of the screen.

3.3.1 Status View Equivalents

TTY mode includes complementary commands for the Status View button. See Section 2.1.1, “System Status Panel,” on page 2-3 and Section 2.2.4.4, “Status View Button,” on page 2-24 for more information on the system status display. The TTY commands change the behavior of the System Status Display in the following ways:

[NEXT]
Typing **`next`** or **`n`** scrolls the system status display to the next page, if applicable.

[PREV]
Typing **`prev`** or **`p`** scrolls the system status display to the next page, if applicable.

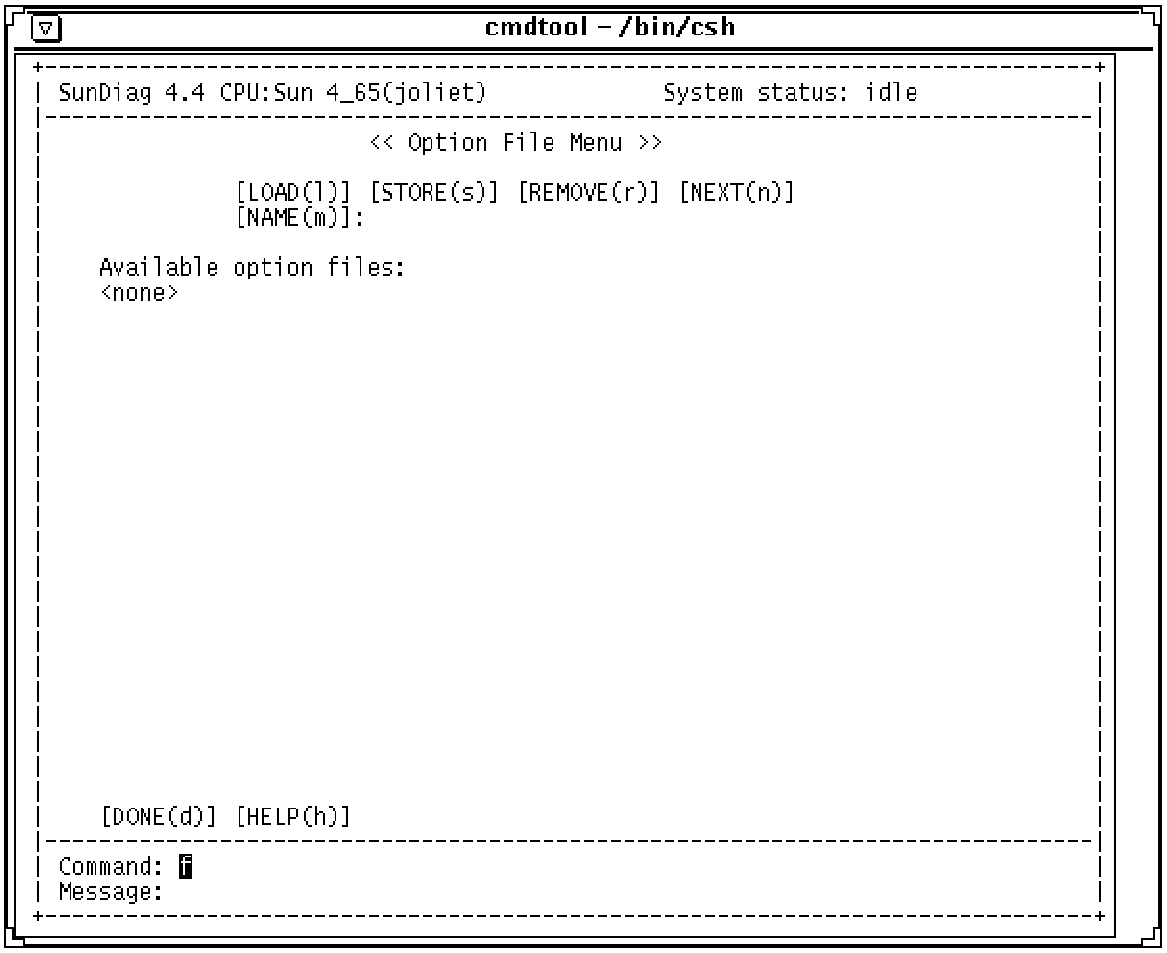
[GOTO]
Typing **`goto`** or **`g`** and then typing a page number scrolls the display to the specified page. For example, typing **`g 3`** would scroll the display to page 3.

[ERRORS]
Typing **`errors`** or **`e`** restricts the system status display to show only those tests which have failed.

[ALL]
Typing **`all`** or **`a`** returns the display to the default: all tests.

3.4 Option Files Menu

Type **`optfiles`** or **`f`** from the control panel to display the Option File menu.



```
cmdtool - /bin/csh
+------------------------------------------------------------------------------+
| SunDiag 4.4 CPU:Sun 4_65(joliet)            System status: idle              |
|------------------------------------------------------------------------------|
|                         << Option File Menu >>                                |
|                                                                              |
|             [LOAD(l)] [STORE(s)] [REMOVE(r)] [NEXT(n)]                       |
|             [NAME(m)]:                                                        |
|                                                                              |
|    Available option files:                                                   |
|    <none>                                                                    |
|                                                                              |
|    [DONE(d)] [HELP(h)]                                                        |
|------------------------------------------------------------------------------|
| Command: f                                                                   |
| Message:                                                                     |
+------------------------------------------------------------------------------+
```

Figure 3-4 TTY Option Files Menu

The Option Files menu provides the opportunity to load and store the global system options and the individual test options you have selected. The default option file is `.sundiag`. You can create option files with different names, and use the **`-o`** option when you invoke the SunDiag exerciser to use the options specified in the named file. Any command you enter applies to the latest file that was named.

```
[NAME]
```

For example, if you want the selected options to be stored in the `.sundiag` file, listed after `[NAME]`, type **`store`** or **`s`**. If you want to remove the `.sundiag` file, type **`remove`** or **`r`**. If you want to use the options that are stored in the named file the next time you run SunDiag tests, type **`load`** or **`l`**.

If you want to create a new option file, type:

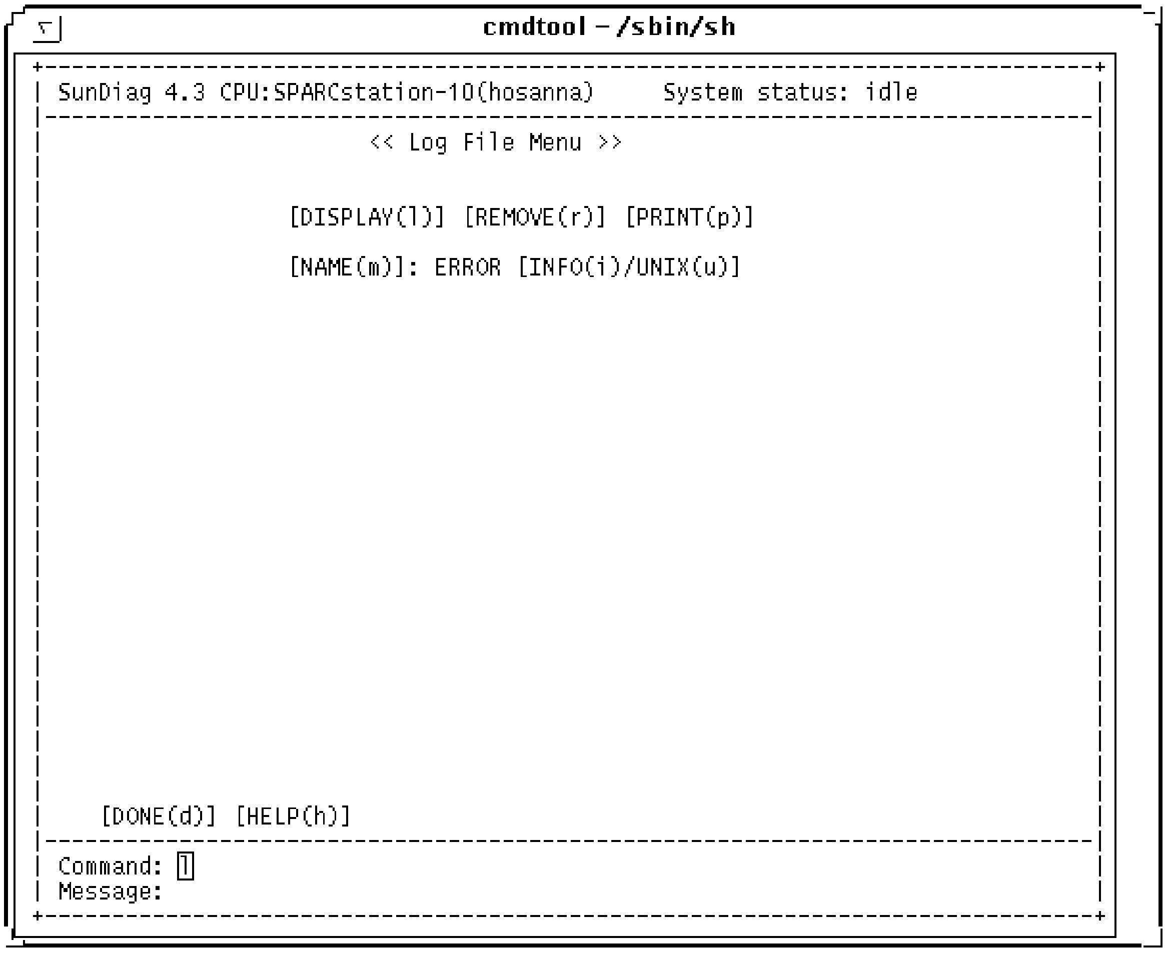
```
name newfilename
```

The name you enter is displayed after `[NAME]`. To save the file, type **`store`**.

The option files named in this display are stored in the `/var/adm/sundiaglog/options` directory. If there are more files than the screen is able to display at one time, type **`next`** or **`n`** to page forward. When you want to return to the previous display, type **`done`** or **`d`**.

3.5 Log Files Menu

Type **logfiles** or **l** in the control panel to display a menu similar to the one shown in Figure 3-5.



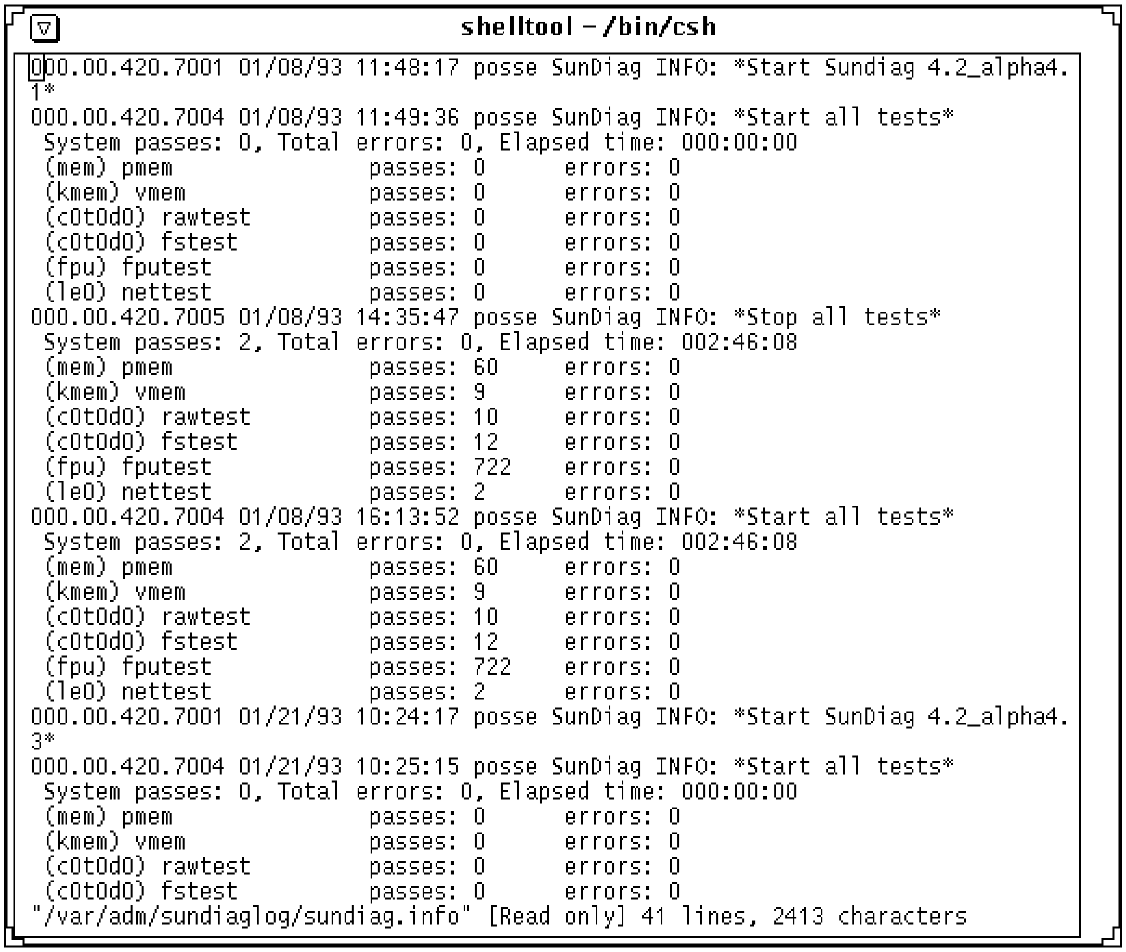
```
+------------------------------------------------------------------------------+
| SunDiag 4.3 CPU:SPARCstation-10(hosanna)     System status: idle             |
|------------------------------------------------------------------------------|
|                         << Log File Menu >>                                  |
|                                                                              |
|                                                                              |
|                   [DISPLAY(l)] [REMOVE(r)] [PRINT(p)]                        |
|                                                                              |
|                   [NAME(m)]: ERROR [INFO(i)/UNIX(u)]                         |
|                                                                              |
|                                                                              |
|    [DONE(d)] [HELP(h)]                                                       |
|------------------------------------------------------------------------------|
| Command: l                                                                   |
| Message:                                                                     |
+------------------------------------------------------------------------------+
```

Figure 3-5 TTY Log File Menu

Type **error**, **info**, or **unix** after typing **name** to toggle between the three available files you want to view or edit. When you type **unix** and **display**, for example, the `/var/adm/messages` file is displayed in the vi editor format (see

Figure 3-6). You can use `vi` commands such as Control-F or Control-B to page forward and backward through the file. To return to the Log File menu, use the **`:q`** command.

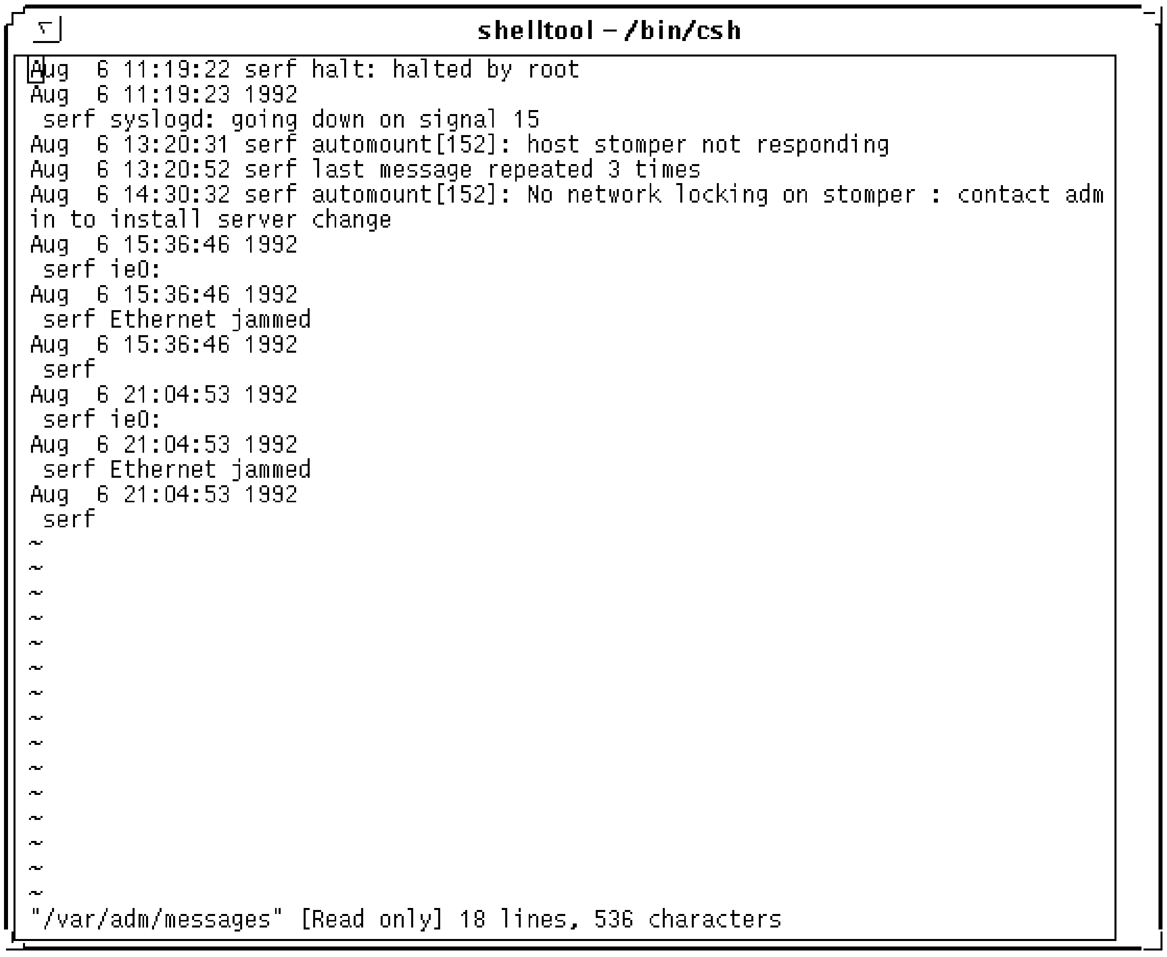
Warning – Do not attempt to edit message files while the SunDiag exerciser is running. You may only display, print, or remove these files. Editing these files while the program is running will cause the program to crash.



```
000.00.420.7001 01/08/93 11:48:17 posse SunDiag INFO: *Start Sundiag 4.2_alpha4.
1*
000.00.420.7004 01/08/93 11:49:36 posse SunDiag INFO: *Start all tests*
 System passes: 0, Total errors: 0, Elapsed time: 000:00:00
 (mem) pmem                passes: 0      errors: 0
 (kmem) vmem               passes: 0      errors: 0
 (c0t0d0) rawtest          passes: 0      errors: 0
 (c0t0d0) fstest           passes: 0      errors: 0
 (fpu) fputest             passes: 0      errors: 0
 (le0) nettest             passes: 0      errors: 0
000.00.420.7005 01/08/93 14:35:47 posse SunDiag INFO: *Stop all tests*
 System passes: 2, Total errors: 0, Elapsed time: 002:46:08
 (mem) pmem                passes: 60     errors: 0
 (kmem) vmem               passes: 9      errors: 0
 (c0t0d0) rawtest          passes: 10     errors: 0
 (c0t0d0) fstest           passes: 12     errors: 0
 (fpu) fputest             passes: 722    errors: 0
 (le0) nettest             passes: 2      errors: 0
000.00.420.7004 01/08/93 16:13:52 posse SunDiag INFO: *Start all tests*
 System passes: 2, Total errors: 0, Elapsed time: 002:46:08
 (mem) pmem                passes: 60     errors: 0
 (kmem) vmem               passes: 9      errors: 0
 (c0t0d0) rawtest          passes: 10     errors: 0
 (c0t0d0) fstest           passes: 12     errors: 0
 (fpu) fputest             passes: 722    errors: 0
 (le0) nettest             passes: 2      errors: 0
000.00.420.7001 01/21/93 10:24:17 posse SunDiag INFO: *Start SunDiag 4.2_alpha4.
3*
000.00.420.7004 01/21/93 10:25:15 posse SunDiag INFO: *Start all tests*
 System passes: 0, Total errors: 0, Elapsed time: 000:00:00
 (mem) pmem                passes: 0      errors: 0
 (kmem) vmem               passes: 0      errors: 0
 (c0t0d0) rawtest          passes: 0      errors: 0
 (c0t0d0) fstest           passes: 0      errors: 0
"/var/adm/sundiaglog/sundiag.info" [Read only] 41 lines, 2413 characters
```

Figure 3-6 TTY UNIX (System Messages) Log

Type **name info** or **i**, and then **display** or **l**, to display the TTY info log.



shelltool – /bin/csh

```
Aug  6 11:19:22 serf halt: halted by root
Aug  6 11:19:23 1992
 serf syslogd: going down on signal 15
Aug  6 13:20:31 serf automount[152]: host stomper not responding
Aug  6 13:20:52 serf last message repeated 3 times
Aug  6 14:30:32 serf automount[152]: No network locking on stomper : contact adm
in to install server change
Aug  6 15:36:46 1992
 serf ie0:
Aug  6 15:36:46 1992
 serf Ethernet jammed
Aug  6 15:36:46 1992
 serf
Aug  6 21:04:53 1992
 serf ie0:
Aug  6 21:04:53 1992
 serf Ethernet jammed
Aug  6 21:04:53 1992
 serf
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"/var/adm/messages" [Read only] 18 lines, 536 characters
```

Figure 3-7 TTY Info Log

Enter **remove** or **r** to remove the log file listed after [NAME].

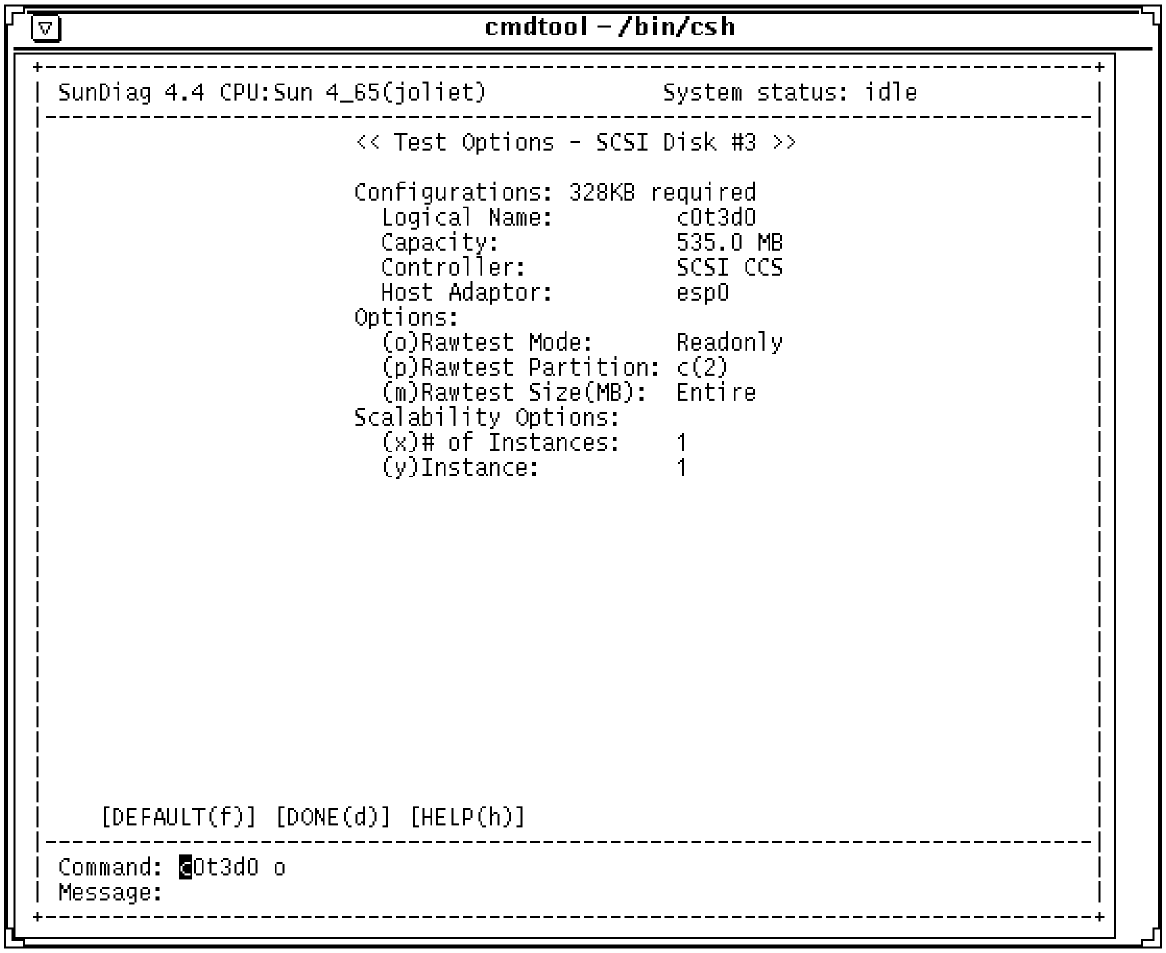
When you are ready to return to the previous menu, type **done** or **d**.

3.6 Setting Individual Test Options in TTY Mode

When you enter a test name from the control panel window, such as:

```
Command: c0t3d0 option
```

or just **c0t3d0 o**, a menu of individual test options is displayed.



```
cmdtool - /bin/csh
+-----------------------------------------------------------------------------+
| SunDiag 4.4 CPU:Sun 4_65(joliet)              System status: idle           |
|-----------------------------------------------------------------------------|
|                      << Test Options - SCSI Disk #3 >>                      |
|                                                                             |
|                      Configurations: 328KB required                         |
|                        Logical Name:          c0t3d0                        |
|                        Capacity:              535.0 MB                      |
|                        Controller:            SCSI CCS                      |
|                        Host Adaptor:          esp0                          |
|                      Options:                                               |
|                        (o)Rawtest Mode:       Readonly                      |
|                        (p)Rawtest Partition:  c(2)                          |
|                        (m)Rawtest Size(MB):   Entire                        |
|                      Scalability Options:                                   |
|                        (x)# of Instances:     1                             |
|                        (y)Instance:           1                             |
|                                                                             |
|    [DEFAULT(f)] [DONE(d)] [HELP(h)]                                         |
|-----------------------------------------------------------------------------|
| Command: c0t3d0 o                                                           |
| Message:                                                                    |
+-----------------------------------------------------------------------------+
```

Figure 3-8 TTY Test Options for `c0t0d0` (`rawtest`)

These options vary according to the type of test. Each menu displays `[DEFAULT] [DONE] [HELP]` at the bottom, and the devices found during the kernel probe are listed. Type **`f`** for `default` and the default options are enabled. Type **`d`** for `done` to go back to the previous menu. Help is not implemented at this time.

Type the character in parentheses next to the sub-test or option to toggle the option from `Enable` to `Disable`. Type the character again to toggle the option or sub-test back from `Disable` to `Enable`.

Some options, like the Wait Time option to `vmem`, have a range of values to choose from. Typing the option letter (**`w`**, in the case of the `vmem` Wait Time option) and then pressing Return will cycle the option through the range of available values.

Other options, like the Reserve option to `vmem`, can be set to arbitrary values. Type the option letter (**`r`**, in the case of the `vmem` Reserve option) followed by the value to be set. For example, typing **`r 10`** reserves 10 megabytes in the vmem test.

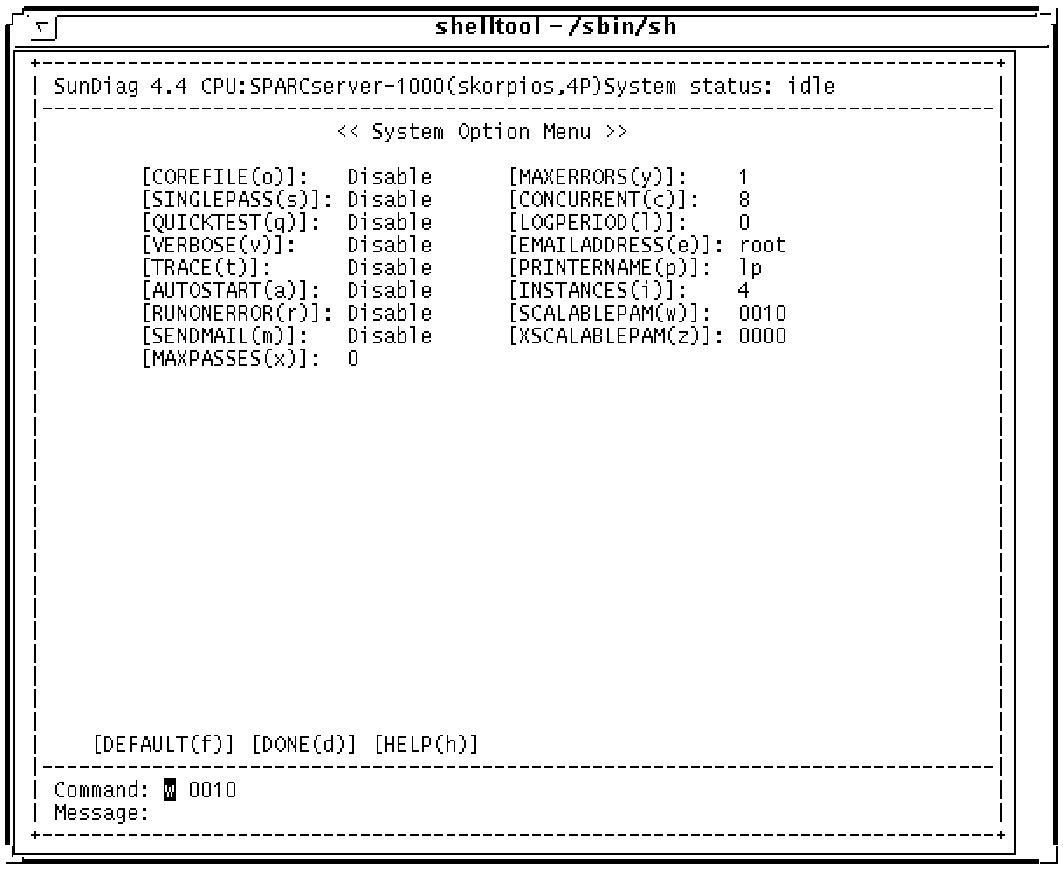
In all cases, though, the TTY test option window emulates the OpenWindows option menu for each individual test where possible. For toggled items, typing the command is sufficient. For text items, a value needs to follow the command. If you have trouble changing individual test options, refer to the option menu printed in Chapter 6, “SunDiag Test Descriptions” for more detail on how the option works.

3.7 Setting Processor Affinity Masks in TTY Mode

The processor affinity mask is a tool that enables you to specify which processor you want a test to run. By default the UNIX kernel will randomly distribute testing among all the processors it controls.

All processor affinity masks (PAMs) in TTY mode are set by turning on a "bit position." In other words, you need to enter a series of `1`'s and `0`'s for each affinity mask, depending on the configuration of your multiprocessor system.

Note – For more information on the scalability features in the SunDiag environment, see Chapter 4, "Scaling SunDiag Hardware Tests."



```
shelltool – /sbin/sh
+-----------------------------------------------------------------------------+
| SunDiag 4.4 CPU:SPARCserver-1000(skorpios,4P)System status: idle            |
|-----------------------------------------------------------------------------|
|                          << System Option Menu >>                           |
|                                                                             |
|        [COREFILE(o)]:    Disable       [MAXERRORS(y)]:     1                |
|        [SINGLEPASS(s)]:  Disable       [CONCURRENT(c)]:    8                |
|        [QUICKTEST(q)]:   Disable       [LOGPERIOD(l)]:     0                |
|        [VERBOSE(v)]:     Disable       [EMAILADDRESS(e)]:  root             |
|        [TRACE(t)]:       Disable       [PRINTERNAME(p)]:   lp               |
|        [AUTOSTART(a)]:   Disable       [INSTANCES(i)]:     4                |
|        [RUNONERROR(r)]:  Disable       [SCALABLEPAM(w)]:   0010             |
|        [SENDMAIL(m)]:    Disable       [XSCALABLEPAM(z)]:  0000             |
|        [MAXPASSES(x)]:   0                                                  |
|                                                                             |
|    [DEFAULT(f)] [DONE(d)] [HELP(h)]                                         |
|-----------------------------------------------------------------------------|
| Command: w 0010                                                             |
| Message:                                                                    |
+-----------------------------------------------------------------------------+
```

Figure 3-9 Setting the Processor Affinity Masks on Multiprocessor Systems

In the Figure 3-9, there are four binary digits beside each PAM option for the four processors in the system. There are four digits because there are four processors in this system.

```
[SCALABLEPAM(w)]:  0010
[XSCALABLEPAM(z)]: 0000
```

Note – On the OPEN LOOK interface, these four processors would be numbered: 0, 1, 2, and 3.

By default, no processor will be selected, so all the digits will be 0's. To select only one of the processors for testing, you must set the processor digit to 1.

The `[SCALABLEPAM(w)]` option is used to reserve processors for scalable tests. In Figure 3-9, the third processor was chosen for testing by typing **`w 0010`**. In this example, all the scalable tests will be run only on the third processor. To run the scalable tests on only the second processor, you would type: **`w 0100`**.

The `[XSCALABLEPAM(z)]` option is used to reserve processors for non-scalable tests. For example, to run non-scalable tests on only the fourth processor, you would type: **`z 0001`**.

Note – For a complete discussion on the PAM tool, see Section 4.1.6, "Processor Affinity Mask."

Scaling SunDiag Hardware Tests 4

4.1 Scalability Options

To thoroughly test single processor and multiprocessor systems, the SunDiag software has been expanded to allow a much greater breadth and depth of testing. Testing can now be scaled up as far as you need, to fully “stress” a system. The SunDiag program allows you to run multiple copies of some specific tests. This section describes the these scalability options and how to use them.

4.1.1 Scalable Tests

To stress processors adequately, some SunDiag tests have been modified so that multiple “Instances” (copies) of them can be run on a device simultaneously. These tests are known as Scalable Tests.

Scalable Tests

- `pmem` (Read-only Physical Memory Test)
- `vmem` (Write / Read / Compare Virtual Memory Test)
- `fputest` (FPU Test)
- `fstest` (Disk Filesystem Test)
- `rawtest` (Disk Write / Read / Compare Test)
- `nettest` (Network Interface Test, e.g. Ethernet)

4.1.2 Non-scalable Tests

All SunDiag tests that are described in Chapter 6, "SunDiag Test Descriptions" that are not listed in the "Scalable Tests" section are non-scalable. Only one copy of a non-scalable test can be run on a device at any one time.

4.1.3 Setting Scalable Test Options

Scalable test options can be set from the Set SunDiag Options menu (see Figure 4-3) or from the individual test option menus. The processors available have boxes next to their names. If a check mark appears in the box, then tests will run on that processor. Click SELECT on these check boxes to toggle the setting from enable to disable and back.

Figure 4-3 shows a Set SunDiag Options menu illustrating the scalability options near the bottom of the menu. Since no boxes are checked in this figure, the operating system will run scalable tests on any available processor. Figure 4-1 shows an `fputest` option menu with scalability options. In this figure, processor1 is enabled for testing.

Note – Saving scalable options to an options file is a somewhat complicated. If you set scalable options globally using the Set SunDiag Options menu and then save those options to an options file, the scalability options will appear to be lost when you then load that options file. The options *are* reflected on the individual test option menus, but not globally in the Set SunDiag Options menu.

4.1.4 Setting Your Own Scalable Tests

If you have added your own scalable SunDiag test, you will need to add the `SCA` option flag to the `/opt/SUNWdiag/bin/.usertest` file. The `SCA` flag specifies that the test will be scalable.

See Section 1.9.1, "Setting up a .usertest file," on page 1-17 for more information about the `.usertest` file.

4.1.5 Test Instances

"Instances" are copies of a test running simultaneously on the same device. For example, if a test has eight instances, then eight copies of that test can be run, and each copy can be run on either one or all the processors in the system being tested.

The number of instances can be set from the Set SunDiag Options menu or from the individual test option menus. The maximum number of instances for an individual test is defined, by default, as twice the number of processors in the system. You can run any number of instances of a test up to the default maximum.

The maximum number of instances can be overridden by specifying the "**`-i`** *number*" option when you start the SunDiag exerciser. See Section 1.4, "Starting the SunDiag Exerciser," on page 1-12 for more details.

You enable or disable testing on a processor by clicking SELECT on the check box next to each processor number.

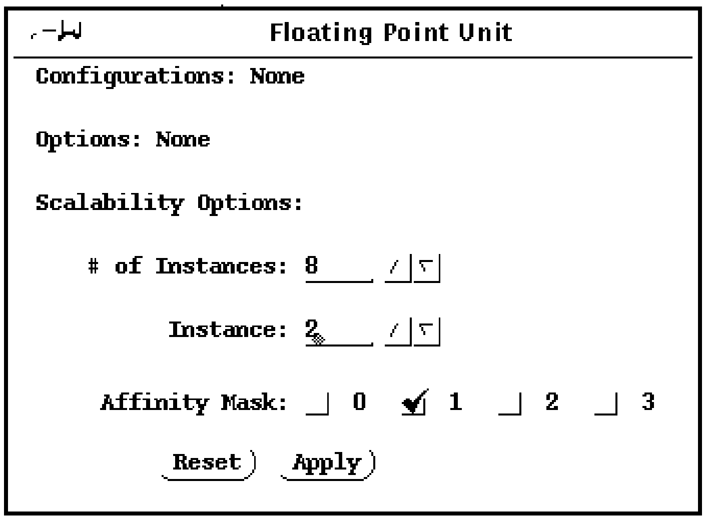
4.1.6 Processor Affinity Mask

The Processor Affinity Mask (PAM) is a tool that enables you to specify on which processors a test will run. Masks are only used on multiprocessor systems, and each processor has one corresponding mask. When a processor's mask is checked, that processor is enabled to run the test. You can set global masks from the Set SunDiag Options menu, or you can set a mask for each test from the individual test option menus.

If no processors are specified, then the SunDiag program will let the operating system randomly distribute testing among the processors it controls. It is not recommended you use random distribution, because there is no way of verifying that all processors have been adequately tested.

4.2 A Scalability Example

Figure 4-1 shows the `fputest` option menu on a multiprocessing system, illustrating the scalability features. The "Scalability Options" section allows you to set the number of Instances (copies) of a test that will run on the processor you specify. Figure 4-2 shows the same option menu on a uniprocessing system.


Floating Point Unit

Configurations: None

Options: None

Scalability Options:

# of Instances: 8

Instance: 2

Affinity Mask: ☐ 0 ✓ 1 ☐ 2 ☐ 3

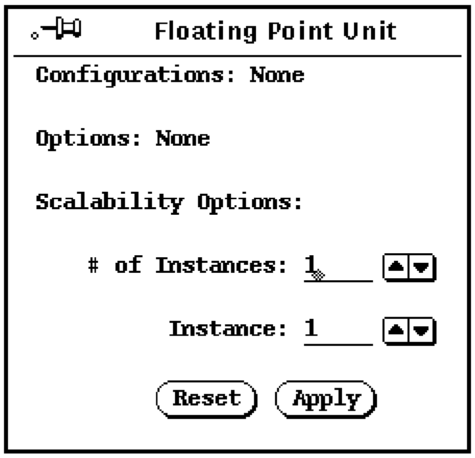
Figure 4-1 `fputest` Option Menu Example (Multiprocessor)

The "# of Instances" field displays how many copies of a scalable test are available to run. The default maximum number of instances is twice the number of processors in the system under test. The system in Figure 4-1 has four processors, so there are eight available test instances.

The Instance field in Figure 4-1 shows that the second of the eight instances of this test will be run on processor1.

Any of the instances of a scalable test can be set to any one of the processors available. That processor selection is set with the "Affinity Mask" field. That field contains a check box for each of the processors available.

Click SELECT on the check box next to the processor you wish to enable for testing. Click SELECT again to disable testing on that processor. A checked box shows which processor the test will be run on. You can only check one processor to run the test on; if no processors are selected, the test is run on all of the processors.



Configurations: None

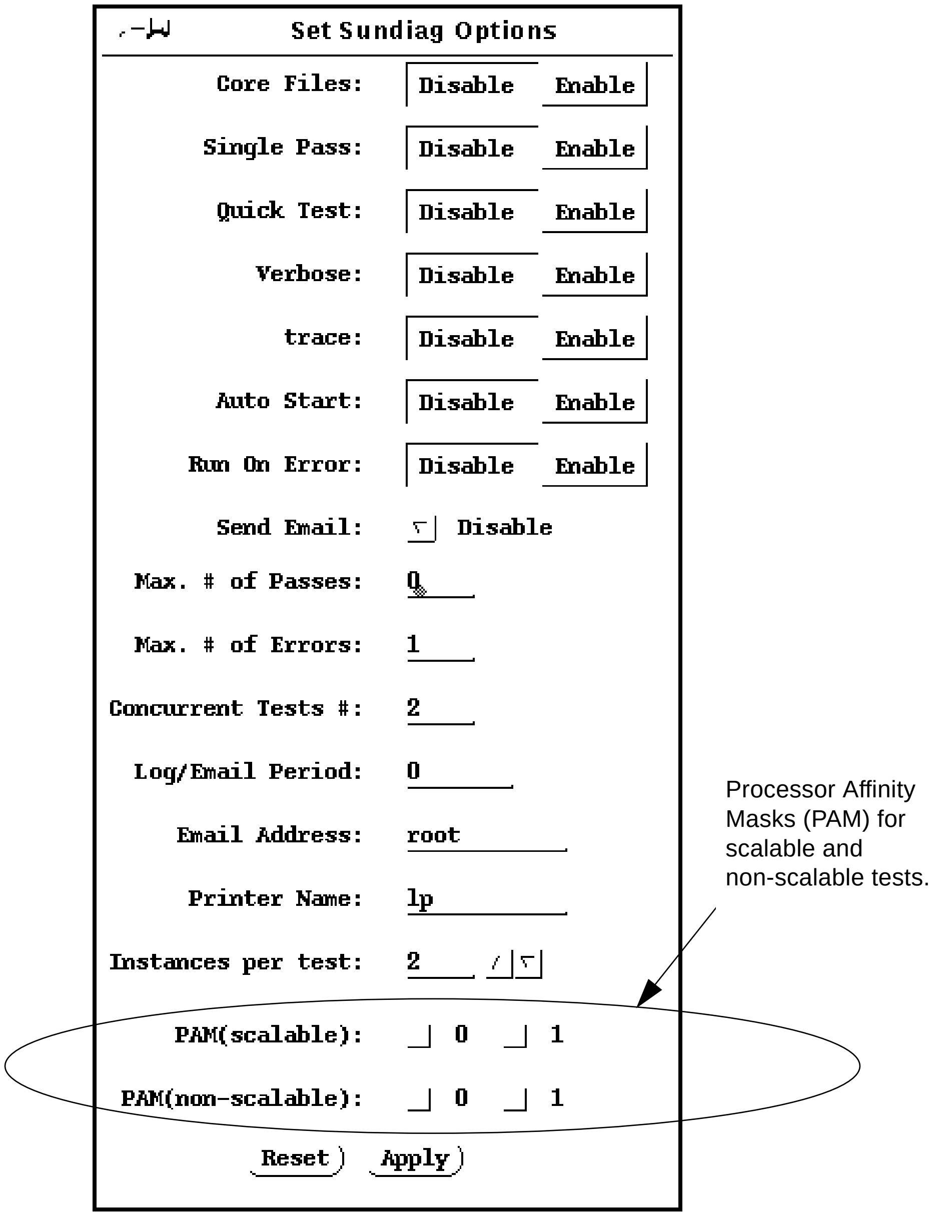
Options: None

Scalability Options:

# of Instances: 1

Instance: 1

Figure 4-2 `fputest` Option Menu Example (Single Processor)



Set Sundiag Options

Core Files: Disable Enable

Single Pass: Disable Enable

Quick Test: Disable Enable

Verbose: Disable Enable

trace: Disable Enable

Auto Start: Disable Enable

Run On Error: Disable Enable

Send Email: Disable

Max. # of Passes: 0

Max. # of Errors: 1

Concurrent Tests #: 2

Log/Email Period: 0

Email Address: root

Printer Name: lp

Instances per test: 2

PAM(scalable): 0 1

PAM(non-scalable): 0 1

Reset
Apply
Processor Affinity Masks (PAM) for scalable and non-scalable tests.

Figure 4-3 Processor Affinity Masks

▼ How to Set up Scalable Testing

1. **Decide which SunDiag tests you want to run and enable them on the control panel.**
2. **Set the testing parameters using the SunDiag control panel buttons. Open the Set SunDiag Options menu and assign global Scalable and Non-scalable tests to specific processors.**
   The choices you make here will be constant across all tests and processors. Remember, if you save these options to an options file and then load that options file again, the tests won't appear on the Set SunDiag Options menu. They will, however, be reflected in the individual option menus.
3. **Adjust the scalability options in the individual test option menus, if desired.**
   The Set SunDiag Options menu and individual test option menus can override each other. The SunDiag exerciser uses the settings applied last.

Running Individual SunDiag Tests from the Command Line 5

This chapter describes how to execute individual SunDiag tests from a command line. When running SunDiag tests individually from a command line, change directories to the directory where SunDiag is located.

5.1 Standard Arguments

Every SunDiag test will accept ten standard command arguments. The common line syntax is explained in Table 5-1.

Table 5-1 Standard Argument Syntax for Individual SunDiag Tests

`/opt/SUNWdiag/bin/`*testname test_specific_arguments* **`[cprquvdt] [h`** *hostname***`]`**

Arguments:

testname	The name of the SunDiag test as it appears in the `/opt/SUNWdiag/bin` directory.
test_specific_arguments	Refer to the individual test descriptions in “SunDiag Tests” for test-specific command arguments.
`c`	Enable Core Dump. Creates a core dump file if a system crash happens.
`p`	Single Pass Only.
`r`	Run On Error. If an error occurs, the test continues with the next test sequence instead of exiting.

Table 5-1 Standard Argument Syntax for Individual SunDiag Tests (Continued)

/opt/SUNWdiag/bin/*testname test_specific_arguments* **[cprquvdt] [h** *hostname***]**	
Arguments:	
q	Quick Test. Runs a faster, abbreviated version of the test if it exists. Not all tests have a quick option. See the individual test descriptions in the chapters that follow for more details.
u	List Usage**.** Shows how to run the test. It displays three parts: command line usage, standard arguments, and routine specific arguments.
v	Verbose Mode. Displays verbose messages regarding the test. These messages tell you more about the testing process. This mode is more valuable for some tests than others; for example, graphics tests only return start and stop messages/failures.
d	Debug Mode. Displays debug messages from the test. These messages provide more sophisticated information (mainly for test programmers).
t	Trace Mode. Displays messages that enable you to trace down function calls and the sequences being used by the test code.
h *hostname*	RPC host name. Allows you to specify a separate host name to receive messages regarding this test.

5.1.1 Tests Supported by the Command Line Interface

The following table shows each individual test in the SunDiag exerciser supported by the command line interface and its identification number defined within the SunDiag kernel. For further information about these tests, refer to the individual test descriptions in “SunDiag Tests” or the *“SunDiag User’s Guide — Addendum for SMCC Hardware.”*

Test Names and ID Numbers			
`sundiag`	0	`probe`	1
`autest`	2		
`cdtest`	4		
`cg6test`	6		
		`rawtest`	8
`fstest`	9	`mono`	17
`fputest`	12	`nettest`	18
`newtest`	19		
`pmem`	21	`sptest`	23
`sunbuttons`	24	`sundials`	25
`sunlink`	26		
`tapetest`	28		
`vmem`	30	`pstest`	31
`cg12`	32	`bpptest`	33
`lpvitest`	34		
`spiftest`	36	`mptest`	37
`gttest`	38	`fbtest`	39
		`audbri`	41
		`xbtest`	43
		`isdntest`	45

Table 5-2 Tests Supported by the Command Line Interface

5.2 Running SunDiag Tests - Overview

The following is an overview of the SunDiag testing procedure.

1. **Log into the system under test as superuser and type the appropriate command to start SunDiag (as described in Section 1.4, "Starting the SunDiag Exerciser,").**
   Be sure to obtain display permission (server access) using the `xhost +` *hostname* command. See Chapter 1, "Introducing the SunDiag System Exerciser," for a list of all software and hardware requirements for running the program.

2. **Display the SunDiag window and check the devices shown in the control panel against the devices you know to be physically present in the system. Section 2.1.4, "Control Panel," if there is a discrepancy.**
   Starting the SunDiag exerciser often serves as a quick check for most hardware devices. If you have just installed a device and reconfigured your machine accordingly, the SunDiag test for that device can confirm proper installation.

   Likewise, if the SunDiag environment fails to display a device you know is physically present in your system, there is a problem, and you should recheck your installation carefully.

3. **Choose the global options from the Set SunDiag Options menu.**
   These include both testing parameters and scalability options.

   You can create a special global options file. You can load this file before you begin testing, or you can specify the file with the `-o` option when starting the SunDiag exerciser.

4. **Choose options from the individual test option menus.**
   If you want to exercise any different options from those set globally in the Set SunDiag Options menu, click the Options button next to the test name to display individual test option menus. These menus include individual test parameters, test-specific options, and scalability options.

5. **Click the Start Button.**
   Or, if you have enabled the Auto Start option from the Set SunDiag Options menu and saved an options file, type **`sundiag -o`** *options_filename* from a command line.

Part 2— SunDiag Tests

SunDiag Test Descriptions 6

This chapter describes tests that run on machines with a SPARC-based architecture. These tests will be displayed under the following sections on the SunDiag Control Panel:

Memory Devices	
`pmem` (Read-only Physical Memory Test)	*page 6-2*
`vmem` (Write / Read / Compare Virtual Memory Test)	*page 6-3*
CPU Devices	
`fputest` (FPU Test)	*page 6-6*
`nettest` (Network Interface Test)	*page 6-7*
`autest` (Audio Hardware Test)	*page 6-13*
`mptest` (Multiprocessing Test)	*page 6-14*
Disk Devices	
`rawtest` (Disk Read / Write / Compare Test)	*page 6-20*
`fstest` (Disk Filesystem Test)	*page 6-24*
`cdtest` (Compact Disc Test)	*page 6-28*
Tape Devices	
`tapetest` (Tape Drive Test)	*page 6-32*
SBus Devices	
`fbtest` (Framebuffer Test)	*page 6-39*
`sptest` (Serial Ports Test)	*page 6-40*

6.1 *Physical Memory Test* (`pmem`)

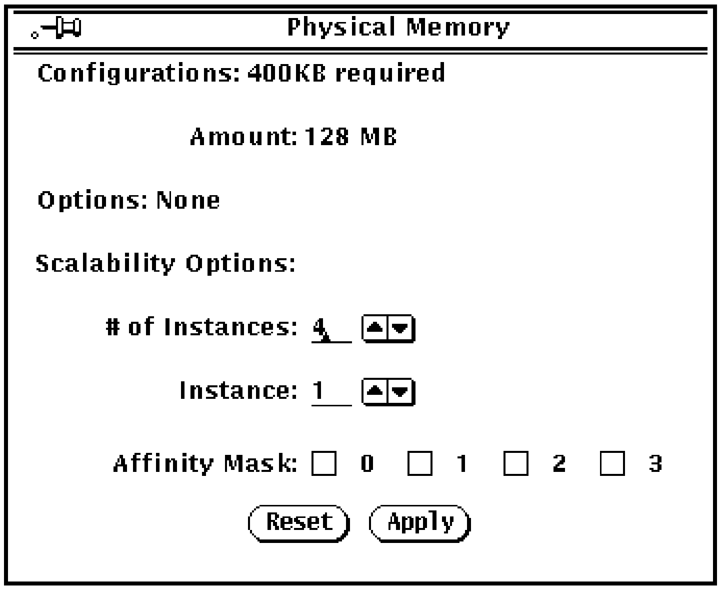
This test checks the physical memory of the system.

6.1.1 `pmem` *Test Description*

`pmem` locates parity errors, hard and soft error correction code (ECC) errors, memory read errors, and addressing problems. Through the memory device `/dev/mem`, `pmem` maps and then reads a page repeatedly throughout memory.

6.1.2 `pmem` *Option Menu*

Click the Options window button for the `pmem` test to display the window shown in Figure 6-1.


Physical Memory

Configurations: 400KB required

Amount: 128 MB

Options: None

Scalability Options:

# of Instances: 4

Instance: 1

Affinity Mask: ☐ 0 ☐ 1 ☐ 2 ☐ 3

Figure 6-1 `pmem` Option Menu

6.1.3 `pmem` *Configurations*

The amount of memory shown in the Configurations field is the total physical memory probed by the kernel. It reflects the amount of memory found, rounded up to the nearest megabyte by any fraction.

6.1.4 `pmem` *Command Line Syntax*

`/opt/SUNWdiag/bin/pmem s=`*size* *standard_arguments*

Arguments	
`s`=*size*	*size* is the amount of memory to be tested in page frames. For example, a system with 32 Mbytes of physical memory would have 8192 page frames of 4096 bytes.

6.1.5 `pmem` *Quick Test Description*

Specifying the `q` (quick test) option from the *standard_arguments* does not change the `pmem` test procedure.

6.2 *Virtual Memory Test* (`vmem`)

This test checks virtual memory; that is, it tests the combination of physical memory and the swap partitions of the disk(s).

Note – This test may not stop immediately after being disabled.

6.2.1 *Swap Space Considerations*

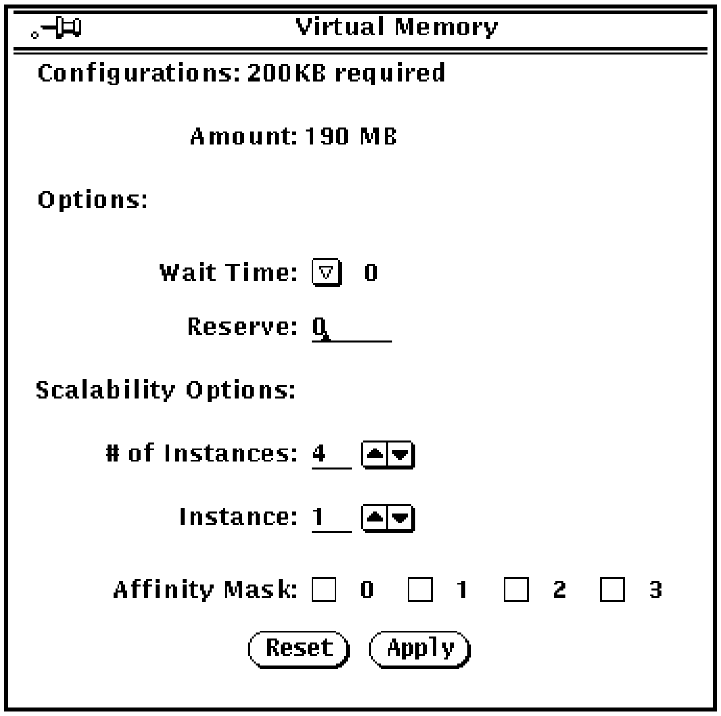
Running this test puts a significant burden on the operating system, since it uses the majority of swap space available for testing. You should use the swap space reserve option to `vmem` when it is run with other tests or processes. See “Swap Space Requirements” on page 1-7 for a complete discussion of swap space requirements. In addition, setting the Wait Time option can help you avoid problems of `vmem` hogging the system memory.

6.2.2 `vmem` *Test Description*

This test uses the Solaris `valloc` (page aligned) system call to allocate, write, and then read as much virtual memory as is feasible. This test also detects ECC parity errors, memory read errors, addressing problems, and displays the corresponding virtual memory addresses on failure.

6.2.3 `vmem` *Option Menu*

Click the Options window button for the `vmem` test to display the window shown in Figure 6-2.



Configurations: 200KB required

Amount: 190 MB

Options:

Wait Time: 0

Reserve: 0

Scalability Options:

# of Instances: 4

Instance: 1

Affinity Mask: ☐ 0 ☐ 1 ☐ 2 ☐ 3

Figure 6-2 `vmem` Option Menu

6.2.4 `vmem` *Configurations*

The amount of memory listed in the Configurations section is equivalent to the sum of the used and available swap space amounts returned by the `swap -s` command. It reflects the amount of virtual memory found, rounded up to the nearest megabyte by any fraction.

6.2.5 `vmem` *Options*

Wait Time

The Wait Time option controls the time interval between `vmem` sessions. The available time intervals are 0, 5, 10, 15, 30, 60, 90, or random minutes. Specifying "random" will use time intervals between 0 and 90 minutes for each pass.

Reserve

The reserve option specifies the amount of memory to reserve from being tested by `vmem`. The reserved space is used for other processes or SunDiag tests running concurrently. The reserve option can be used to reserve memory in addition to the default.

6.2.6 `vmem` *Command Line Syntax*

/opt/SUNWdiag/bin/vmem U M=*pattern_modifier* **m=***#_of_MB* **R=***#_of_MB*
page cerrs=*#_of_errors* **nerrs=***#_of_errors* *standard_arguments*

Arguments	
U	Returns an enhanced description of the vmem test, but does not run the test.
M=*pattern_modifier*	pattern_modifier can be any integer. The integer defines the pattern used in the vmem test.
m=*#_of_MB*	*#_of_MB* is the number of megabytes of memory to reserve for the operating system instead of the default amount.
R=*#_of_MB*	*#_of_MB* is the number of megabytes of memory to reserve in addition to the default amount.

Arguments	(Continued)
`page`	Tells the write/read memory to proceed one page at a time.
`cerrs`=*#_of_errors*	Tells the test to simulate a number of contiguous errors.
`nerrs`=*#_of_errors*	Tells the test to simulate a number of noncontiguous errors.

6.2.7 `vmem` *Quick Test Description*

Specifying the `q` (quick test) option from the *standard_arguments* does not change the `vmem` test procedure.

6.3 Floating Point Unit Test (`fputest`*)*

Note – This test is not supported by machines with x386 or x486 processors.

This test checks the floating point unit on machines with SPARC architecture. It performs the following subtests:

- FSR Register test
- Registers test
- NACK test
- Move Registers test
- Positive to Negative test
- Negative to Positive test
- Absolute test
- Single Precision Integer to Floating Point test
- Double Precision Integer to Floating Point test
- Single Precision Floating Point to Integer test
- Double Precision Floating Point to Integer test
- Single Precision Round Toward Zero test
- Double Precision Round Toward Zero test
- Single to Double Precision Format Conversion test
- Double to Single Precision Format Conversion test
- Single and Double Precision Addition, Subtraction, Multiplication, Square-root, Division and Compare tests
- Single and Double Precision Compare and Exception if Unordered tests
- Branching and no Branching on Condition Instructions tests
- Single and Double Precision Chaining tests
- Weitek Status tests

- Lock test
- Single and Double Precision Datapath tests
- Timing (load) test
- Linpack test

6.3.1 `fputest` *Command Line Syntax*

`/opt/SUNWdiag/bin/fputest` *standard_arguments*

6.3.2 `fputest` *Quick Test Description*

Specifying the `q` (quick test) option from the *standard_arguments* does not change the `fputest` test procedure.

6.4 *Network Hardware Test (*`nettest`*)*

Note – This new version of `nettest` will be used for *all* networking devices, including Ethernet (`ie` and `le`), token ring (`tr`), quad ethernet (`QED`), fiber optic (`fddi`), SPARCcluster™ 1 System (`em`), and 100 megabits per second ethernet (`be`) devices.

This test checks all the networking hardware on the system CPU board and separate networking controllers (for example, a second SBus Ethernet controller). For this test to be meaningful, the machine under test must be attached to a network with at least one other system on the network.

6.4.1 `nettest` *Test Description*

`nettest` mainly uses the internet control message protocol (ICMP), and requires at least two machines on a network — the machine under test and another machine reliable enough to be a test target. Both machines must support the transport control protocol/interface program (TCP/IP) (ICMP is part of TCP/IP). The target machine must either be configured to respond to ICMP broadcast or to RPC broadcast.

The first thing `nettest` does is to determine the target machine(s) to test against. If no targets are specified, it will send an ICMP broadcast to find them. If it fails to find all necessary targets, it will try RPC broadcast to the RPC port mapper daemon. If you specify the targets, `nettest` uses the specified target(s) instead.

Random, Incremental, and Pattern

After finding the necessary targets, `nettest` performs the Random test, the Incremental test, and the Pattern test.

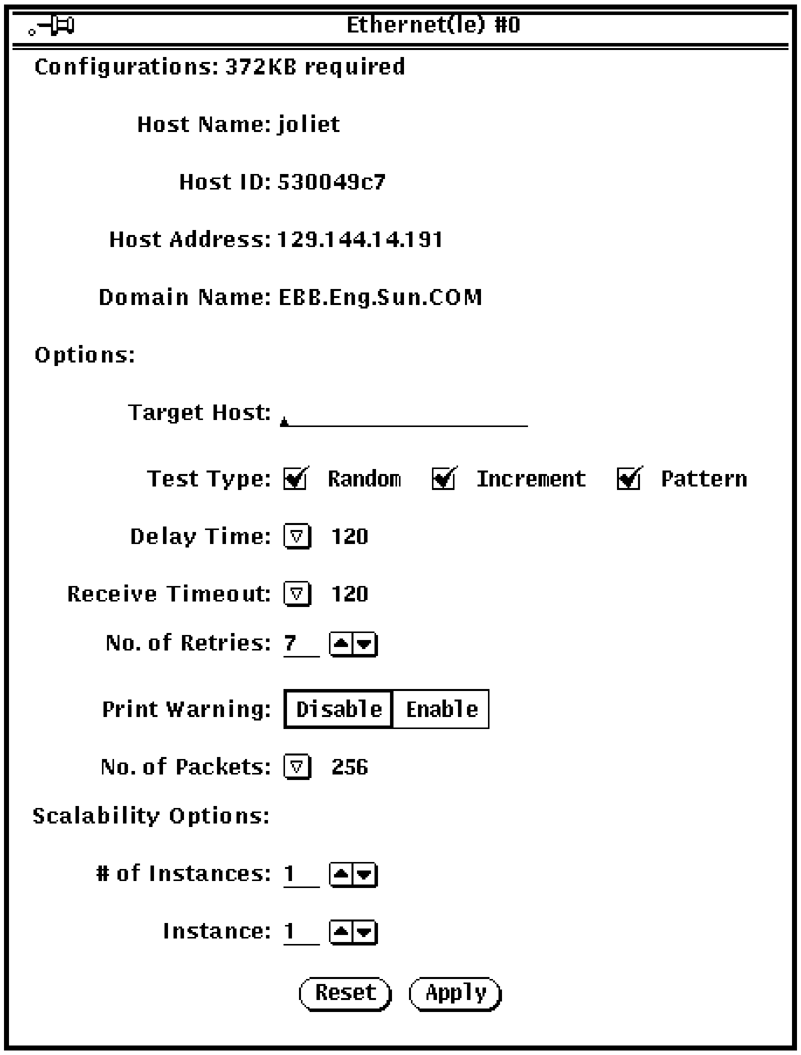
The Random test sends out 256 packets with random data length and random data.

The Incremental test sends out packets with length from minimum to maximum packet size using incremental data. (Minimum and maximum values differ for each device.)

The Pattern test sends 256 packets of maximum length, where each packet contains one test pattern, and all byte patterns (0 to 0xFF hex) will be used. That is, first packet contains pattern 0, second packet contains pattern 1, and so on, until the last packet pattern of 0xFF.

Note – `nettest` is a scalable test. However, the maximum number of networked devices allowed on a system is 255, and the number of instances for each device is limited to 200. So, if you start the SunDiag exerciser using the `-i` option to specify a default number of instances for all tests, note that `nettest` cannot assign more than 200 instances per each networked device.

6.4.2 `nettest` *Option Menu*



Ethernet(le) #0

Configurations: 372KB required

Host Name: joliet

Host ID: 530049c7

Host Address: 129.144.14.191

Domain Name: EBB.Eng.Sun.COM

Options:

Target Host:

Test Type: ✓ Random ✓ Increment ✓ Pattern

Delay Time: 120

Receive Timeout: 120

No. of Retries: 7

Print Warning: [✓ Disable] Enable

No. of Packets: 256

Scalability Options:

# of Instances: 1

Instance: 1

Figure 6-3 `nettest` Option Menu

6.4.3 `nettest` *Configurations*

The Configurations section of the `nettest` Options menu describes the Host Name, Host ID, Host Address, and Domain Name of the system under test.

6.4.4 `nettest` *Options*

The Target Host field specifies one or more targets to be tested against. Target host entries may either be a host name or an internet address. When no target host is specified, the test will find necessary targets via broadcasting. The default setting leaves this field empty.

Use the check boxes to select either the Random, Increment, or Pattern tests. All three tests are selected as a default.

The Delay Time field default is 120 seconds, but it can be changed. Choose a value from this list: 10, 30, 60, 120, 180, 240, 300, 360, 420, 480, 540, 600, or 1200.

The Receive Timeout field default is 120 seconds, but it can be changed. Choose a value from this list: 10, 30, 60, 120, 180, 240, 300, 360, 420, 480, 540, or 600.

The No. of Retries field default retry is 7 retries before flagging an error. This field can also be changed — use a range between 0-128 retries.

The Print Warning setting is disabled by default. Click on Enable to see warning errors, such as retry on timeout.

The No. or Packets field default is 256. You can choose another field value from this list: 256, 512, 1024, 2048, 4096, 8192, 16000, 32000, or 64000.

6.4.5 `nettest` *Command Line Syntax*

`/opt/SUNWdiag/bin/nettest [TARGET=`*h1*`+`*h2*`+...] [IF=`*interface*`] [TEST=`*n*`] [N=`*nopkts*`] [P=`*pattern*`] [D=`*seconds*`] [T=`*seconds*`] [R=`*retries*`] W` *standard_arguments*

Arguments	
`[TARGET=`*h1*`+`*h2*`+...]`	Specify a list of test targets, by hostname or internet address.
`[IF=`interface**`]`**	Network interface name. The default value is **`le0`** for Ethernet networks.
`[TEST=`*n*`]`	Specifies the test type. Type **`1`** for only the Random test, **`2`** for the Incremental test, **`3`** for the Pattern test. The default value is **`7`**, which specifies all tests to be run.
`[N=`*nopkts*`]`	Number of random / pattern packets. The default is 256.
`[P=`*pattern*`]`	Specifies a data pattern, in hexadecimal form. The default is all patterns from `0` to `0xff`.
`[D=`*seconds*`]`	Delay time between subtests in seconds. The default is 30 seconds.
`[T=`*seconds*`]`	Number of seconds to wait before time-out. The default is 30 seconds.
`[R=`*retries*`]`	Number of test timeout retries. The default is 7 retries.
`W`	Print warning messages.

6.4.6 `nettest` *Quick Test Description*

If you specify the `q` (quick test) option from the *standard_arguments,* `nettest` runs only the Random test with 256 packets.

6.4.7 `nettest` *Error Messages*

`Unable to find any test target`

No target is found, test can’t be done. This test requires a reliable target host be connected to the network.

`ICMP echo reply timeout from` <*target_hostname*>

Transmitted packet is lost or the target host couldn’t receive, or the target doesn’t respond, and the machine under test has tried three times without success.

`ICMP echo reply incorrect length, exp %d obs %d`

Packet is corrupted.

`ICMP echo reply length %d, data mismatch at byte %d, exp 0x%02x obs 0x%02x`

Packet data is corrupted at the indicated byte offset.

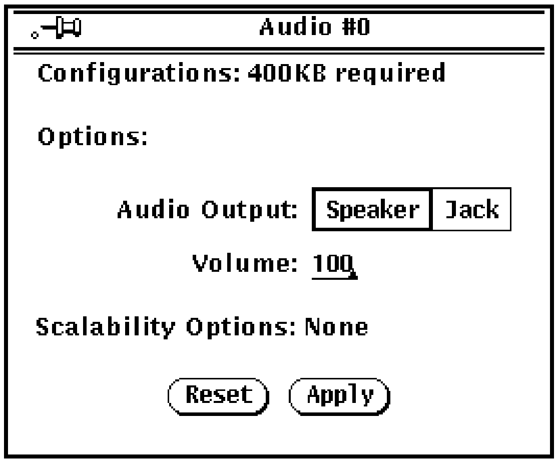
6.5 Audio Hardware Test (autest)

Note – This test is not supported by machines with x386 or x486 processors.

This test checks the MAP registers in the audio chip.

6.5.1 autest Option Menu

Clicking the `autest` Options window button displays the window shown in Figure 6-4.



Audio #0

Configurations: 400KB required

Options:

Audio Output: [✓ Speaker] Jack

Volume: 100

Scalability Options: None

Figure 6-4 `autest` Option Menu

You can set volume and output to specified options and play whatever is in the `autest.data` file. You set the volume by entering 0 to 255 in the volume field. The Audio Output setting toggles between Speaker and Jack. The default is Speaker, which sends audio output to the system's built-in speaker.

Note – Some systems, such as the SPARCstation 10, SPARCstation LX, and SPARCclassic systems from Sun Microsystems, use an external speaker for audio output. Refer to the hardware-specific information that came with your system.

6.5.2 `autest` *Command Line Syntax*

`/opt/SUNWdiag/bin/autest V=`*volume* **`O=`***output standard_arguments*

Arguments	
`V=`*volume*	Specifies the audio volume. The range is 0 through 255.
`O=`*output*	Specifies the audio output. The choices are **`speaker`** or **`jack`**.

6.5.3 `autest` *Quick Test Description*

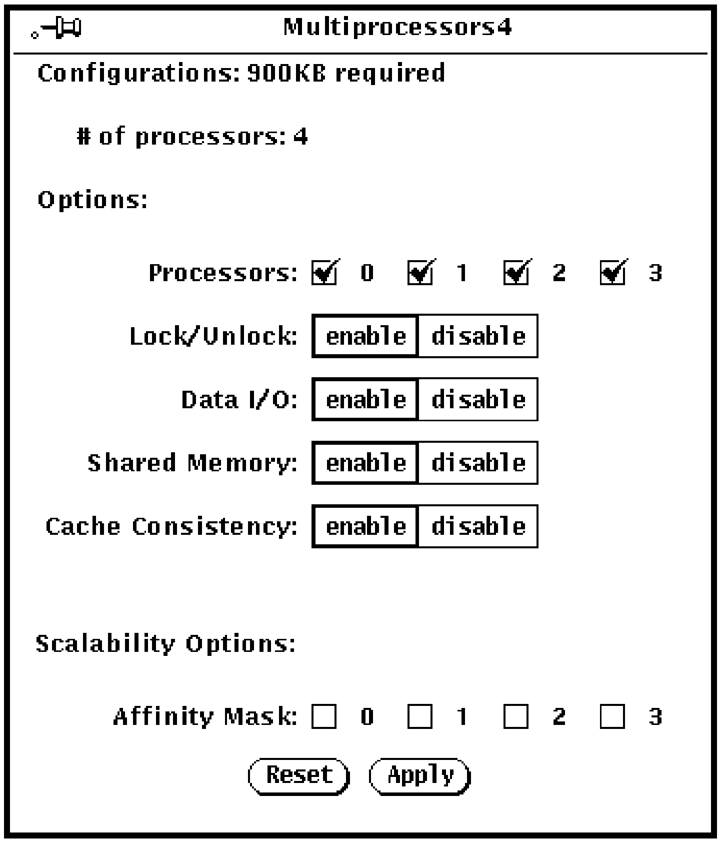
Specifying the `q` (quick test) option limits the test data size length to 0x1000.

6.6 *Multiprocessing Test* (`mptest`)

6.6.1 `mptest` *Test Description*

The SunDiag Multiprocessor test verifies the functionality of multiprocessing hardware. This test starts by allocating a page of virtual memory for the test, declaring the page shared, locking the page against swapping, and forking child processes to each of the processors being tested. This test uses those child processes to verify that the processors are functioning properly.

6.6.2 `mptest` *Option Menu*


Multiprocessors4

Configurations: 900KB required

# of processors: 4

Options:

Processors: ✓ 0 ✓ 1 ✓ 2 ✓ 3

Lock/Unlock: [✓ enable] disable

Data I/O: [✓ enable] disable

Shared Memory: [✓ enable] disable

Cache Consistency: [✓ enable] disable

Scalability Options:

Affinity Mask: ☐ 0 ☐ 1 ☐ 2 ☐ 3

Figure 6-5 `mptest` Option Menu

`mptest` *Configurations*

The processors available for test are listed here. The multiprocessing test can be enabled or disabled for individual processors on this menu.

6.6.3 `mptest` *Options*

The following options can be run individually or concurrently:

Processors

This option allows you specify which processors to test. Click SELECT on the check boxes to enable or disable each processor. A check mark means the processor is enabled for testing; the default setting is all processors enabled. Note that `mptest` requires at least two enabled processors to test multiprocessing systems.

Lock/Unlock

This option tests the lock/unlock mechanism which guarantees exclusive access to a physical page to one processor. A child process is forked to each of the processors. Each processor uses the SPARC atomic instruction `ldstub` to write to the same shared physical memory page. While one processor is attempting the write, the other processors should be free spinning for their turn. As each processor acquires the lock, it writes an ordinal number to a shared tracebuffer using a shared write pointer. After the test cycle is complete, the tracebuffer is dumped for analysis.

This test fails and returns an error message if the trace buffer does not contain an equal number of ordinal numbers for each processor. For example, if the specified loop count is 5, the trace buffer should contain five 0s, five negative 1s, five 2s, etc.

Data I/O

This test requires two or more processes, each of which locks onto one of the processors. Each processor, in turn, writes data to a temporary file that has been mapped to the physical address. The modified data is immediately read by other processors being tested. This test will hang and fail if the processors do not recognize the expected data.

Shared Memory

A shared memory buffer is divided into a number of contiguous chunks, one for each of the CPUs participating in the test. Each CPU is assigned a unique chunk based upon its ID (1-N).

This subtest has two parts. First, each CPU locks and writes data to its data chunk. The data written is identical for each CPU. Then each CPU reads and compares the information on its data chunk with that of another CPU.

If two CPUs do not confirm consistent data, the test will fail and return an error message. If that happens, testing stops, and this test is run again in verbose mode to return more detailed information.

Cache Consistency

This option requires two or more processors to access and write to the same physical address. This test verifies that a change in physical address by one processor is confirmed by another.

If two processors do not confirm consistent data, the test will continue to run, but the Pass Count in the SunDiag status window will stop incrementing. If that should happen, stop testing and run this test again in verbose mode for a more detailed picture of the problem.

Scalability Options

See Section 4.1, “Scalability Options.”

6.6.4 `mptest` Command Line Syntax

`/opt/SUNWdiag/bin/mptest` **`T=`***n* **`C=`***n* *standard_arguments*

Arguments	
`T`=*n*	*n* is one of the following integers, representing the bit pattern of the subtests to be enabled. If **`T=`** is not specified, the default is to enable all options. 1 = lock/unlock test 2 = Data I/O test 3 = lock/unlock and Data I/O test 4 = Shared memory test 5 = lock/unlock and Shared memory test 6 = Data I/O and Shared memory test 7 = lock/unlock and Data I/O and Shared memory test 8 = Cache consistency test 9 = lock/unlock and Cache consistency test 10 = Data I/O and Cache consistency test 11 = lock/unlock and Data I/O and Cache consistency test 12 = Shared memory and Cache consistency test 13 = lock/unlock and Shared memory and Cache consistency test 14 = Data I/O and Shared memory and Cache consistency test 15 = lock/unlock and Data I/O and Shared memory and Cache consistency test
`C`=*n*	*n* is one of the following integers, representing the bit pattern of the processors to be enabled. If **`C=`** is not specified, the default is to enable all available processors. 1 = processor0 Enabled 2 = processor1 Enabled 3 = processor0 and processor1 Enabled 4 = processor2 Enabled 5 = processor0 and processor2 Enabled 6 = processor1 and processor2 Enabled 7 = processor0, processor1, and processor2 Enabled 8 = processor3 Enabled 9 = processor0 and processor3 Enabled 10 = processor1 and processor3 Enabled 11 = processor0, processor1, and processor3 Enabled 12 = processor2 and processor3 Enabled 13 = processor0, processor2, and processor3 Enabled 14 = processor1, processor2, and processor3 Enabled 15 = processor0, processor1, processor2, and processor3 Enabled

6.6.5 `mptest` *Quick Test Description*

The quick option to `mptest` is not available. If this option is enabled, `mptest` runs normally.

6.6.6 `mptest` *Error Messages*

`mptest` may return one of the following error messages. These messages specify problems with specific command-line arguments:

`ERROR: Must be super-user`

You must start SunDiag as superuser.

`Please specify the ’T=’ argument again`

The integer specified for the “T=*n*” argument was not acceptable; you must specify another. See 6.6.4“mptest Command Line Syntax” for a list of acceptable choices.

`Can not run mptest! You need to enable at least two processors.`

There was an error in the non Multiprocessor environment. You must add another processor using the `mptest` Option Menu, or the “C=n” option.

`Not an MP system!`

The system being tested does not have Multiprocessing capability.

`FAIL: MIOCSPAM, mask =` <*mask value representing number processors*>

The kernel support `ioctl` failed to get the correct processor affinity mask.

`FAIL: to open /tmp/iotest.`<*processID*>

`/tmp/iotest.`<*processID*> could not be opened for the Data I/O test.

`FAIL: to open /dev/null`

`/dev/null` could not be opened.

`FAIL: to open /dev/mem`

`/dev/mem` could not be opened for the Lock/Unlock test, the Data I/O test, or both.

`FAIL: to fork!`

A new process could not be spawned.

`FAIL: Lock/Unlock: mmap address space to device`

SunDiag could not map the virtual address to the physical address for the Lock/Unlock test.

`FAIL: Cache Consistency Test: mmap address apace to device`

SunDiag could not map the virtual address to the physical address for the Cache Consistency test.

`FAIL: Data I/O Test: mmap address to space to device`

SunDiag could not map the virtual address to the physical address for the Data I/O test.

`FAIL: <single/double> precision FPU test, process` <*process number*>`, processor` <*processor number*>

FPU test failed on a specified process and processor.

`FAIL: in function getnextbitmsk!`

The `getnextbitmsk` function fails. This function will search the next higher order bit for the mask and base given.

`Lock/Unlock Test: Trace buffer is corrupted due to broken lock.`

The processors being tested are not correctly locking or unlocking a child process.

6.7 Disk Test (`rawtest`)

This test performs read-only or read-write tests on a 'raw' disk partitions. This test treats a disk as one large chunk of contiguous data, which it will attempt to read. In order to test the structural elements (filesystems) on a disk, see “File-system Test (fstest)” on page 6-24.

Caution – If a power failure occurs while `rawtest` is being run in read-write mode, disk data will be destroyed.

6.7.1 `rawtest` Test Description

`rawtest` performs 64 Kbytes of read and/or write in a loop until the total number of data selected through the Rawtest Size option is reached (see Figure 6-6).

`rawtest` is a scalable test; you can run multiple copies of it in read-write mode on the same disk partition. To avoid data corruption, all simultaneous instances of `rawtest` communicate through a shared memory service. This ensures that the different copies of `rawtest` don’t overlay the same 64 Kbyte block at the same time.

`rawtest` will not perform the read-write test on a mounted disk partition because doing so risks the danger of corrupting data integrity on the disk. Conversely, `fstest` *requires* a disk partition to be mounted for testing and will mount and unmount partitions during testing (see the `fstest` description). Consequently, there is a possibility that `rawtest` may fail on a partition because it is mounted by an instance of `fstest` that is running concurrently.

This test supports most types of disk drives, such as SCSI disks, native or SCSI floppy diskettes, and so on. The type of drive under test is named at the top of the options menu.

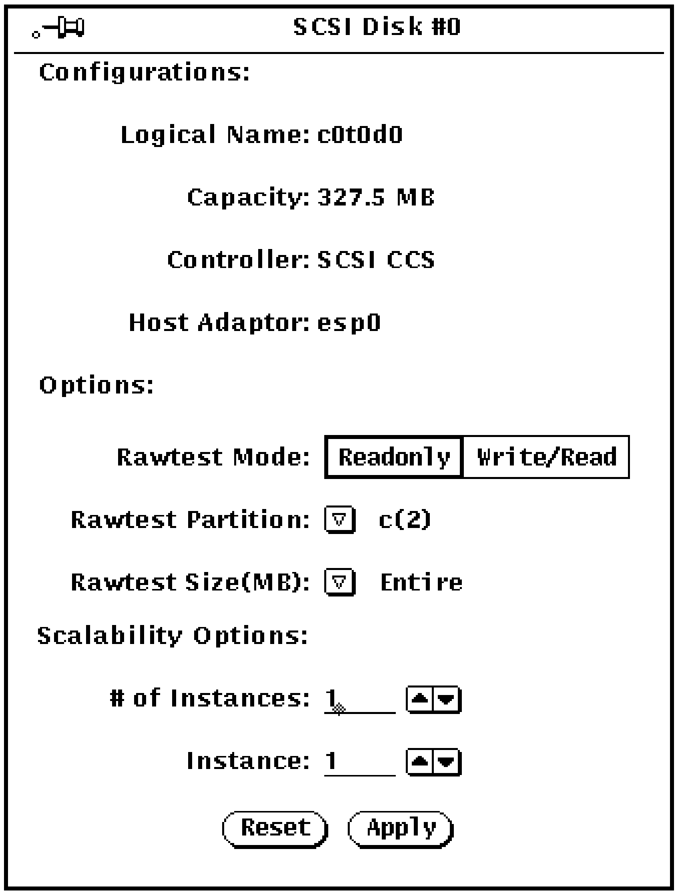
Note – `fstest` will fail if `rawtest` is running in read-write mode on the same disk. It is recommended that you set up `fstest` and `rawtest` so that they don't generate any conflict during system testing.

Volume Management Software and Diskette Drives

`rawtest` tests the diskette drive(s) regardless of whether the Volume Management software is installed and running. If the Volume Management software is installed, `rawtest` will test the diskette drive with device name `D=/vol/dev/aliases/floppy0`. If the Volume Management software is not installed, `rawtest` tests the diskette drive with device name `D=/dev/diskette`.

Do not edit the `/etc/vold.conf` file to change the logical name of CD-ROM (`cdrom0`) and diskette drives (`floppy0`). Currently, the SunDiag software is hard-wired to use these pathnames as the default logical name.

6.7.2 `rawtest` *Option Menu*


SCSI Disk #0

Configurations:

Logical Name: c0t0d0

Capacity: 327.5 MB

Controller: SCSI CCS

Host Adaptor: esp0

Options:

Rawtest Mode: [✓ Readonly] Write/Read

Rawtest Partition: c(2)

Rawtest Size(MB): Entire

Scalability Options:

# of Instances: 1

Instance: 1

Figure 6-6 `rawtest` Option Menu

6.7.3 `rawtest` *Options*

Rawtest Mode

Choose read-only or read-write testing. The default is read-only. In read-only mode, `read()` is the only operation performed. In read-write mode, the data on disk is first backed up in memory, and then a known pattern of data is written to disk and read back for comparison. The original data is restored and the test proceeds on to the next 64 Kbyte of data.

Rawtest Partition

Specifies the device partition to test. The default is partition `c(2)`.

Rawtest Size

Specify the Rawtest Size option parameter; the default setting is the entire partition size (in megabytes). If you specify a value for Rawtest Size that is smaller than the partition size, `rawtest` will randomly pick a contiguous data area of that size within the partition for testing. Hence, each pass of `rawtest` exercises different areas within the partition.

6.7.4 `rawtest` *Command Line Syntax*

`/opt/SUNWdiag/bin/rawtest D=`*device_name* **`W P=`***partition* **`S=`***size*
standard_arguments

Arguments	
`D=`*device_name*	*device_name* is the device to be tested. See “Volume Management Software and Diskette Drives” for specific path names when running `rawtest` with the Volume Management software.
`W`	This option causes `rawtest` to be run in read-write mode; the default (without this option) is read-only.
`P`=*partition*	*partition* is the partition to be tested. The choices are `a`, `b`, `c`, `d`, `e`, `f`, and `g`; the default is partition `c`.
`S`=*size*	*size* is the size of the partition (100MB-1200MB). The default is the entire partition. (See Rawtest Size above).

6.7.5 `rawtest` *Quick Test Description*

Specifying the **q** (quick test) option from the *standard_arguments* restricts this test to a maximum of 64 KBytes of data on the chosen partition.

6.8 File-system Test (`fstest`)

This test verifies the integrity of the software structural elements (file systems) of a disk. In contrast, `rawtest` treats disks as one large chunk of contiguous data, which it attempts to read.

6.8.1 `fstest` *Test Description*

This test exercises the disk controller and drive using the Solaris filesystem device driver. It writes two 0.5 Mbyte files with a specified data pattern, then reads and compares them.

`fstest` requires the partition under test to be mounted. If it is not already mounted, `fstest` will mount the partition, perform the test, and unmount the partition when the test is complete. The mount point bears the name of the disk partition appended with a system-wide unique number. For example, if the disk name is `/dsk/c0t3d0`, `fstest` will mount it as superuser under the name `c0t3d0.`*XXXXXX* where *XXXXXX* is a six digit system-wide unique number.

`fstest` is a scalable test; it is possible to run multiple copies of `fstest` simultaneously on the same disk partition. If that partition is not mounted, the first `fstest` to run will mount the partition, and the final `fstest` will unmount that partition.

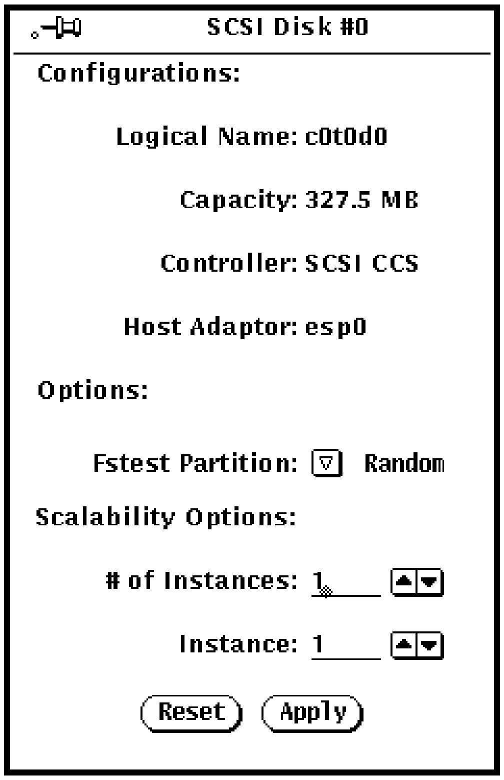
Note – `fstest` will fail if `rawtest` is running in read-write mode on the same disk. It is recommended that user set up `fstest` and `rawtest` so that they don't generate any conflict during system testing.

Volume Management Software and Diskette Drives

`fstest` tests the diskette drive(s) regardless of whether the Volume Management software is installed and running. If the Volume Management software is installed, `fstest` will test the diskette drive with device name `D=/vol/dev/aliases/floppy0`. If the Volume Management software is not installed, `fstest` tests the diskette drive with device name `D=/dev/diskette`.

Do not edit the `/etc/vold.conf` file to change the logical name of the diskette drive (`floppy0`). Currently, the SunDiag exerciser is hard-wired to use these pathnames as the default logical name.

6.8.2 `fstest` *Option Menu*


SCSI Disk #0

Configurations:

Logical Name: c0t0d0

Capacity: 327.5 MB

Controller: SCSI CCS

Host Adaptor: esp0

Options:

Fstest Partition: Random

Scalability Options:

# of Instances: 1

Instance: 1

Figure 6-7 `fstest` Option Menu

6.8.3 `fstest` *Options*

Partition

Select an available partition to run the test on. The selected partition must meet the following three requirements:

1. It must have a file system built on it
2. It must have enough space to hold two 0.5 MByte files
3. There can not be a read-write `rawtest` test running on it (see the `rawtest` description above), or the test will fail.

`fstest` also allows you to randomly test partitions by selecting random as the option value. If random is selected, `fstest` will try to test different partitions, except partitions b(1) and c(2), with each pass. If you want to test partition b(1) or c(2), then you need to manually mount the partition and select either b(1) or c(2) from the Fstest Partition menu.

Note – Since most disks use a partition "C" (which overlays all of the other partitions), running `fstest` on this partition causes problems and is not a recommended procedure. The only exception is if `c` is the only partition on the disk and if you have mounted it manually. The `random` selection does not pick "C" as a candidate for testing. Thus, if you want to test the "C" partition, you must mount the partition manually.

6.8.4 `fstest` *Command Line Syntax*

`/opt/SUNWdiag/bin/fstest` **`D=`***device_name* **`P=`***partition* **`p=`***data_pattern* *standard_arguments*

Arguments		
`D`=*device_name*	*device_name* is the name of the device to be tested. See "Volume Management Software and Diskette Drives" for specific pathnames when running `fstest` with the Volume Management software.	
`P`=*partition*	*partition* is the partition to be tested. The choices are a, b, c, d, e, f, g, h, and random.	
`p`=*data_pattern*	*data_pattern* is the data pattern specified by one of the following arguments:	
	`s`	sequential,
	`a`	all a's,
	`0`	all zeros
	`1`	all ones
	`5`	all fives
	`r`	random.

6.8.5 `fstest` *Quick Test Description*

Specifying the `q` (quick test) option from the *standard_argument*s restricts this test to testing two blocks of data on the specified partition.

6.9 Compact Disc Test (`cdtest`)

This test checks the CD table of contents and when the proper CD is specified, the test verifies that the table of contents matches `cdtest`'s own TOC table. `cdtest` classifies each track as either Mode 1 or Mode 2. In Mode 1 the track uses error detection/correction code (288 bytes), and in Mode 2 it uses that space for auxiliary data, or as an audio track. `cdtest` is not a scalable test.

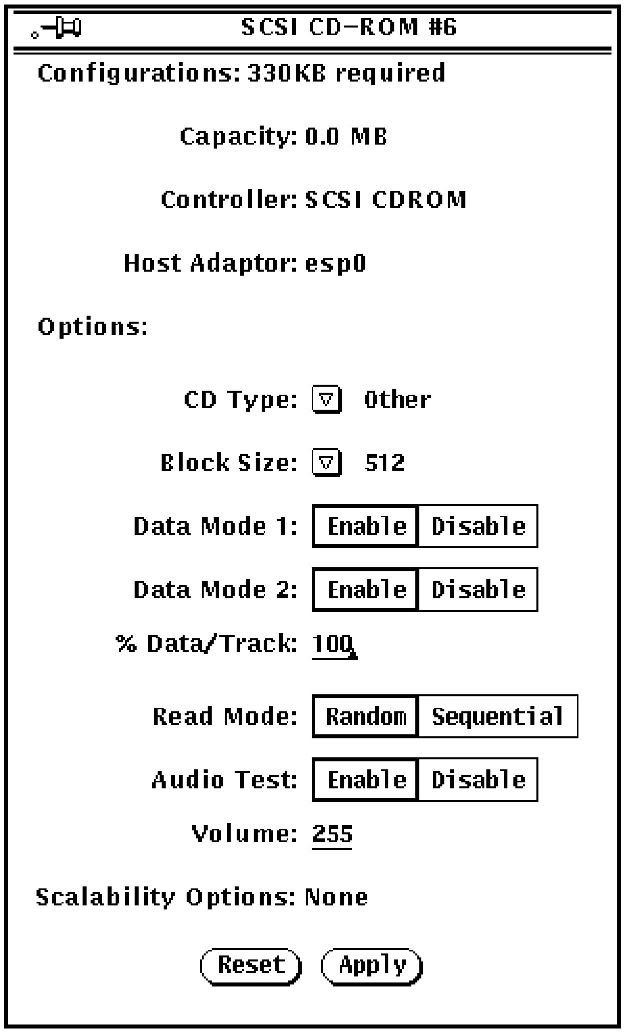
Note – Make sure a scratch CD disc with a well-known table of contents is loaded into the drive before starting this test. It is recommended you use a demonstration CD shipped with the drive, such as the "SunCD™ Demo Disc-1.0" (Sun Microsystems Part No. 704-1000-10.) Do not use an operating system distribution disc. See Section 1.2.8, "Scratch CDs, Tapes, Diskettes" for more information.

Volume Management and Compact Discs

`cdtest` tests the CD-ROM drive(s) regardless of whether the Volume Management software is installed and running. If the Volume Management software is installed, `cdtest` will test the CD-ROM drive with device name `D=/vol/dev/aliases/cdrom0`. If the Volume Management software is not installed, `cdtest` tests the CD-ROM drive with device name `D=/dev/dsk/c0t6d0`.

Do not edit the `/etc/vold.conf` file to change the logical name of the CD-ROM drive (`cdrom0`). Currently, the SunDiag exerciser is hard-wired to use these pathnames as the default logical name.

6.9.1 `cdtest` *Option Menu*


SCSI CD-ROM #6

Configurations: 330KB required

Capacity: 0.0 MB

Controller: SCSI CDROM

Host Adaptor: esp0

Options:

CD Type: Other

Block Size: 512

Data Mode 1: [✓ Enable] Disable

Data Mode 2: [✓ Enable] Disable

% Data/Track: 100

Read Mode: [✓ Random] Sequential

Audio Test: [✓ Enable] Disable

Volume: 255

Scalability Options: None

Figure 6-8 `cdtest` Option Menu

6.9.2 `cdtest` Options

CD Type

Select a type of compact disk to test from the CD Type menu. The choices are: sony2, hitachi4, cdassist, pdo, multi-session, SunOS, or other. The default CD type is other.

Note – Your choice must correspond with the disk used for testing.

Data Mode 1 and Data Mode 2

Click to enable or disable the corresponding track test.

% Data/Track

Type a value between 0 and 100 in this field to test a percentage of data on each track.

Read Mode

Choose between Random or Sequential reading.

Audio Test

Click to enable or disable the audio test. You must connect headphones or a speaker to the audio jack on the CD player to hear audio output.

Volume

Adjust the volume by typing a value between 0 and 255 in this field.

6.9.3 `cdtest` *Command Line Syntax*

`/opt/SUNWdiag/bin/cdtest` **`D=`***device_name* **`S=`***skip_track* **`R=[random | sequential]`** **`P=`***%_of_data* **`V=`***volume* **`T=`***CD_type* *standard_arguments*

Arguments	
`D=`*device_name*	*device_name* is the name of the raw device to be tested. See "Volume Management and Compact Discs" for specific pathnames when running `cdtest` with the Volume Management software.
`S`=*skip_track*	Skip tracks. The choices are **`data1`**, **`data2`** or **`audio`**.
**`R=[random	sequential]`**	Read access.
`P=`*%_of_data*	The percentage of data to be tested. You can specify **`0`** through **`100`** percent.
`V=`*volume*	Audio volume control. You can specify **`0`** through **`255`**. The default is **`255`**.
`T=`*CD_type*	The type of CD used for the test. It can be **`sony2`**, **`hitachi4`**, **`cdassist`**, **`pdo`**, **`multi-session, sunos, other`**. The default is **`other`**.

6.9.4 `cdtest` *Quick Test Description*

Specifying the `q` (quick test) option from the *standard_arguments* causes `cdtest` to test only one block of data. The first block of data is tested if the read access option is set to `sequential`, and one random block is tested if the read access option is set to `random`.

6.10 Tape Drive Test (`tapetest`)

This test first waits for 60 seconds to clear out any bus traffic, then rewinds the tape, erases it, writes a pattern to a specified number of blocks (or, for a SCSI tape, writes to the end of the tape). It then rewinds the tape and reads and compares the information just written. The test expects a device name and number of sectors as input parameters, and first writes to the device from a 126x512 byte buffer, then from a 512-byte buffer for any “leftovers.” `tapetest` is not a scalable test.

Note – If you have a tape drive in your system, load a blank writable tape (scratch tape) before you start the SunDiag exerciser. If you fail to do so, the program will show `drive type:unknown` on the option menu for the tape test.

When running both `tapetest` and a framebuffer test, the following notice window may display:

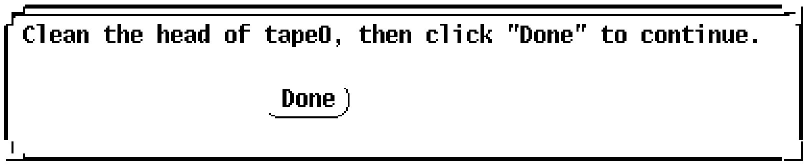
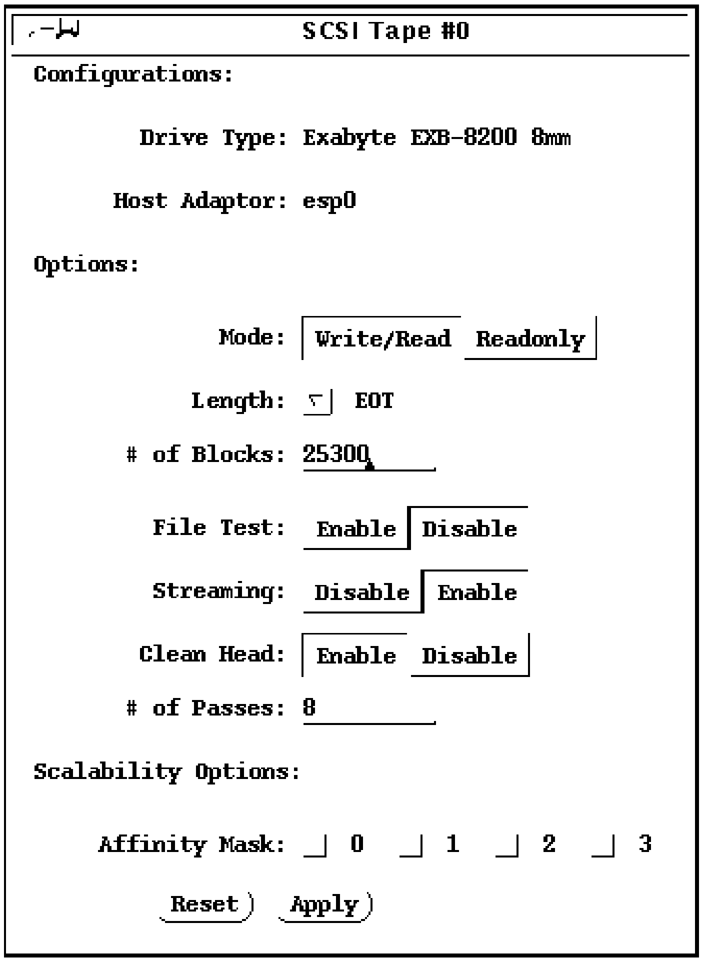

Clean the head of tape0, then click "Done" to continue.

Figure 6-9 `tapetest` Notice Window

Until you click the Done button, the framebuffer test on the same menu screen will not run.

The SunDiag exerciser supports 4mm, 8 mm, 1/4” cartridge, and 1/2” front-load tape drive testing. The options available for each of the tape devices differ slightly. Examples of the option pop-up menus for some devices are shown below.



SCSI Tape #0

Configurations:

Drive Type: Exabyte EXB-8200 8mm

Host Adaptor: esp0

Options:

Mode: Write/Read Readonly

Length: EOT

# of Blocks: 25300

File Test: Enable Disable

Streaming: Disable Enable

Clean Head: Enable Disable

# of Passes: 8

Scalability Options:

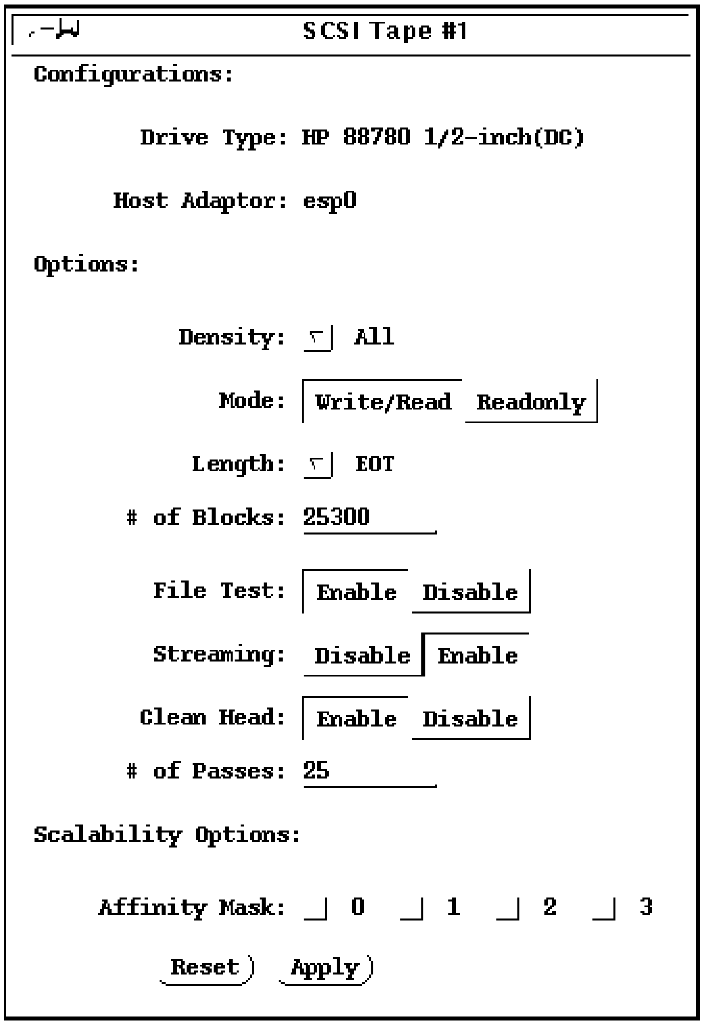
Affinity Mask: 0 1 2 3

Figure 6-10 `tapetest` Option Menu (Exabyte™ 8mm 2.3 GB)

Note – This test may take a while to stop after being disabled.

The example above depicts the options menu for an 8mm tape drive. This menu differs from other tape drive option menus in that it has no format or reconnect option choices.

If the drive is a half inch front-load tape drive, the menu shown in Figure 6-11 will be displayed.



SCSI Tape #1

Configurations:

Drive Type: HP 88780 1/2-inch(DC)

Host Adaptor: esp0

Options:

Density: All

Mode: [✓ Write/Read] Readonly

Length: EOT

# of Blocks: 25300

File Test: [✓ Enable] Disable

Streaming: Disable [✓ Enable]

Clean Head: [✓ Enable] Disable

# of Passes: 25

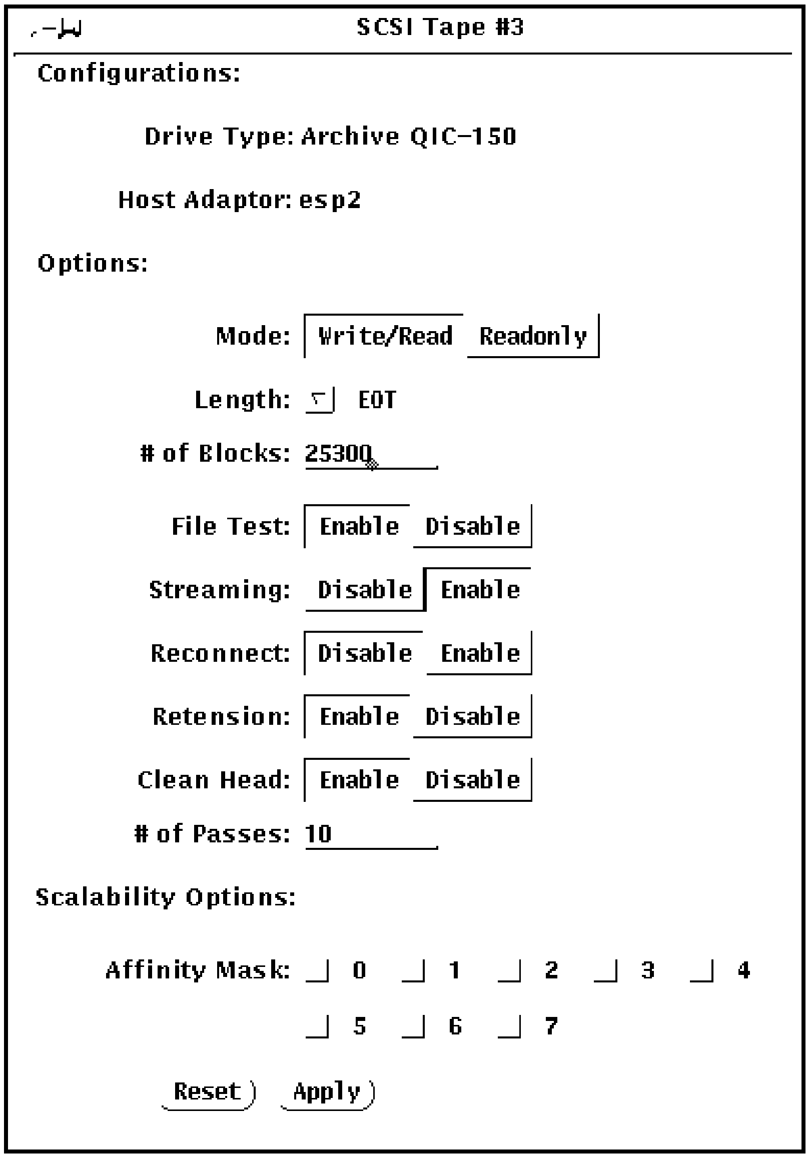
Scalability Options:

Affinity Mask: 0 1 2 3

Reset Apply

Figure 6-11 `tapetest` Option Menu (half-inch front-load)

A quarter-inch tape drive option menu looks like Figure 6-12.



SCSI Tape #3

Configurations:

Drive Type: Archive QIC-150

Host Adaptor: esp2

Options:

Mode: Write/Read Readonly

Length: EOT

# of Blocks: 25300

File Test: Enable Disable

Streaming: Disable Enable

Reconnect: Disable Enable

Retension: Enable Disable

Clean Head: Enable Disable

# of Passes: 10

Scalability Options:

Affinity Mask: 0 1 2 3 4 5 6 7

Figure 6-12 `tapetest` Option Menu (quarter-inch)

6.10.1 `tapetest` *Options*

Format:

QIC-11 and QIC-24 are quarter-inch tape formats that this test uses when it writes to the scratch tape you inserted. QIC-11 uses a 1-byte block ID while QIC-24 uses 4 bytes, meaning that each block on a QIC-24 tape is uniquely identifiable.

Use a standard scratch tape for this test.

QIC-11 format is the default testing format. If you cycle through the options, you may choose QIC-24 only, or both of QIC-11 and QIC-24 formats. If you choose both, the test will first write one pass to the tape in QIC-11 format, and then write a second pass over it in QIC-24 format.

Density

The following settings are available for 8mm Tape Drives:

`EXB8200`	Writes 2.3 Gbytes of data to the tape.
`EXB8500`	Increases the density and writes 5 GBytes of date to the tape.
`Both`	Writes both 2.3 GBytes, and 5 GBytes of data to the tape.

For half-inch tape drives the available settings are `800`, `1600`, and `6250` BPIs.

Mode

If you enable Write/Read mode, the test first writes to the tape and then reads it back to compare. If you enable Readonly mode, the test assumes the tape has been properly written and merely reads and compares. This mode is useful to check proper head alignment.

Length

The amount of the tape to be tested. The choices are:

`EOT`	The default; tests to the entire tape.
`Long`	The SCSI tape tests 70,000 blocks of the tape.
`Short`	Only the first 1000 blocks are tested.
`Specified`	You must type the number of blocks to be tested in the `# of blocks` field.

# of Blocks

If you select Specified under the Length option, you must type a number of blocks you want to test.

File Test

The *tape file* test writes three files, rewinds and then reads part of the first file and forward spaces to the start of the second file, reads the second file, forward spaces to the start of the third file, and tries to read to the end of that file. For SCSI tapes only, the test then tries to back space to the start of the second file and read it.

Streaming

When streaming is enabled, the test performs the Write/Read at top speed to “stream” to tape.

Reconnect

When you enable Reconnect from the options menu, the SunDiag exerciser attempts to check whether or not disconnect/reconnect is working, on a system with a SCSI3 board and both SCSI disk and tape devices. The test forks a child process to test the SCSI disk, which sets up a signal handler to catch a signal from the parent process. It then sleeps for three seconds, reads in two blocks from the disk, sleeps again, and expects to have received a signal from the parent process in the interim. The parent process retensions the tape and then notifies the child.

Retension

When enable is selected, the program retensions the tape.

Clean Head

Click enable to select head cleaning.

of passes

If you have enabled the head cleaning option, you must enter the number of test passes the SunDiag exerciser should execute before suspending testing to provide time to clean the tape drive head.

6.10.2 `tapetest` *Command Line Syntax*

`/opt/SUNWdiag/bin/tapetest D=`*device_name* **`b=`***block_count* **`sq nr ro ns ft rc m=`***x* **`k=`***y* *standard_arguments*

Arguments	
`D=`*unit_number*	*unit_number* is the unit number of the tape device to be tested. For example: **`D=0`** for st0.
`b`=*block_count*	Specifies the number of blocks to be tested.
`sq`	Tells the test to switch the tape format
`nr`	Specifies no retension.
`ro`	Specifies a read-only test.
`ns`	Tells the test no tape sleep.
`ft`	Enables a file test
`rc`	Enables a reconnect test.
`m=`*x*	For 20 Gbyte 4 mm tape auto-loader type tape drives, *x* is the magazine size. The choices are **`4`** or **`12`**.
`k=`*y*	Indicates if the tape drive is a 20 Gbyte 4 mm tape auto-loader type. The choices are: **`0`** - not a 20 Gbyte 4 mm tape auto-loader type drive **`1`** - 20 Gbyte 4 mm tape auto-loader type drive

6.10.3 `tapetest` *Quick Test Description*

If the `q` standard argument is specified, `tapetest` tests three large blocks (512x126 bytes) and two small blocks (512 bytes) of memory, and does not sleep for 60 seconds.

6.11 *Framebuffer Test (*`fbtest`*)*

Note – This test is not supported by machines with x386 or x486 processors.

`fbtest` is a generic test for all dumb framebuffers used with Solaris 2.2.

`fbtest` tests the framebuffer by sequentially writing, reading, and verifying small blocks of random patterns across the entire video RAM. The block size is 64 by 64 pixels. If a mis-compare occurs, the test stops with an error message indicating the location of the error.

If a generic framebuffer device name (`/fb`) is specified, `fbtest` automatically detects the 'depth' of the framebuffer, and adjusts testing to the framebuffer size. There is no option menu for `fbtest`.

6.11.1 `fbtest` *Command Line Syntax*

`/opt/SUNWdiag/bin/fbtest` **`D`**=*device_name* **`L`** *standard_arguments*

Arguments	
`D`=*device_name*	*device_name* specifies which framebuffer to test.
`L`	Disables framebuffer locking. See the "Special Note on Testing Multiple Framebuffers" in Chapter 1 of the SunDiag 4.3 User's Guide for details. Framebuffer locking is enabled by default on the window server running the OpenWindows software.

6.11.2 `fbtest` *Error Messages*

```
Video memory error at 0x%X, expect 0xXXXXXXXX, see
0xXXXXXXXX
```

This error is displayed when `fbtest` finds a mismatch between the data patterns it expects to find.

6.11.3 `fbtest` *Quick Test Description*

Specifying the `q` (quick test) option from the *standard_arguments* does not change the `fbtest` test procedure.

6.12 *Serial Ports Test* (`sptest`)

6.12.1 `sptest` *Test Description*

This test checks the system's on-board serial ports (`zs0`, `zs1`), as well as any Multi-Terminal Interface (ALM2) boards (`mcp0`, `mcp1`, `mcp2`, `mcp3`). It writes data to the source device and then reads it back from the receiver device, verifying the data after each byte sent.

Intervention Mode must be enabled before clicking the option window button for `sptest`. `sptest` is not a scalable test.

6.12.2 `sptest` *Syncloop Testing*

With this release of the SunDiag software, `sptest` has been enhanced to support syncloop testing. Syncloop testing involves the sending and receiving of data packets as HDLC frames over synchronous serial lines.

The syncloop test to `sptest` runs in three phases. The first phase is to listen to the port for any activity. If no activity is seen for at least four seconds, the test proceeds to the next phase. Otherwise, you are informed that the line is active and the test cannot proceed. `sptest` will exit with an error if the line is active.

In the second phase, called the ‘first packet” phase, the test attempts to send and receive one packet. If no packets are seen after five attempts, the test exits with an error message. If a packet is returned, the result is compared with the original. If the length and content of the packets do not match exactly, the test issues an error message.

The third phase is the “multiple packet” phase, where the test attempts to send many packets through the loop. Because the test has verified the integrity of the link in the first packet phase, the test will not fail after a particular number of time-outs. If a packet is not seen after the wait time expires, a message is displayed. The number and size of the packets sent during this phase is determined by a default value. Each packet is compared with its original for length and content. If a mismatch is detected, the test issues an error message.

6.12.2.1 Syncloop Testing Software Requirements

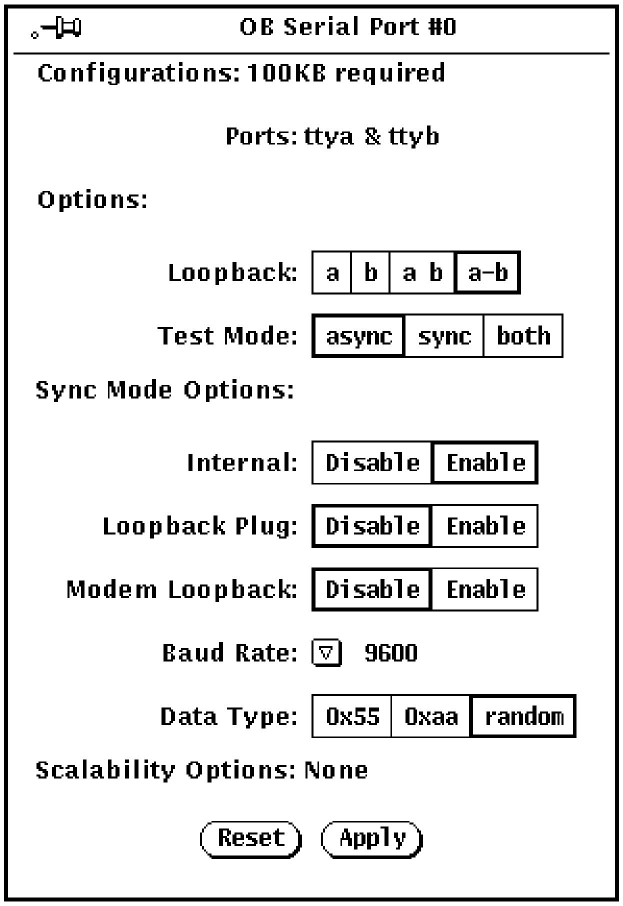
Syncloop testing requires the `zsh0` and `zsh1` devices to be located in `/dev`. If you don’t have these devices configured, make sure the following two lines are in the `/etc/devlink.tab` file:

```
type=ddi_pseudo;name=zshzsh\M0
type=ddi_pseudo;name=clone;minor=zshzsh
```

Warning – The white spaces in the lines above must be a single Tab character before and after the `zsh` variables; using spaces will not work.

When these lines have been added to the `/etc/devlink.tab` file, change directories to `/kernel/drv`, and execute the **`add_drv zsh`** command. If this command does not work at first, execute the **`rem_drv zsh`** command and then execute the **`add_drv zsh`** command again.

6.12.3 `sptest` *Option Menus*



OB Serial Port #0

Configurations: 100KB required

Ports: ttya & ttyb

Options:

Loopback: a | b | a b | [✓ a-b]

Test Mode: [✓ async] | sync | both

Sync Mode Options:

Internal: Disable | [✓ Enable]

Loopback Plug: [✓ Disable] | Enable

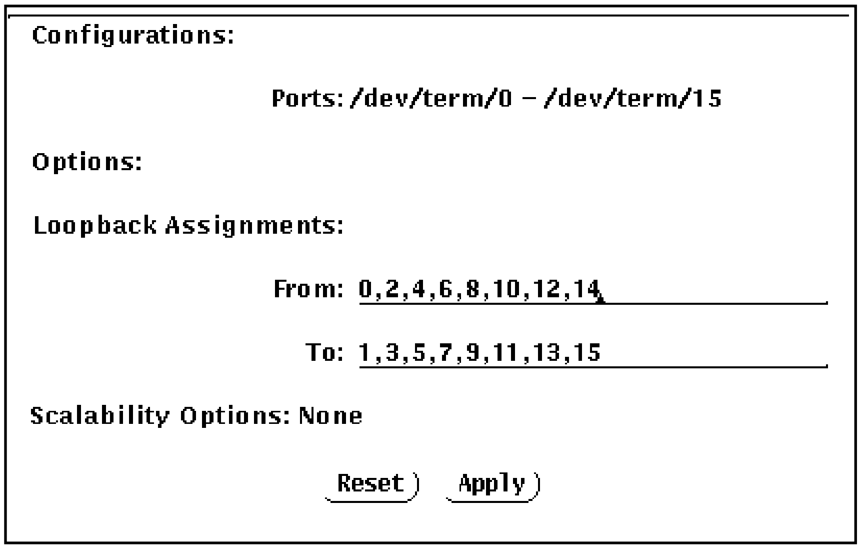
Modem Loopback: [✓ Disable] | Enable

Baud Rate: 9600

Data Type: 0x55 | 0xaa | [✓ random]

Scalability Options: None

Figure 6-13 `sptest` Option Menu



Configurations:

Ports: /dev/term/0 – /dev/term/15

Options:

Loopback Assignments:

From: 0,2,4,6,8,10,12,14

To: 1,3,5,7,9,11,13,15

Scalability Options: None

Figure 6-14 `sptest` Option Menu for Multi-Terminal Interface Boards

6.12.4 `sptest` *Options*

Loopback

The loopback settings refer to serial port testing. You must use loopback connectors (described in Appendix C) to connect the CPU board ports you choose. The default is to link ports A and B with a loopback cable, as indicated by a-b. If you choose a only or b only, the SunDiag exerciser tests just that port, and expects a loopback connector on that port. If you choose a b, the program tests ports A and B separately, and expects a loopback connector on each port.

Loopback Assignments (Multi-Terminal Interface)

The loopback assignments for Multi-Terminal Interface boards are listed here. You must use loopback connectors (described in Appendix C) to test these ports. All ports (from 0 to 15) are listed as the default.

Test Mode

Choose between synchronous or asynchronous testing, or both.

Internal Test

Click Enable to perform a quick internal check of the serial ports on the CPU board. You do not need to connect any loopback connectors to the serial ports to perform this test. This option is available only in synchronous mode.

Loopback Plug

Click Enable to provide data transmission loopback testing of the selected serial port. You must attach a loopback connector (See Appendix C) to the serial port being tested. This option is available only in synchronous mode.

Modem Loopback

Click Enable if the Transmit and Receive clock sources are external (via modem). This option requires that one of the local modems or the remote modems be set in a loopback configuration. This option is available only in synchronous mode.

Baud Rate

Press MENU to select the baud rate. The choices are: 110, 300, 600, 1200, 2400, 4800, 9600, 19200, or 38400 baud. If you have changed the baud rate, the new value will be used. This new value may not correspond to what the Baud Rate menu displays.

Note – You can only change the baud rate if you are testing in synchronous mode. If you are testing in asynchronous mode, the `pstest` will use the default baud rate (usually 9600).

Data Type

Data Type specifies the selected hexadecimal type pattern. The choices are 0x55555555, 0xaaaaaaaa, or random. 0x55555555 and 0xaaaaaaaa are abbreviated on the option menu. This option is only available in synchronous mode.

6.12.5 `sptest` *Command Line Syntax*

`/opt/SUNWdiag/bin/sptest` **`D=`***device_name* **`M=`***mode* **`T=`***subtest_number* **`B=`***baud_rate* **`I=`***loopback_pattern* *standard_arguments*

Arguments	

`D=`*device_name*	*device_name* specifies which serial port(s) to test. There is no default value; you must specify device name such as:
	`zs0 /dev/term/a`
	`zs0 /dev/term/b`
	`zs0 /dev/term/a,b`

`M=`*mode*	The default test mode is asynchronous. Specify one of the following modes: **`s`** - synchronous **`a`** - asynchronous **`b`** - both synchronous and asynchronous
`T=`*subtest_number*	Specify one of the following numbers: **`1`** - Internal Test **`2`** - Test using Loopback Plug **`3`** - Internal Test and Test using Loopback Plug **`4`** - Modem Loopback Test

`B=`*baud_rate*	If you are testing in synchronous mode, you can specify one of the following baud rates:
	`110`
	`300`
	`600`
	`1200`
	`2400`
	`4800`
	`9600`
	`19200`
	`38400`
	Note: If you are testing in asynchronous mode, the default baud rate will be used (usually **`9600`**).

`I=`*loopback_pattern*	The default test data pattern is **`r`**. You may specify the following: **`5`** = 0x55555555 **`a`** = 0xaaaaaaaa **`r`** = random

6.12.6 `sptest` *Quick Test description*

Specifying the `q` (quick test) option from the *standard_arguments* limits the number of bytes per test pass to 100. 3000 bytes per pass is the default. Also, if multiple ports are available for testing, the quick test will only test the first port.

Developing Your Own Tests A

A.1 How to Use This Appendix

A feature of the SunDiag system exerciser is the ability to develop your own tests. If you're familiar with the C programming language, you can now use the libraries and routines in the `/opt/SUNWdiag/lib` and `/opt/SUNWdiag/include` directories to build your own SunDiag tests. You'll use these routines to communicate directly with the SunDiag kernel.

After you've created your own tests, refer to Section 1.9, "Adding Your Own Tests in .usertest," on page 1-17 to learn how to run your tests.

The header file `sdrtns.h` is in the `/opt/SUNWdiag/include` directory, and the library file `libtest.a` is in the `/opt/SUNWdiag/lib` directory. This Appendix is divided into the following sections:

Test Design Guidelines: general guidelines to think about when designing your tests	*page A-2*
Test Implementation Guidelines: description of the SunDiag programming environment, including global variables, special messaging facilities, and reserved error codes	*page A-4*
Requirements: five requirements you must meet to be compatible with the SunDiag kernel	*page A-8*

A.2 Test Design Guidelines

This section provides general guidelines to follow when developing your own SunDiag tests. You should try to incorporate most of these ideas when planning your tests. Most SunDiag tests already included in the `/opt/SUNWdiag/bin` directory adhere to these guidelines.

Options

Each SunDiag test should have two main options — a quick test mode, and a detailed step-by-step mode:

- The quick test mode for periodic system monitoring to ensure that the unit being tested is available and on-line. This mode has a requirement that, if the quick test option is set by the user, the test will run for a maximum of 30 seconds under normal system load.
- The step-by-step mode for detailed information in case the quick test fails to diagnose the device and provide service information.

For example, a quick test for a disk drive could be a read/write/compare test on the diagnostics cylinder. If the quick test fails, a sequence of sub-tests could be used to verify the path of the I/O device. Generally, all I/O devices under UNIX pass data through several stages before reaching the final destination. Build your test with these intermediate stages in mind, instead of using the standard, functional I/O route.

Error Messages

The SunDiag environment provides special messaging facilities that allow your test to send information and error messages to log files or the console window, or to both. See “Special Messaging Facilities” on page A-6 for more information on how to account for these error messages.

Device Driver Support

Since all SunDiag tests run under UNIX, hardware components and external devices are accessed through their associated device drivers. The purpose of the device driver is to provide a uniform interface to the application program, providing calls such as `open()`, `read()`, `write()`, and `ioctl()`. This design philosophy of device-independent programming in UNIX actually works *against* writing an on-line diagnostic program, since device-specific characteristics are "hidden" by the device driver. The "hooks" that can be implemented into the device driver are provided through the `ioctl()` calls. Base your test design not only on existing driver support, but also on possible ways of directly accessing the device through new `ioctl()` calls.

FRU Fault Isolation

If a device can be partitioned into more than one FRU (Field Replaceable Unit), then organize your test so that each FRU can be tested separately.

Low Impact on System Performance

Sometimes, it is desirable to design a test to exploit system resources to uncover any hidden stress/load failure conditions. But, the test should be able to co-exist with whatever applications are running and not interfere with most of those applications' activities.

Non-Intrusiveness

If your test involves I/O, design the test so that it does not introduce any possibility for the system to lose or damage data. For example, if the test would need to write a data pattern on a type of media and read that pattern back for verification, make the test write to a diagnostic or maintenance area in the media, rather than to a normal, user-accessible area.

Utilization of Self-test Capability

If a device to be tested has a built-in self-test capability, then use that self-test capability to gain more detailed information about a failure. Also, many device drivers provide the `IOCTLDIAG ioctl()` flag to find the error code for the last command issued to the device.

A.3 Test Implementation Guidelines

The purpose of defining SunDiag test implementation guidelines is to minimize, as much as possible, the restrictions and requirements placed on you when you develop new tests, without compromising the SunDiag system integrity or the test efficiency. In addition to detailing the SunDiag programming environment, SunDiag library routines included in `/opt/SUNWdiag/lib` are explained. Use these libraries so that you do not have to re-invent the basic programming environment each time you develop a new test.

A.3.1 The SunDiag Programming Environment

A.3.1.1 Global Variables

When users begin testing with SunDiag, they specify options from the OPEN LOOK user interface (which is, in fact, the SunDiag kernel.) These options are translated by the kernel into command line arguments (see "Standard Command Line Arguments" on page A-15) which are passed to individual tests, where the corresponding global variables will be set. It is the responsibility of each test to examine the state of these global variables and act accordingly.

Using these global variables insures consistent behavior in your tests.

Table A-1 An Overview of SunDiag Global Variables

Read-Only Global Variables	**Definition**
`int single_pass = FALSE;`	TRUE, if single_pass enabled
`int core_file = FALSE;`	TRUE, if core_file to be created
`int verbose = FALSE;`	TRUE, if verbose mode enabled
`int quick_test = FALSE;`	TRUE, if quick_test enabled
`int run_on_error = FALSE;`	TRUE, if run_on_error enabled
`int exec_by_sundiag = FALSE;`	TRUE, if executed by Sundiag
`int debug = FALSE;`	TRUE, if debug mode is enabled
`int trace = FALSE;`	TRUE, if trace mode is enabled
`int instances = n;`	where *n* is the number of total instances for the test
`int which_instance = n;`	where *n* is the instance this test is assigned

Table A-1 An Overview of SunDiag Global Variables *(Continued)*

Writeable Global Variables	**Description**
You must set these four values in the tests you develop:	
`char *versionid=”unknown_version”;`	SCCS version id; printed in trace mode
`char *device_name=”unknown_device”;`	name of UNIX device being tested
`char *func_name = “unknown_function”;`	function (routine) name; primarily used in trace mode
`int subtest_id = 999;`	subtest identification number, where you supply a value for a functional subtest within your test. See “Standard Formats Display” for this variable’s usage.
Global Reserved Variables	
The following are SunDiag reserved global variables and therefore should not be used:	
`char *test_name;`	
`char *hostname;`	
`int list_usage = FALSE;`	TRUE, if list_usage is enabled

`int single_pass` is set to FALSE by default; if a user specifies single_pass from the command line, then your test needs only to run one pass.

`int core_file` is flagged by the user to capture core files. If this flag is set to TRUE, the tests should not catch the signals. See `signal()` handling in the “Requirements” section of this appendix.

`int verbose` is set to TRUE when the user wants to see each test‘s functionality being traced.

`int quick_test` is set to TRUE for quick testing mode (a faster, abbreviated version of the test).

`int run_on_error` is set to TRUE by the user. If an error occurs, the test continues to run the next test sequence instead of exiting.

`int exec_by_sundiag` is set to TRUE if the test is executed by the SunDiag kernel, and FALSE if the test is executed from the command line.

`int list_usage` shows the user how to run the test. It provides three parts: command line usage, standard arguments, and routine-specific arguments.

`int debug` is set to TRUE if debugging mode is enabled.

`int trace` is set by the user to see the test's functionality traced down by function name in the console window.

`char *func_name` is a label used to indicate the name of the function within the test. It is used mainly in test trace mode to indicate which function the test is currently executing. This label is used as an argument in the debugging macros, TRACE_IN and TRACE_OUT.

`instances = n` is the number of total instances for the test.

`which_instance = n` is the instance this particular test is assigned.

`char *versionid` is a label used to indicate the SCCS version of the particular test. It is used to identify the version level of the test's source revision level. See "Standard Formats Display" on page A-16.

`char *device_name` is the name of the device being tested.

A.3.1.2 Special Messaging Facilities

The SunDiag software logs errors in a tiered manner using two log files. The first log file is the Info log file, where non-catastrophic errors are logged. The second log file is the Error log file, where more serious (catastrophic) errors are logged.

The SunDiag error-logging / message-handling routine `send_message()` (see "Standard Library Routines" on page A-11) recognizes the message types shown in Table A-2. They direct the error logger to send user-provided message to the proper log files.

Note – The #define statements are reserved and unique in the SunDiag exerciser, so you should not use the same define name for other purposes.

Table A-2 SunDiag Reserved #define Statements

Define Name	ID #	Defined...
`#define INFO`	0	to info log only
`#define WARNING`	1	to info log only
`#define FATAL`	2	to info and error logs
`#define ERROR`	3	to info and error logs
`#define LOGFILE`	4	to info log (unformatted)[1]
`#define CONSOLE`	5	to SunDiag console (unformatted)
`#define VERBOSE`	6	same as CONSOLE, but formatted in SunDiag mode, unformatted in on-line command mode
`#define DEBUG`	7	same as VERBOSE
`#define TRACE`	8	to console with trace format

1. The format is described in “Standard Formats Display” later in this appendix.

Note – You should *not* use `printf()` or `fprintf()` in your test code to display messages or information. Instead, use `send_message()` with one of the message types defined in Table A-2.

VERBOSE lets general SunDiag users understand how testing is progressing. When VERBOSE mode is enabled by the user, test and subtest messages are printed to the console window while testing is conducted.

DEBUG gives test programmers or sophisticated users more detail or internal information about the tests they are developing. This type of information may include a data structure snapshot, data buffer contents, more detailed procedures, or even describe a changed variable value that could be important for debugging.

TRACE lets users understand the test code structure and the procedures the test goes through. More importantly, TRACE can tell a developer at what function scope level the system hangs.

VERBOSE and DEBUG messages are not associated; they don't control or supervise each other.

For the sake of standardization, the SunDiag console window always displays "formatted" messages for all the message types except CONSOLE and LOGFILE. However, for testing in the on-line command mode, all messages are displayed "unformatted." TRACE messages are somewhat different, but they should be self-explanatory to the user. Note that there is one special message type, VERBOSE+DEBUG or DEBUG+VERBOSE, that allows the specified message to be displayed while either the VERBOSE or DEBUG flag is enabled.

A.3.1.3 Reserved Exit Codes

Table A-3 shows a reserved exit code used by `send_message()`. You can define your own exit codes; however, the SunDiag reserved exit code should not be changed.

Table A-3 Exit codes reserved by `send_message()`

Define Name	ID #	Definition
`#define SKIP_ERROR`	0	skip error, continue testing

The exit codes `-1` and `90-99` are SunDiag reserved exit codes and therefore should not be used.

A.4 Requirements

Now that you know the basic structure of the SunDiag programming environment, you'll need to follow these five requirements.

1. **Use the standard routine `send_message()` for error or message handling.**

The standard routine `send_message()` has only two categories of exit status: zero (SKIP_ERROR), and nonzero (ERROR). Use the nonzero status, message type, and string for SunDiag logging errors and program exit. If some error is minor and tolerable, and the program needs to continue after error logging, use SKIP_ERROR in `send_message()`.

Do not use the `printf()` type of macro, since `stdout` is redirected to the SunDiag console window.

To write a message to the SunDiagLog use `send_message()` with `ret_status zero` (SKIP_ERROR); the program will continue to the next line of code after `send_message()` returns.

Always use `send_message()`, instead of `printf()`, to report text messages as follows:

`send_message(SKIP_ERROR, VERBOSE, “....%s..%d”, par1, par2);`

or

`send_message(0, DEBUG, “......”);`

or

`send_message(0, DEBUG+VERBOSE, “....%d..%s”, par1, par2);`

Note – The current version of `send_message()` is capable of handling variable arguments, just like `printf()` does. Often, it’s useful to print the error message after a system call error. Just pass `strerror(errnum)` as the third argument to `send_message()`.

---

2. **Always use the proper exit code for terminating tests.**

For the normal non-error termination, the test code should use `exit(0)` at the end. Do not use error codes `-1` and `90-99`, since they are reserved SunDiag exit codes.

3. **Define `device_name` in your test code.**

`send_message()` uses the global variable `device_name` to pass the current device name to the server for error logging. This variable has to be set with the proper string values at the earliest phase. For example:

```
main(argc, argv)
 int argc;
 char *argv[];
    {
        device_name = “/dev/rsd0”
        ...
    }
```

Code Example A-1 Defining `device_name`

In this example, the `device_name` is set to a fixed device string name at the beginning of the main routine.

However, for some complicated tests that may go through several devices in one test, you might want to assign a string pointer variable to `device_name`. But beware; avoid using the statement:

```
strcpy (device_name, “/dev/llllllonggggggnnnname”)
```

or

```
strcpy (device_name, device_string);
```

since `device_name`, from your viewpoint, is just defined as:

```
extern char *device_name;
```

indicating no *known* adequate space allocated to this pointer.

4. Always provide `clean_up()` in your test code.

`clean_up()` is a routine that is called by the SunDiag library routine `finish()`, which will be called by the `signal()` handler described below. This routine allows you to return the programming environment to the state it was before running the test. You’ll need to write your own `clean_up()`.

For example, if you disabled the mouse pointer during your test, this is the routine where you would re-enable the mouse pointer.

5. `signal()` Handling

One of the functions of the SunDiag library routine `test_init()` is to catch `signal()` for the SIGHUP, SIGTERM, and SIGINT signals. When any of the above signals are detected, a call is made to the SunDiag library routine `finish()`, which, in turn, calls your test routine `clean_up()`, and then the test will exit.

If you want to use your own signal-catching mechanism, you must remap the signal vector in your test code. This can be done by calling `signal()` after calling `test_init()`.

A.5 Standard Library Routines

The SunDiag program provides some standard routines for communication with each test. You should use these routines in your test to communicate with the SunDiag kernel.

1. **void send_message(exit_code, type, format[, arg]...)**
   This routine is used for SunDiag message handling.
   It has three required arguments:

   `int exit_code`: If this argument is zero (SKIP_ERROR), `send_message()` logs the error message and returns to its caller; if non-zero, it calls your `clean_up()` routine and exits with `exit_code`. The `exit_code` will also become the "message identifier" in the logged message. See Section A.5.2, "Standard Formats Display" for details.

   `int type`: Indicates message type based on its error status or severity (for example, ERROR, WARNING or INFO).

   `char *format`: This argument uses the same format as `printf()`. This defines the number of arguments that will follow. The message provided here will be formatted by the SunDiag program if its `msg_type` is VERBOSE, INFO, WARNING, FATAL, or ERROR; otherwise, the message will appear unformatted.

Note – If the `exec_by_sundiag` flag is not set, `send_message()` will redirect all output to `stderr`, thus allowing the test to be executed from the UNIX shell command line.

2. **void test_init(argc, argv, process_test_args, routine_usage, test_usage_msg)**
   This routine simplifies the procedures required by SunDiag for each test code. It combines almost all the necessary test initialization procedures required by SunDiag into one routine.

   This routine requires five arguments:

   `int argc`: UNIX standard input arguments

   `char *argv[]`: UNIX standard input arguments

   `int (*process_test_args)(argv, argindex)`:
   This argument is a pointer to a function that processes test-specific command line arguments. This function is invoked once for each

command line argument process by `test_init()`, and it requires two arguments: `char *argv[]`, and `int argindex`, where `argindex` is an index for the `argv[]` array.
For SunDiag naming conventions, the test portion of the routine name is usually replaced by "`test`". For example, in the disk test, the name is `process_scsidisk_args()`.

`int (*routine_usage)()`:
This argument is a pointer to a function that displays explanations of the test-specific command arguments.

`char *test_usage_msg`:
This is a pointer to a string that contains the general usage of the test command. For example, if the test uses a device "foo," then the `test_usage_message` might say something like:
`testname D=foo`

`test_init` takes care of the following test initialization procedures:

- Gets `test_name` from `argv[0]`.
- Sets up the signals SIGHUP, SIGTERM, and SIGINIT, and links them to the SunDiag library `finish()` routine.
- Checks and processes the SunDiag standard command line arguments.
- Checks and processes test-specific arguments by calling the user-provided `process_args` routine, which gives you some flexibility to check and manage your own test-specific arguments.
- Checks the existence of any `test_usage_msg` for test-specific arguments. This argument allows the user to type the test name to run in an on-line command mode if no `test_usage_msg` is required.
- Displays a user-provided `test_usage_msg` if an argument check error occurs.
- Displays test usage by invoking the user-provided `routine_usage()` routine when the '**u**' (usage) argument is detected at command line parsing.
- If the global variable `core_file` is set to TRUE, sets up the signals SIGILL, SIGBUS, and SIGSEGV to link with the SunDiag library `exception()` routine to avoid core dumps.
- If the verbose flag is set to TRUE, displays the verbose message "Started" to show a test has started executing.

3. **void test_end()**
   This routine works in tandem with `test_init()` to do the following:

   - Display the verbose message “Stopped” as indication of end of testing
   - Exit with `exit(0)`.

   You should use `test_init()` and `test_end()` to start and end your SunDiag test code, since they simplify your test code structure. Using `test_init()` and `test_end()` also ensures compatibility with future, enhanced versions of SunDiag software.

4. **void trace_before(string)**
   This routine attaches the user-provided string to the end of the trace format message. It traces down the control flow of the testing program by displaying the function names being called and their scope level relative to the program’s main routine (`L0`).

   This routine takes one argument:

   `char *string`: The string argument is any user-supplied string value.

   This routine derives one macro, called TRACE_IN, that is often used in SunDiag test code. TRACE_IN is defined in `sdrtns.h` as

   ```
   (void)trace_before((char *)NULL)
   ```

   which is also equivalent to

   ```
   (void)trace_before(func_name).
   ```

You may want to put a `trace_before()` call before you enter some routine that may be suspected:

```
trace_before(“entering suspect routine”)
suspect_routine()
trace_after(“exited out of suspect_routine”)
```

Code Example A-2 Using `trace_before` and `trace_after` to bracket suspect function calls

5. void trace_after(string)

This routine works in conjunction with `trace_before()`. Use `trace_before()` prior to entering one area, then use `trace_after()` after leaving that area. As in `trace_before()`, a macro called TRACE_OUT is defined in `sdrtns.h` as

```
(void)trace_out((char *)NULL)
```

To properly write a SunDiag test, you should use the TRACE_IN and TRACE_OUT macros for each function to complete the trace mode of your test:

```
function()
   {
       func_name = “function”;
       TRACE_IN
       ...
       ...
       TRACE_OUT
       return(0)
   }
```

Code Example A-3 Using the TRACE_IN and TRACE_OUT macros to bracket each function in your test. Remember that “func_name” is a library defined global variable.

A.5.1 Standard Command Line Arguments

Every SunDiag test should be able to handle the command arguments shown in Table A-4.

Table A-4 SunDiag Standard Command Line Arguments

Command Line Argument	Global Variables[1]
s, sundiag mode	`exec_by_sundiag`
c, enable core dump	`file_core_file`
p, single pass only	`single_pass`
r, run on error	`run_on_error`
q, quick test	`quick_test`
u, list usage	`list_usage`
v, verbose mode	`verbose`
d, debug mode	`debug`
t, trace mode	`trace`
i = *n*, where *n* is the number of total instances for the test	`instances` = *n*
w = *n*, where *n* is the instance this test is assigned	`which_instance` = *n*

1. The variable will be set to TRUE if the option was specified in the command line and after test_init() was called.

For more details about each command line argument function, see Chapter 5, "Running Individual SunDiag Tests from the Command Line." For the global variable associated with these command line arguments, see the section "Global Variables" on page A-4 of this appendix.

A.5.2 Standard Formats Display

Messages that appear in info log files or error log files are formatted in the following way:

```
999.VV.SSS.EEE date time device test [ERROR/FATAL/INFO/WARNING]: message
```

where:

`999` = test identifier (three digits) signifies “user-defined” tests

`VV` = test version identifier (two digits) uses “versionid” variable

`SSS` = subtest identifier (three digits) uses “subtest_id” variable

`EEEE` = error identifier. The first digit is the priority identifier (message type) and the other three digits are the message identifier. See Table 2-1 for the definition of the priority identifier.

For more information, see “Log Files Window Button” on page 2-11.

A.5.2.1 Trace Mode Formats

The formats for trace mode display are the following:

For on-line command mode:

```
@test device: L# func_name(Enter) string (for TRACE_IN)
```

```
@test device: L# func_name(Leave) string (for TRACE_OUT)
```

where L# indicates the scope level of the function `func_name` in the test code relative to the main scope level (`L0`).

For SunDiag window mode:

```
date time device test: L# func_name(Enter) string (for
TRACE_IN)
```

```
date time device test: L# func_name(Leave) string (for
TRACE_OUT)
```

A.6 A Sample Test File

You can use the following sample test file as a template for developing your own SunDiag test.

Code Example A-4 Sample SunDiag Test *(1 of 5)*

```
”@(#)newtest.c 1.11 90/01/05 Copyright Sun Micro”.
#ifndef lint
static char     sccsid[] = “%Z%%M% %I% %E% Copyright Sun Micro”;
#endif
#include <stdio.h>
#include “newtest_msg.h”
#include “sdrtns.h”/* sdrtns.h should always be included */


#define DEVICE_NAME “newdevice”
#define TOTAL_PASS 7/* number of test loops */
#define ERROR_LIMIT 5/* max number of errors allowed if run_on_error */
/* error return code definitions(can be put in a.h file) */
#define NEWTEST_ERROR 1
#define TOO_MANY_ERRORS 2

static char *test_usage_msg = “[test_specific_cmd1] [test_specific_cmd2]”;
static int flag1, flag2;
/*
 * This is a test template for SunDiag, and is intended to show how you
 * should write a test to be run by SunDiag.
 *
 * First, get the command-line arguments, then validate and/or assign the device
 * name, probe for the device, and then run the test(s). The verify mode is
 * not currently used by SunDiag, but may be useful for running the test by
 * itself under Unix. Note that the average test should run from three to
 * five minutes, and the TOTAL_PASS symbolic constant can be adjusted to
 * generate proper testing period.
 *
 * Note: When run by SunDiag, stdout and stderr will be redirected to SunDiag’s
 * console window. stdin is disabled.
 */

main(argc, argv)
int argc;
char *argv[];
{
    extern int process_test_args();
```

Code Example A-4 Sample SunDiag Test *(2 of 5)*

```
    extern int routine_usage();
    versionid = “%I%”;

    /* device_name can be fixed or passed in from command line */
    device_name = DEVICE_NAME;/* fixed in this case */

    /* Always start with test_init to enter SunDiag environment */
    test_init(argc, argv, process_test_args, routine_usage, test_usage_msg);

    if (!exec_by_sundiag)
        {
            valid_dev_name();
            probe_newdev();
    }
    run_tests(single_pass ? 1 : TOTAL_PASS);
    /* pass the desired loop count and do real testing from here */

    clean_up();
    /* Always end with test_end to close up SunDiag environment */
    test_end();
}

/*
 * Process_test_args() processes test-specific command line arguments.
 */
process_test_args(argv, argindex)
char *argv[];
int argindex;
{
    if (strcmp(argv[argindex], “test_specific_command_1”) == 0)
            flag1 = TRUE;
    else if (strcmp(argv[argindex], “test_specific_command_2”) == 0)
            flag2 = TRUE;
    else
            return FALSE;

    return TRUE;
}

/*
 * routine_usage() explain the meaning of each test-specific command
 * argument.
 */
```

Code Example A-4 Sample SunDiag Test *(3 of 5)*

```
routine_usage()
{
    send_message(0, CONSOLE, “%s specific arguments [defaults]:\n\
            test_specific_cmd1 = meaning of command 1\n\
            test_specific_cmd2 = meaning of command 2\n”, test_name);
}

/*
 * You may also want to consider validating the device name, if your test is
 * also to be run standalone (i.e., without SunDiag) under Unix.
 */
valid_dev_name()
{
    func_name = “valid_dev_name”;
    TRACE_IN
    TRACE_OUT
    return (0);
}

/*
 * The probing function should check that the specified device is available
 * to be tested(optional if run by Sundiag). Usually, this involves opening
 * the device file, and using an appropriate ioctl to check the status of the
 * device, and then closing the device file. There are several flavors of
 * ioctls: see dkio(4s), fbio(4s), if(4n), mtio(4), and so on. It is nice to
 * put the probe code into a separate code module, because it usually has
 * most of the code which needs to be changed for a new SunOS release or
 * port.
 */
probe_newdev()
{
    func_name = “probe_newdev”;
    TRACE_IN
        TRACE_OUT
        return (0);
}

/*
 * Run the test while pass < total_pass.
 *
 * Re: send_message, there are 3 variables for send_message(exit_code, type,
 * msg). “exit_code” indicates the type of error(such as TOO_MANY_ERRORS). A
 * non-zero exit_code will force send_message to terminate the test and use
```

Code Example A-4 Sample SunDiag Test *(4 of 5)*

```
 * it as exit code. Ret_code 97-99 are reserved by SunDiag for
 * RPCFAILED_RETURN(97), INTERRUPT_RETURN(98), and EXCEPTION_RETURN(99). For
 * “type”, use INFO, WARNING, FATAL, ERROR, LOGFILE, or CONSOLE. For “msg”,
 * use a pointer (char *) to a message buffer.
 *
 * send_message() always writes the “msg” to stderr, and if run by SunDiag, it
 * will also log “msg” to INFO log and ERROR log (for msg_type FATAL and
 * ERROR).
 *
 * The “msg” is formatted as follows before written: (void)sprintf(msg_buf,
 * “%02d/%02d/%02d %02d:%02d:%02d %s %s %s: %s\n”, (tp->tm_mon + 1),
 * tp->tm_mday, tp->tm_year, tp->tm_hour, tp->tm_min, tp->tm_sec,
 * device_name, test_name, msg_type, msg);
 *
 * For more details, read sdrtns.c and sdrtns.h.
 */
run_tests(total_pass)
int total_pass;
{
    int pass = 0;
    int errors = 0;

    func_name = “run_tests”;
    TRACE_IN
        while (++pass <= total_pass)
    {
            send_message(0, VERBOSE, “Pass= %d, Error= %d”, pass, errors);

            if (!newdev_test())
                if (!run_on_error)
                        send_message(NEWTEST_ERROR, ERROR, failed_msg);
                else if (++errors >= ERROR_LIMIT)
                        send_message(TOO_MANY_ERRORS, FATAL, err_limit_msg);
    }
    TRACE_OUT
}

/*
 * The actual test. Return True if the test passed, otherwise FALSE.
 *
 * Note: the “quick_test” flag is used to force an error here. Its normal use is
 * to force a “quick(short)” version of the test.
 */
```

Code Example A-4 Sample SunDiag Test *(5 of 5)*

```
int
newdev_test()
{
    func_name = “newdev_test”;
    TRACE_IN
        TRACE_OUT
        return (quick_test ? FALSE : TRUE);
}

/*
 * clean_up(), contains necessary code to clean up resources before exiting.
 * Note: this function is always required in order to link with libtest.a
 * successfully.
 */
clean_up()
{
    func_name = “clean_up”;
    TRACE_IN
        TRACE_OUT
        return (0);
}
```

A.7 A Sample Makefile

The following Makefile is a general example of what happens when you “make” your SunDiag test code.

Code Example A-5 Sample SunDiag Makefile *(1 of 2)*

```
#
# @(#)Makefile 1.12 90/01/05 Copyright(c) Sun Microsystems, Inc.
#

.DEFAULT:
	sccs get -G$@ $@
DBX=-O
# specify DBX=-g for dbx version

DESTDIR =
PROGRAM = newtest
INCLUDES= newtest_msg.h sdrtns.h
LIBS =
SOURCES = newtest.c newtest_msg.c
OBJECTS = $(SOURCES:.c=.o)
LINTFILES = $(SOURCES:.c=.ln)

CFLAGS = $(DBX) -D$(REV) -D`arch`
LDFLAGS = $(DBX)
LINTFLAGS= -D$(REV) -D`arch`

.KEEP_STATE:

##### beginning of dependency lines #####

all: $(INCLUDES) $(PROGRAM)

$(PROGRAM): $(OBJECTS)
	cc $(LDFLAGS) -o $@ $(OBJECTS) $(LIBS)

install: all FRC
	@if [ $(DESTDIR) ]; then \
		set -x; \
		install -s $(PROGRAM) $(DESTDIR); \
	else \
		set -x; \
		install $(PROGRAM) ../../bin; \
	fi
```

Code Example A-5 Sample SunDiag Makefile *(2 of 2)*

```
clean: FRC
	rm -f $(PROGRAM) $(OBJECTS) $(LINTFILES) core

lint: $(LINTFILES)
	lint $(LINTFLAGS) $(LINTFILES) $(LIBS)

info: FRC
	sccs info
```

Loopback Connectors B

Loopback connectors are designed for the testing of communication ports. They take the form of either a single plug or a port-to-port cable with some communication connections shorted (looped-back).

Note – Loopback connectors must be wired properly and connected firmly for the Serial Port Tests to work correctly. Miswired, poorly soldered, or missing loopback connectors can cause erroneous diagnostic error messages.

Table B-1 depicts the pin assignments for most loopback plugs and cables that may be used when testing a system.

Table B-1 Pin Connections for Loopback Plugs

Signal Description	EIA	CCITT #	RS-449 "A"	RS-449 "B"	DIN 8 8-pin round	DB9 9-pin	DB25 25-pin	Direction	Alpha ID
Chassis/frame ground	AA	101	1	NC	NC	NC	1	none	AA
Transmit Data (TxDa)	BA	103	4	22	3	3	2	output	BA
Receive Data (RxDa)	BB	104	6	24	5	2	3	input	BB
Request To Send (RTSa)	CA	105	7	25	6	7	4	output	CA
Clear To Send (CTSa)	CB	106	9	27	2	8	5	input	CB
Data Set Ready (DSRa)	CC	107	11	29	NC	6	6	input/ output	CC
Signal Ground (SG)	AB	102	19	NC	4	5	7	none	AB
Data Carrier Detect (DCDa)	CF	109	13	31	7	1	8	input	CF
Transmit Clock In (TRxCa)	DB	114	5	23	NC	NC	15	input	DB
Receive Clock in (RTxCa)	DD	115	8	26	8	NC	17	input	DD
Data Terminal Ready (DTRa)	CD	108	12	30	1	4	20	output	CD
External Clock Out (TRxCa)	DA	113	17	35	NC	NC	24	output	DA
Secondary Data Carrier Detect (DCDb)	SCF	122	NC	NC	NC	NC	12	input	SCF
Secondary Clear to Send (CTSb)	SCB	121	NC	NC	NC	NC	13	input	SCB

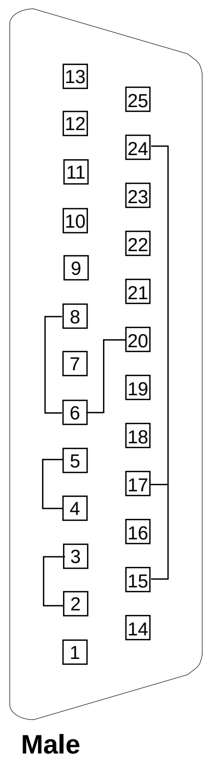
Table B-1 Pin Connections for Loopback Plugs *(Continued)*

Signal Description	**EIA**	**CCITT #**	**RS-449 “A”**	**RS-449 “B”**	**DIN 8 8-pin round**	**DB9 9-pin**	**DB25 25-pin**	**Direction**	**Alpha ID**
Secondary Transmit Data (TxDb)	SBA	118	NC	NC	NC	NC	14	output	SBA
Secondary Receive Data (RxDb)	SBB	119	NC	NC	NC	NC	16	input	SBB
Secondary Request to Send (RTSb)	SCA	120	NC	NC	NC	NC	19	output	SCA

Notes: NC = No connection

B.1 25-Pin RS-232 Loopback Plug

The RS-232 and RS-423 single-port loopback plug is a specially wired male DB-25 connector. It is plugged in to a serial port in the back of the system under test. The wiring is shown in Figure B-1.

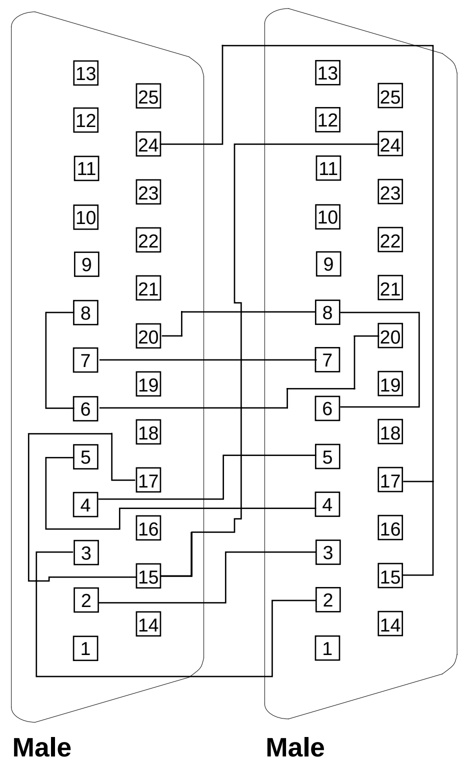


Connect:	pin 3	to	pin 2
	pin 5	to	pin 4
	pins 6 and 8	to	pin 20
	pins 15 and 17	to	pin 24

Figure B-1 25-pin RS-232 Loopback Plug

B.2 25-pin RS-232 Port-to-Port Loopback Cable

Use these wiring directions for 25-pin RS-232 and RS-423 port to 25-pin RS 232 and RS 423 port loopback cables (two DB-25 connections). It is plugged into a pair of serial ports in the back of the system under test. Both connectors are male. The wiring is shown in Figure B-2.



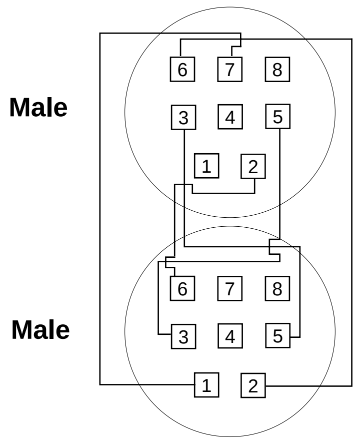
Connect:

First Connector	to	Second Connector
pin 2		pin 3
pin 3		pin 2
pin 4		pin 5
pin 5		pin 4
pins 6 and 8		pin 20
pin 7		pin 7
pins 15 and 17		pin 24
pin 20		pins 6 and 8
pin 24		pins 15 and 17

Figure B-2 25-pin RS-232 Port-to-Port Loopback Cable

B.3 8-Pin to 8-Pin Loopback Cable

Use these wiring directions for 8-pin round DIN RS-232 port to RS-423 to 8-pin round-DIN RS-232 and RS-423 port loopback cable. Both connectors are male.



Connect:

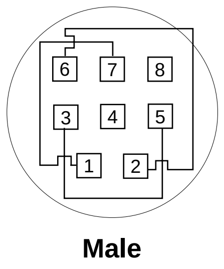
First Connector	to	Second Connector
pin 3		pin 5
pin 5		pin 3
pin 6		pin 2
pin 2		pin 6
pin 7		pin 1

Figure B-3 8-Pin to 8-Pin Loopback Cable

Pin 8, Receive clock In (DD), remains unconnected.

B.4 8-Pin Loopback Plug

Use these wiring directions for male 8-pin round-DIN RS-232 and RS-423 Single port loopback plugs.



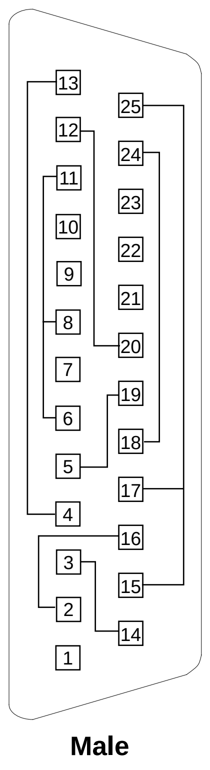
Connect:	pin 3	to	pin 5
	pin 6	to	pin 2
	pin 1	to	pin 7

Figure B-4 8-Pin Loopback Plug

Pin 8, Receive Clock In (DD), remains unconnected.

B.5 25-pin Port A-to-Port B Loopback Plug

Use these wiring directions for a 25-pin Port A to Port B loopback plug for most systems.

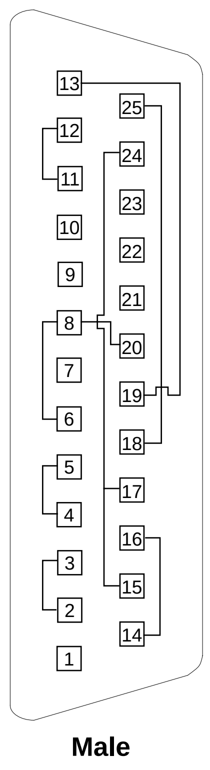


Connect:	pin 16	to	pin 2
	pin 3	to	pin 14
	pin 13	to	pin 4
	pin 5	to	pin 19
	pins 6 and 8	to	pin 11
	pin 12	to	pin 20
	pin 18	to	pin 24
	pins 15 and 17	to	pin 25

Figure B-5 Port A-to-Port B Loopback Plug

B.6 25-pin Port A-to-A Port B-to-B Loopback Plug

If your system has a single communication port to connect it to peripherals, use these wiring instructions for making a male 25-pin loopback plug for that communication port:



Connect:	pin 3	to	pin 2
	pin 5	to	pin 4
	pins 6 and 8	to	pin 20
	pin 12	to	pin 11
	pin 13	to	pin 19
	pin 16	to	pin 14
	pins 15 and 17	to	pin 24
	pin 25	to	pin 18

Figure B-6 Port A-to-A, Port B-to-B Loopback Plug

The `what_rev` *Utility* C

This utility provides version control facilities and is provided in the `/opt/SUNWdiag/bin` directory with the release. Its purpose is to allow you to determine which files in the SunDiag exerciser are different from those officially released.

When you type the command `what_rev`, you receive a report of the differences between the officially released files and those files in your SunDiag directory.

The `-v` option gives you the SunDiag version; the `-h` option provides you with a usage message.

Index