SunVTS 6.0 Patch Set 1 Documentation Supplement

Sun Microsystems, Inc.
www.sun.com

Part No. 819-1804-11
March 2005, Revision B

Submit comments about this document at: http://www.sun.com/hwdocs/feedback

Contents

CHAPTER 1

Introduction

The Sun™ Validation and Test Suite (SunVTS) software runs multiple diagnostic hardware tests from a single user interface. SunVTS verifies the connectivity, functionality, and reliability of most hardware controllers and devices.

This manual is a supplement to the SunVTS™ 6.0 documentation and describes new features, tests, and test enhancements that are developed in the SunVTS 6.0 patch releases. The new features, tests, and test enhancements included in this document are provided in the SunVTS 6.0 Patch Set 1 software.

For overall SunVTS features, test configuration modes, interfaces, and options refer to the *SunVTS 6.0 User's Guide*. Refer to the *SunVTS 6.0 Test Reference Manual* for detailed information on SunVTS test software and the full collection of tests released with SunVTS 6.0.

Refer to the latest version of the *SunVTS 6.0 Patch Set 1 Release Notes* for important release information and open issues; this document is available at:
`http://www.sun.com/products-n-solutions/hardware/docs/Software/system_administration/tools/testing_valid/sunvts/index.html`

New Features

The following tests are introduced in this 6.0 patch release:

- Cryptographics Test (`cryptotest`) – Described in Chapter 2
- Graphics Buffer Test (`graphicstest`) – Descibed in Chapter 5
- Infiniband Test Host Channel Adapter (`ibhcatest`) – Descibed in Chapter 6

The following test was enhanced in this release:

- Disk and Floppy Drives Test (`disktest`) – Described in Chapter 3
- Fast Frame Buffer Test (`ffbtest`) – Described in Chapter 4

- Sun™ XVR-1200 Graphics Accelerator Test (`jfbtest`) – Described in Chapter 7
- Sun™ XVR-100 Graphics Accelerator Test (`pfbtest`) – Described in Chapter 8
- Sun™ XVR-4000 Graphics Accelerator Test (`zulutest`) – Described in Chapter 10
- RAM test (`ramtest`) – Described in Chapter 9

Getting SunVTS Version Information

The standard command line argument, `-V`, displays the SunVTS version and release date of the test if available.

SunPCi-3 Card Support

`sunpci2test` now supports the SunPCi-3 cards. Solaris 10 supports SunPCi-3 Version 3.2.2 with Patch 118591-01 only. Solaris 10 does not support the SunPCi-2 card.

x86 Solaris Support

Starting with Solaris 10, the SunVTS infrastructure and a few core diagnostics are available for x86 Solaris platforms. The current x86 support is for the 32-bit operating system only with the exception of three tests. See "32- and 64-bit Support With x86 Platforms" on page 3.

SunVTS is supported and tested on the following Sun x86 platforms:

- Sun Fire V60x
- Sun LX50
- Sun Fire B100x
- Sun Fire B200x
- Sun Fire V20z
- Sun Fire V40z

You must install the x86 version of the SunVTS packages to perform SunVTS on x86 platforms. The software packages use the same names as in the SPARC environment. The SunVTS packages delivered separately for both x86 and SPARC Solaris platforms are as follows:

- `SUNWvts` – Contains the SunVTS core framework that includes the kernel and user interface.
- `SUNWvtsmn` – Contains the SunVTS online manual pages

- SUNWvtsr – Contains the SunVTS framework configuration files in the root partition (Superuser).
- SUNWvtsts – Contains the SunVTS test binaries.

The SunVTS components available for x86 Solaris platforms are as follows.

Infrastructure:

- sunvts
- vtsk
- vts_cmd
- vtstty
- vtsui
- vtsprobe

SunVTS Tests:

- CD DVD Test (cddvdtest)
- CPU Test (cputest)
- Disk and Floppy Drives Test (disktest)
- Data Translation Look-aside Buffer (dtlbtest)
- Floating Point Unit Test (fputest)
- Network Hardware Test (nettest)
- Ethernet Loopback Test (netlbtest)
- Physical Memory Test (pmemtest)
- RAM test (ramtest)
- Serial Port Test (serialtest)
- System Test (systest)
- Universal Serial Board Test (usbtest)
- Virtual Memory Test (vmemtest)

32- and 64-bit Support With x86 Platforms

Three SunVTS tests support 64-bit x86 Solaris for the AMD64 based Sun platforms. By default, these files are installed in the /bin/64 directory under the base SunVTS install location.

The following tests support 64-bit architecture on x86 Solaris.

- pmemtest - Physical Memory Test
- ramtest - Memory DIMMs (RAM) Test
- vmemtest - Virtual Memory Test

Currently, SunVTS does not provide 64-bit infrastructure support for x86 Solaris. For 64-bit x86 support, these tests can be performed as standalone with the Command Line Interface (CLI) only. To view the command-line options use the -u (usage) option as follows:

- ./pmemtest -u

- `./ramtest -u`
- `./vmemtest -u`

Refer to the `README.64` file for more information.

SunVTS Overview

SunVTS is composed of many individual tests that support testing of a wide range of products and peripherals. Most of the tests are capable of testing devices in a 32-bit or 64-bit Solaris environment.

Use SunVTS to test one device or multiple devices. Some of the major test categories are:

- Audio tests
- Communication (serial and parallel) tests
- Graphic/video tests
- Memory tests
- Network tests
- Peripherals (disks, tape, CD-ROM, DVD-ROM, printer, floppy) tests
- Processor tests
- Storage tests

Such flexibility means that the proper test modes and options need to be selected to maximize its effectiveness.

Note – When an error occurs in VTS testing, the test message window displays the error number, the error description, the probable cause of the error, and the recommended actions. Because this information is displayed at the time of the error, error messages are not included in this manual.

The default installation directory for SunVTS is `/opt/SUNWvts`. However, when you are installing SunVTS, you can specify a different directory. Refer to the *SunVTS 6.0 User's Guide* for installation information.

Test Requirements

SunVTS 6.0 and 6.0 patch releases are supported in the Solaris 10 and Operating System and future compatible Solaris releases.

The operating system kernel must be configured to support all peripherals that are to be tested.

Some SunVTS tests have special requirements such as the connection of loopback connectors, installation of test media, or the availability of disk space. These requirements are listed for each test in the corresponding chapter in this book.

Collection of SunVTS Tests

Many individual tests make up the collection of tests in the SunVTS application. Each test is a separate process from the SunVTS kernel. Each test can be run individually from the command line or from the SunVTS user interface.

When SunVTS is started, the SunVTS kernel automatically probes the system kernel to determine the hardware devices. The devices are then displayed on the SunVTS control panel with the appropriate tests and test options. This provides a quick check of your hardware configuration, and no time is wasted trying to run tests that are not applicable to your configuration.

During testing, the hardware tests send the test status and messages to the SunVTS kernel through interprocess communication (IPC) protocols. The kernel passes the status to the user interface and logs the messages.

SunVTS has a shared object library that contains test-specific probing routines. At runtime, the SunVTS kernel dynamically links in and calls these probing routines to initialize its data structure with test-specific information. You can add new tests into the SunVTS environment without recompiling the SunVTS source code.

As of SunVTS 3.0, the SunVTS kernel and most tests support 32-bit and 64-bit operating environments. When the `sunvts` command is used to start SunVTS, the appropriate tests (32-bit or 64-bit versions) are presented.

32-Bit and 64-Bit Tests

Because each test is a separate program, you can run individual tests directly from the command line. When this is done, care must be taken to run the appropriate test (32-bit or 64-bit) that corresponds to the operating system that is running (32-bit or 64-bit). This is done by running tests from specific directories as follows:

- 32-bit tests—`/opt/SUNWvts/bin/`*testname*
- 64-bit tests—`/opt/SUNWvts/bin/sparcv9/`*testname*
  - The test is an actual 64-bit binary test if *testname* is a binary file.
  - The test is a 32-bit test capable of running in the 64-bit environment if *testname* is a symbolic link.

If you use the `sunvts` command to run SunVTS, SunVTS automatically allocates 32-bit or 64-bit tests based on the 32-bit or 64-bit Solaris operating environment that is running. Therefore, the only time that you need to be concerned with the 32-bit or 64-bit operation is when you run the SunVTS kernel or SunVTS tests from the command line.

If you are not sure which operating system is running, refer to the Solaris System Administration manuals. In Solaris 8 2/02 and Solaris 9, the following command can be used to identify the application support of your system.

```
# isainfo -v
```

Note – The `isainfo` command is not available in Solaris 2.6 or earlier releases.

SunVTS User Interfaces

You can run SunVTS tests from various interfaces: The CDE graphical user interfaces, or the TTY interface. SunVTS tests can also be run individually from a shell tool command line, using the command-line syntax for each test (refer to "Running a Test From the Command Line" on page 9). TABLE 1-1 describes the various SunVTS user interfaces. Refer to the *SunVTS User's Guide* for more information on these interfaces.

TABLE 1-1 SunVTS System Interfaces

SunVTS System Interfaces	Description
Graphical user interfaces (GUIs)	Users can select tests and test options by pointing and clicking with a mouse button in the CDE interface.
TTY interface	Users can run SunVTS from a terminal or modem attached to a serial port. This feature requires that users use the keyboard instead of the mouse, and it displays one screen of information at a time.
Command-line execution	Users can run each of the SunVTS tests individually from a shell tool command line using the command-line syntax. Each test description in this book contains the corresponding command-line syntax.

Note – To increase or decrease a numeric value in a SunVTS CDE dialog box, you can use either the up or down arrows, or type a new value in the text box and press Return. Select Apply to apply all dialog box changes.

Running a Test From a User Interface

The common way to run SunVTS testing is through a SunVTS user interface—CDE or the TTY interface.

Test configuration, control, and results are easily accessed through buttons and dialog boxes. These buttons and dialog boxes are covered in the *SunVTS 6.0 User's Guide*. However, the Test Parameter Options dialog box is unique for each test, and is therefore covered in this manual.

Test Parameter Options Dialog Box

The options displayed in this menu differ for each test, but the lower set of buttons are generic and are described below.

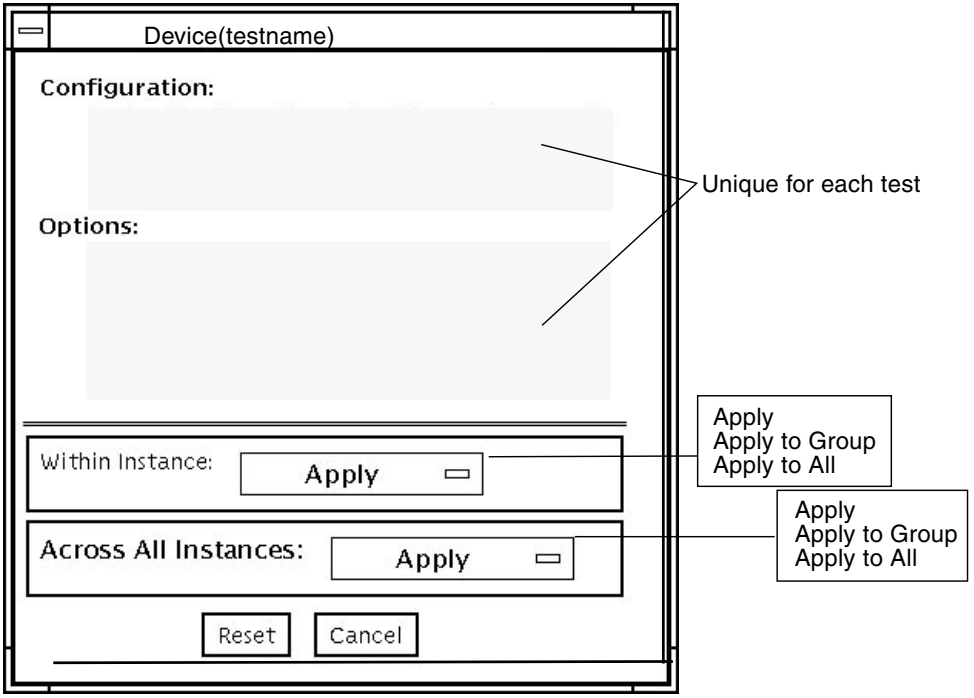

FIGURE 1-1 Test Parameter Options Dialog Box (CDE)

TABLE 1-2 Test Parameter Options Dialog Box Items

Menu Item	Description
Configuration	Information such as device type, capacity, revision, and serial numbers for the selected device. This information cannot be changed.
Options	A list of test options that are used to customize the testing of the selectable device, group, or all devices. The options are specific for each test and are covered in the test specific-chapters in this manual.
Within Instance	Provides the means to apply the settings: • to this device only with Apply, or • to all devices within this group with Apply to Group, or • to all devices (of the *same device type* for *all controllers*) with Apply to All. The option settings are only applied to one instance of the test.

TABLE 1-2 Test Parameter Options Dialog Box Items *(Continued)*

Menu Item	Description
Across All Instances	Provides the means to apply the settings globally: • to this device only with Apply, or • to all devices within this group with Apply to Group, or • to all devices (of the *same device type* for *all controllers*) with Apply to All. The option settings are applied to all instances.
Reset	Returns the option values to their default settings and closes the test parameter option menu.
Cancel	Ignores any changes made to option values and closes the test parameter option menu.

Note – The Test Parameter Options Dialog box descriptions also apply to the Test Parameter Options menu in the TTY interface.

Running a Test From the Command Line

In some cases it may be more convenient to run a single SunVTS test from the command line rather than through a SunVTS user interface. The following information describes how to do this.

Unless specified, the test runs without the SunVTS kernel (`vtsk`). All events and errors are sent to `stdout` or `stderr` and are not logged in the log files.

When you run a test in this way, you must specify all test options in the form of command-line arguments.

There are two types of command-line arguments:

- Standard arguments—common to all tests. Refer to TABLE 1-3 for details.
- Test specific arguments—unique to a specific test. Refer to the test-specific chapters in this book for details.

The standard syntax for all SunVTS tests is:

`testname` [**`-scruvdtelnf`**] [**`-i`** *number*] [**`-w`** *number*] [**`-o`** *test_specific_arguments*]

Note – 64-bit tests are located in the `sparcv9` subdirectory: `/opt/SUNWvts/bin/sparcv9/`*testname*. If a test is not present in this directory, then it may only be available as a 32-bit test. For more information refer to the *SunVTS 6.0 Test Reference Manual*.

Standard Command-Line Arguments

The following table defines the standard SunVTS command-line arguments:

TABLE 1-3 Standard Command-Line Arguments

Argument	Description
`-x`	Runs the test in exclusive test mode. This mode assumes that the test has exclusive access to the device and the system. The testing done in exclusive mode is usually more stressful compared to functional mode. Also, running a test in exclusive mode usually assumes exclusive access to all resources and assumes no other SunVTS test is running at the same time.
`-s`	Runs a test as though it were invoked from the SunVTS kernel (`vtsk`). The default is to send the output to `stdout` or `stderr`.
`-c`	Enables a core image of the test process to be created in the current working directory upon receipt of certain signals, otherwise those signals are caught and handled to prevent a core from being generated. The default is to disable the creation of a core image.
`-r`	Enables run on error so that when an error occurs, the test continues with the next test sequence instead of exiting. The default is false.
`-v`	Runs the test in Verbose mode and displays messages with more detailed information about the testing process. The default is false.
`-d`	Runs the test in debug mode and displays messages to help programmers debug their test code. The default is false.
`-t`	Runs the test in test Trace mode and displays messages that track function calls and sequences currently in use by the test code. The default is false.
`-e`	Runs the test in Stress mode by increasing the system load. The default is false.
`-l`	Runs the test in Online Functional mode. This is the same mode that tests run in when executed with the `vtsui.online` command. It is a non-intrusive version that will not significantly affect other applications. See the note below. The default is true.
`-n`	Runs the test in Connection mode. See the note below. The default is false.
`-f`	Runs the test in full Functional test mode. This mode assumes that the test has complete control of the device under test. See the note below. The default is false.

TABLE 1-3 Standard Command-Line Arguments *(Continued)*

Argument	Description
`-i` *number*	Defines the number of instances for scalable tests.
`-w` *number*	Defines to which instance the test is assigned; this option is for scalable tests.
`-o`	Indicates that the options and arguments that follow are test specific.

Note – Separate each test-specific argument by commas, with no space after each comma.

Note – If you choose to specify a test mode with the `l`, `n`, or `f` option, specify only one option at a time because only one test mode can be selected at a time.

Test-Specific Arguments

There are test-specific arguments, as described in TABLE 1-4. Test-specific arguments follow the format specified in the `getsubopt(3c)` man page. For information about test-specific arguments refer to the specific test chapter in this book.

TABLE 1-4 SunVTS Test-Specific Arguments

Argument	Description
`-o`	Separate each test-specific argument by commas, with no space after the comma. For example: `#`**`./sample -v -o dev=/dev/audio,volume=78`** The test option format is specified by the man page `getsubopt`(3C).

Testing Frame Buffers

Before running a frame buffer test, determine whether the test requires frame buffer locking. Not all frame buffer tests have a locking option. Some tests set the lock automatically. Check the test chapter for each individual test to see if this step is needed. If locking is required, you can set the lock in one of two ways:

- If you are using the CDE SunVTS interface, go to the Option menu of the graphic test and select Enable for the frame buffer locking option.

- If you are working from the command line, you can enable frame buffer locking with the `lock=e/d` option. For example, to run the generic frame buffer test (`fbtest`) with a locked frame buffer, enter:

```
# ./fbtest -o dev=pfb0,lock=enable
```

(See the test command line argument descriptions in the individual test chapters.)

Caution – If frame buffer locking is disabled (unlocked) on frame buffers that are running `vtsui`, or if you move the mouse, you will receive false error messages. Even a slight mouse movement can cause a test to fail.

Caution – Disable the Power Management screen saver option and the Save/Resume option before you run any of the SunVTS frame buffer tests. For information on disabling these Power Management features, refer to the Power Management Chapter in the Solaris User Collection. This document is available at: `docs.sun.com`.

Caution – If you are using the GUI interface for SunVTS, do not conduct frame buffer tests through the `dtlogin` window. Log in as `root` and disable the auto-logout option.

Caution – Do not run TTY mode and frame buffer tests concurrently on the console monitor. The frame buffer test may fail.

Testing Multiple Frame Buffers

The following rules apply when you test multiple frame buffers (displays) simultaneously:

- Only the console monitor can run the window environment (such as CDE or GNOME). The console monitor is the monitor connected to the frame buffer appointed by `/dev/fb`. SunVTS enables frame buffer locking on the console monitor by default.
- The frame buffer that is running the window environment must have window locking enabled to avoid false test failures. All other frame buffers must have window locking disabled.

Remote Testing of Frame Buffers

If you start `sunvts` or `vtsk` from a screen other than the console monitor, frame buffer locking is not available. In this case:

- Disable the window locking option on the remote screen by setting it to `d`.
- Enable frame buffer locking for the console monitor, as shown in the example above. The SunVTS user interface cannot display on a monitor if locking is disabled.

Do not run any graphic programs (including `vtsui`) on the remote frame buffer during graphic testing.

CHAPTER 2

Cryptographics Test (`cryptotest`)

The Encryption Framework in Solaris 10 provides a user level API for access to cryptographic accelerators. This API is based on the PKCS#11 standard. Cryptographic accelerators are referred to as PKCS#11 tokens, and each cryptographic algorithm the token accelerates is referred to as a mechanism.

`cryptotest` tests the mechanism supported by PKCS#11 tokens in Solaris 10.

`cryptotest` supports the Sun Cyrpto Accelerator 500, Sun Cyrpto Accelerator 1000, Sun Cyrpto Accelerator 4000, Niagara Crypto Provider, and all future cryptographic accelerators developed for Solaris 10 onward. PKCS documents and information are available at: `hppt://www.rsasecurity.com/rsalabs/PKCS`

TABLE 2-1 Definitions of the Mechanisms Tested by `cryptotest`

Algorithm	Description
DSA	Digital Signature Algorithm
DES	Data Encryption Standard as defined in FIPS PUB 46-3
MD5 RSA	Data Security MD5 message-digest algorithm.
RSA	Public key cryptosystem.
SHA1	The Secure Hash Algorithm.
RNG	Random Number Generator Algorithm.

cryptotest Subtests

TABLE 2-2 cryptotest Subtests

Subtest	Description
DES	Tests DES bulk encryption
3DES	Tests 3DES bulk encryption
RSA	Tests RSA public and private keys
DSA	Tests DSA signature verification
RNG	Tests random number generation

cryptotest Options

To reach the dialog box below, right-click on the test name in the System Map and select Test Parameter Options. Because graphics test can test multiple types of frame buffers, the test name that is displayed will correspond to the particular framebuffer being tested. If you do not see this test in the System Map, you might need to expand the collapsed groups, or your system may not include the device appropriate to this test. Refer to the SunVTS User's Guide.

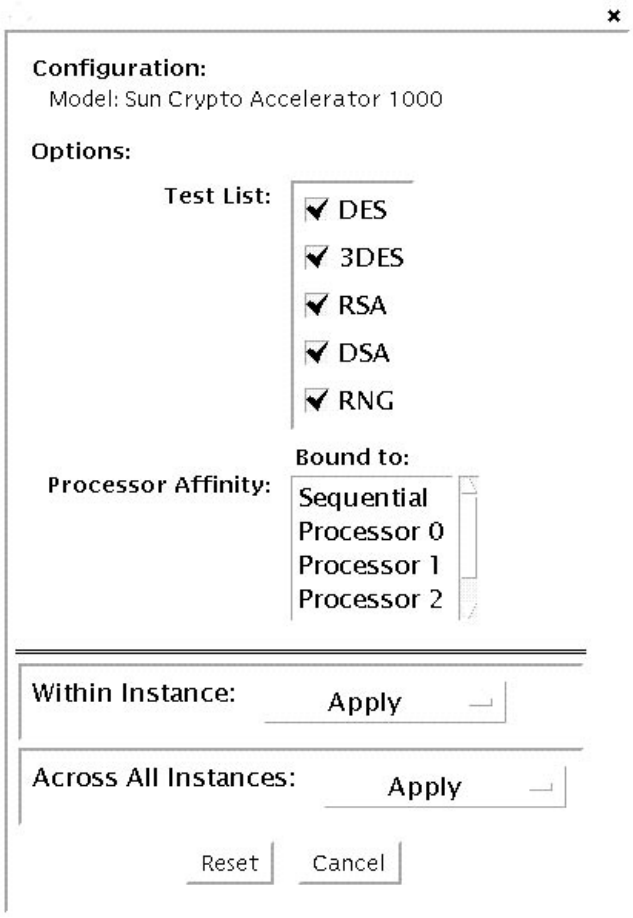

FIGURE 2-1 `dcatest` Test Parameter Options Dialog Box

TABLE 2-3 `dcatest` Options

Option	Description
DES	Tests DES bulk encryption
3DES	Tests 3DES bulk encryption
RSA	Tests RSA public and private keys
DSA	Tests DSA signature verification
RNG	Tests random number generation

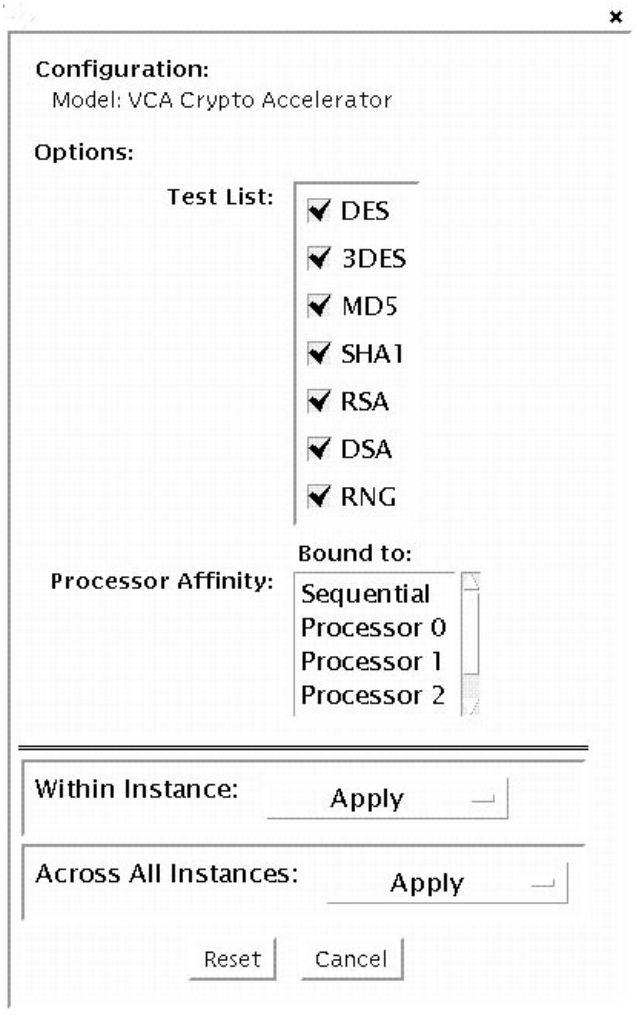

FIGURE 2-2 vcatest Test Parameter Options Dialog Box

TABLE 2-4 `vcatest` Options

Option	Description
DES	Tests DES bulk encryption
3DES	Tests 3DES bulk encryption
MD5	Data Security MD5 message-digest algorithm.

TABLE 2-4 `vcatest` Options *(Continued)*

Option	Description
SHA1	The Secure Hash Algorithm.
RSA	Tests RSA public and private keys
DSA	Tests DSA signature verification
RNG	Tests random number generation

`cryptotest` Test Modes

TABLE 2-5 `cryptotest` Supported Test Modes

Test Mode	Description
Functional	Runs the full set of tests.

`cryptotest` Command Line Syntax for `dcatest`

`/opt/SUNWvts/bin/sparcv9/cryptotest -f -o dev=vca2|dca2, tl=`*RSA+DSA*

TABLE 2-6 `cryptotest` Command Line Syntax for `dcatest`

Option	Description
`dev=dca`*N*	Specifies the instance of the device to test such as `dca0` or `dca2`. Defaults to `dca0` if not included. Note that *N* specifies the placement of the instance number of the device being tested.
`tl=`*testlist*	Specifies the list of subtests to be performed. The subtests for `tl` are separated by the + (plus) character. The supported subtests are DES, 3DES, DSA, RSA, and RNG, so `tl=DES+3DES+DSA+RSA+MD5+SHA1+RNG` enables all subtests. You can also insert `tl=all` which performs all tests. Defaults to `all` if no subtests are specified.

`cryptotest` Command Line Syntax for `vcatest`

`/opt/SUNWvts/bin/sparcv9/cryptotest -f -o dev=vca2, tl=`*RSA+DSA*

TABLE 2-7 `cryptotest` Command Line Syntax for `vcatest`

Option	Description
`dev=vca`*N*	Specifies the instance of the device to test such as `vca0` or `vca2`. Defaults to `vca0` if not included. Note that *N* specifies the placement of the instance number of the device being tested.
`tl=`*testlist*	Specifies the list of subtests to be performed. The subtests for `tl` are separated by the + (plus) character. The supported subtests are DES, 3DES, DSA, RSA, and RNG, so `tl=DES+3DES+DSA+RSA+MD5+SHA1+RNG` enables all subtests. You can also insert `tl=all` which performs all tests. Defaults to `all` if no subtests are specified.

Note – 64-bit tests are located in the `/bin/64` directory, or the relative path in which you installed SunVTS. If a test is not present in this directory, then it might be available as a 32-bit test only. For more information, see "32-Bit and 64-Bit Tests" on page 5.

CHAPTER 3

Disk and Floppy Drives Test (disktest)

disktest verifies the functionality of hard drives and diskette drives using three subtests (see TABLE 3-1): Media, File System, and Asynchronous I/O.

Note – disktest does support x86 platforms on Solaris.

Most disk drives, such as SCSI disks, native or SCSI floppy disks, IPI, and so on, are supported. The type of drive being tested is displayed at the top of the Test Parameter Options dialog box.

Note – disktest is supported on x86 platforms that use the Solaris Operating System. For disks on x86 machines, the disk partitions could range from 0 to 15. disktest can be performed on any of these selected partitions.

The disktest Test Parameter Options dialog box shows all the partitions that are available for testing. The file System subtest can only be run if the selected partition is mounted (described below). The WriteRead option of the Media subtest is allowed only if a selected partition is *not* mounted.

disktest Test Requirements

By default, disktest does not mount any partitions. To have SunVTS pre-mount all mountable partitions, set the environment variable BYPASS_FS_PROBE to 0 (zero) before starting SunVTS. Pre-mounting can be disabled by unsetting BYPASS_FS_PROBE or changing it to a value other than 0 (zero).

The mount point used by `disktest` is the word *disktest* appended by the name of the disk partition. For example, if the disk partition name is `/dev/dsk/c0t3d0s0`, `disktest` mounts it as superuser under the name `/disktest_c0t3d0s0`.

Caution – If a power failure occurs OR if the disktest is terminated abruptly while the Media subtest is running in WriteRead mode, disk data may be corrupted.

Caution – Running the Media subtest on a disk partition in the WriteRead mode may cause data corruption if the same partition is being used by other applications. Please run SunVTS in the offline mode only when there are no other applications running.

`disktest` tests the floppy drive regardless of whether the Volume Management software is running or not. The following mount point names are used:

- If the Volume Management software *is* running, `disktest` tests the disk drive with the mount point name in the `/etc/mnttab` file.
- If the Volume Management software *is not* running, `disktest` tests the disk drive with the device name `dev=/dev/diskette`. Do not edit the `/etc/vold.conf` file to change the diskette drives. Currently, the SunVTS software is hard-coded to use these path names as the default logic names.

Loading an option file (refer to the *SunVTS User's Guide* for option file details) that was created when `BYPASS_FS_PROBE` was set to `0` (zero) might not work if the `BYPASS_FS_PROBE` environment variable is no longer set to `0`. Testing may fail with the following error:

```
SUNWvts.disktest.8088 07/24/98 15:47:22 disktest c0t0d0 FATAL:
"Couldn't get file system information on /disktest_s0t0d0s0,
statvfs() system call failure error: No such file or directory.
```

This error is caused when SunVTS expects to use the predefined mount point names that are created when `BYPASS_FS_PROBE` is set to `0` (zero), but these mount points do not exist while `BYPASS_FS_PROBE` is not set to `0`.

To use option files with `disktest`, create two separate option files for the two different states of the `BYPASS_FS_PROBE` environment variable.

When a large number of `disktest` instances are run in write/read mode, tests might fail with messages similar to the following.

```
03/22/03 03:33:40 ctech140 SunVTS5.1ps2: VTSID 8011 disktest.FATAL
c1t0d0: "Failed lock mtab semaphore. "semop" system call failure,
errmsg: Invalid argument." Probable_Cause(s): <disktest instances
exceeds system semaphore operation limitation (default system
limit for seminfo_semmnu = 30)><System software error>
Recommended_Action(s): <Add the line "set semsys:seminfo_semmnu=
0x100" to your /etc/system file and reboot the machine> <If the
problem persists, call your authorized Sun service provider.
```

To avoid this issue, add the following entry to the `/etc/system` file and reboot the system.

```
set semsys:seminfo_semmnu=0x100
```

disktest Subtests

The following table describes the disktest subtests:

TABLE 3-1 disktest Subtests

Subtest	Description
Media subtest	The Media subtest verifies the disk media by allowing users to run disktest in different modes such as ReadOnly, ReadCompare, and WriteRead. The Media subtest treats the disk partition as one large chunk of contiguous data. In the WriteRead mode, all instances of disktest communicate through a shared memory service to ensure that they do not overlay the same disk area at the same time. This avoids data corruption. Each of the above three modes could run two different methods of disk testings. These are Synchronous I/O and Asynchronous I/O. SyncIO: Test reads and writes data using Read/Write system calls in a sequential fashion until the specified percentage of media is covered. AsyncIO: Test reads and writes data using aio library calls such as aioread(), aiowrite() until the specified percentage of media is covered. aiowait() is used to synchronize aio operations.
File System subtest	The File system subtest is used to verify the disk file system integrity. It exercises mounted disk partitions carrying the file system. By default, the test only runs on system-mounted partitions, it does not pre-mount any additional partitions. If you want SunVTS to pre-mount all of the unmounted partitions which have a file system, you have to set the environment variable BYPASS_FS_PROBE to '0' (zero). The test creates two temporary files of the size specified by File System File Size, writes the data patterns and compares the two files against each other.

disktest Test Options

To reach the dialog box below, right-click on the test name in the System Map and select Test Parameter Options. If you do not see this test in the System Map, you might need to expand the collapsed groups, or your system may not include the device appropriate to this test. Refer to the *SunVTS User's Guide* for more details.

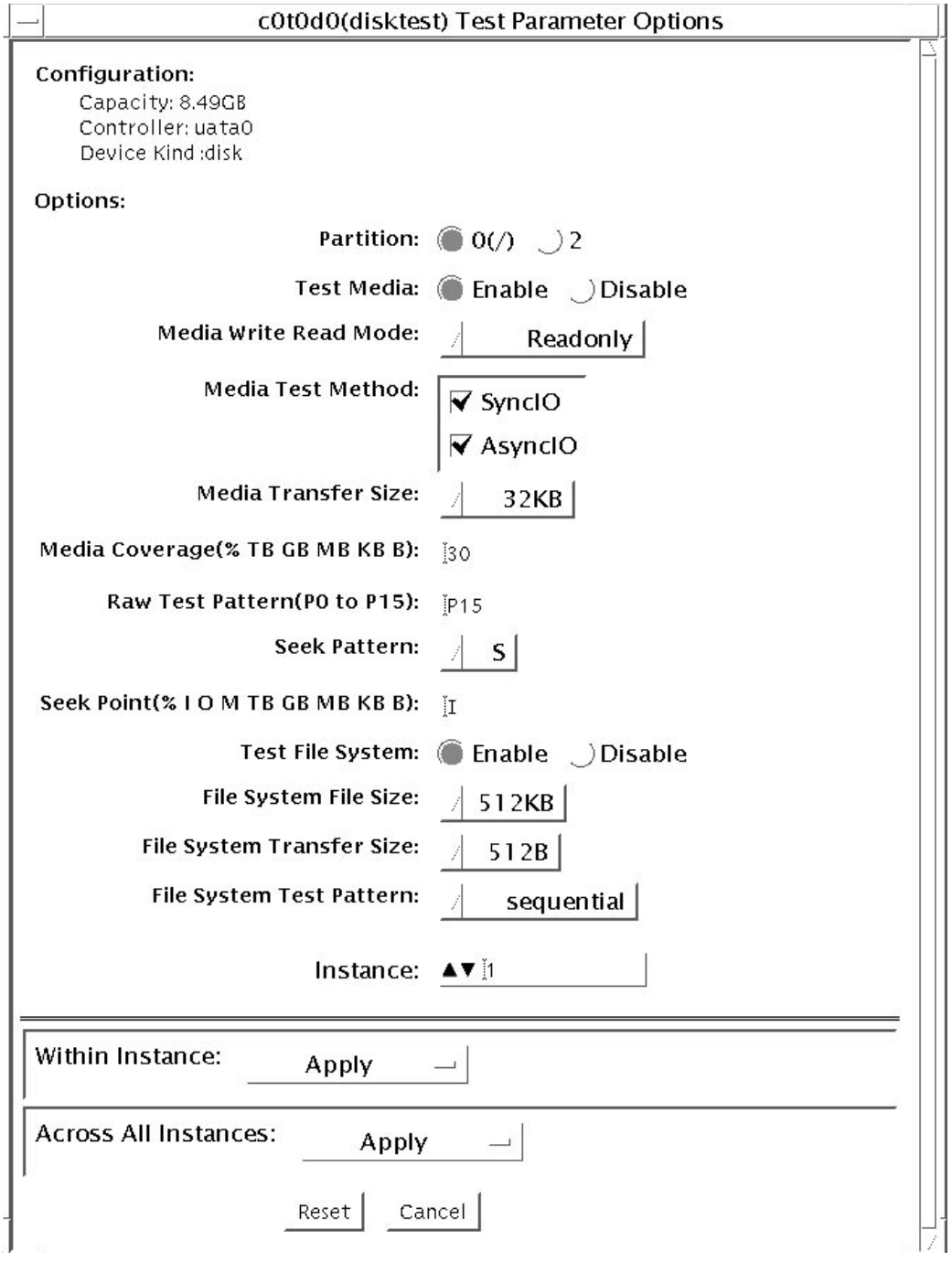

FIGURE 3-1 `disktest` Test Parameter Options Dialog Box

The following table describes the `disktest` option menu for different test modes.

TABLE 3-2 `disktest` Configurations and Options

`disktest` **Options**	**Description**
Partition	Displays the partition for the Media subtest. If a partition is mounted, its mount point is appended after the partition number, such as `1(/usr)`, where `1` is the partition number, and `"(/usr)"` is the mount point.
Test Media	Enable or Disable the media subtest.
Media Write Read Mode	Selects Read-Only or Compare after Read or Read after Write.
Media Test Method	Selects the Media Test Methods (SyncIO and AsyncIO).
Media Coverage (% TB, GB, MB, KB, B)	Enables users to test all or part of a partition (in percentage or in any of TB, GB, MB, KB, B units)
Raw Test Pattern (P0 to P16)	Enables user to specify the write, read pattern. • P0 – Low Frequency Pattern • P1 – Low Transition Density Pattern • P2 – High Transition Density Pattern • P3 – Compliant Jitter Pattern • P4 – Compliant Jitter: RPAT • P5 – Compliant Jitter: CRPAT • P6 – Compliant Jitter: JTPAT • P7 – Compliant Jitter: CJTPAT • P8 – Compliant Jitter: SPAT • P9 – Compliant Jitter: CSPAT • P10 – 8 Bit Cable Pattern • P11 – 16 Bit Cable Pattern • P12 – 8 Bit Xtalk Pattern • P13 – 16 Bit Xtalk Pattern • P14 – MFM Pattern • P15 – Generic Test Patterns • P16 – SATA Test Patterns
Seek Pattern	Enables specifying the pattern of the disk head movement. • S – Sequential • SR – Sequential Reverse • LS – Low Power Sequential • R – Random • LB – Low Power Butterfly • LR – Low Power Reverse Butterfly • AB – Actuator Butterfly • AR – Actuator Reverse Butterfly

TABLE 3-2 `disktest` Configurations and Options *(Continued)*

`disktest` **Options**	**Description**
Seek Point (%, I, O, M, TB, GB, MB, KB, B)	Enables specifying the seek point offset for the I/O. You can specify the offset in percentage or any of TB, GB, MB, KB, B or and I, M, O; that is, Initial, Middle), Outer.
Media Transfer Size	Displays the transfer size of the media subtest.
Test File System	Selects the File System subtest.
File System File Size	Specifies the size for each of the two temporary files for File System testing.
File System Transfer Size	Displays the transfer size of the File System subtest.
File System Test Pattern	Test pattern of File System subtest.
Connection Test for Hard Disk	• Option Menu for hard disk partition—0 - 7 [default] • Test Media—[Enable] (fixed to Enable) • Media Write Read Mode—[Read Only] (fixed to Read Only) • Media Test Method-[SyncIO] (fixed to SyncIO) • Media Coverage(%)—1 • Media Transfer Size—[2 KB] • Test File System—[Disable] (fixed to Disable)

TABLE 3-2 `disktest` Configurations and Options *(Continued)*

`disktest` **Options**	**Description**
Online Mode for Hard Disk	• Partition—0 - 7 [default] • Test Media—[Enable] [Disable] • Test Mode—[Read-only~] (fixed to Read-only) • Media Coverage (% TB GB MB KB B)—[10~] (fixed to 10%) • Media Transfer Size—[2KB~] (fixed to 2 KB) • Test File System—[Disable~] (fixed to Disable) • Media Test Method—[SyncIO] [AsyncIO] • Raw Test Pattern—[P15~] (fixed to P15) • Seek Pattern—[S~] (fixed to S) • Seek Point (% TB GB MB KB B)—[I~] (fixed to I)
Functional Test for Hard Disk	• Partition—0 - 7 [default] • Test Media—[Enable] [Disable] • Media Write Read Mode—[Readonly] [CompareRead] [WriteRead] • Media Test method—[SyncIO] [AsyncIO] • Media Coverage (% TB, GB, MB, KB, B) • Raw Test Pattern (P0 to P16) • Media Transfer Size—[2KB] [16KB] [32KB] [64KB] [128KB] [256KB] [512KB] • Test File System—[Enable] [Disable] • File System File Size—[512KB] [2MB] [8MB] [20MB] [100MB] [200MB] • File System Transfer Size—[512B] [1024B] [10KB] [40KB] [80KB] • File System Test Pattern—[sequential] [0x00000000] [0xffffffff] [0x5aa55aa5] [0xdb6db6db] [random] • Seek Pattern —[S~] (fixed to S) • Seek Point (% TB GB MB KB B)—[I~] (fixed to I)
Functional Test for Floppy Disk	• (under Other-Devices group)—partition: 0 - 7 [default] • Test Media—[Enable] [Disable] • Media Write Read Mode—[Read-only] [BackupWriteRead] • Media Test Method—[SyncIO] [AsyncIO] • Media Coverage (% TB, GB, MB, KB, B) • Raw Test Pattern (P0 to P16) • Media Transfer Size—[2KB] [10KB] [20KB] • Test File System—[Enable] [Disable] • Floppy File Size— [100KB] [200KB] • Floppy Transfer Size—[512B] [1024B] [10KB] • File System Test Pattern—[sequential] [0x00000000] [0xffffffff] [0x5aa55aa5] [0xdb6db6db] [random]

disktest Test Modes

TABLE 3-3 disktest Supported Test Modes

Test Mode	Description
Connection	Only one instance of disktest (which monitors UNIX error messages) is allowed for each disk device. disktest displays messages and reports errors. The test also opens the hard disk, checks the disk configuration, reads a few blocks, and then closes the hard disk. No File System subtest is run. No Write option is available in Connection test mode.
Functional	More than one instance of disktest is allowed for one disk device. The File System subtest, Media subtests, and floppy test can be run in Functional test mode. In Functional mode, disktest performs an additional subtest (Write/Read device buffer subtest) for enclosures.
Online	SunVTS disktest runs the Read Only rawtest with fixed transfer size and fixed rawtest pattern. Both SyncIO and AsyncIO test methods are available. The File system subtest is disabled in the Online test mode. Only one disktest instance could be run in the Online test mode.

disktest Command-Line Syntax

/opt/SUNWvts/bin/disktest *standard_arguments* **-o partition=**<*0-7*> [*"<(mount_point)>"*]**, rawsub=***E(nable)*|*D(isable)***, rawrw=** *Readonly*|*CompareRead*|*WriteRead***, rawiosize=**<*number*>{*...*|*KB*|*kb...*}|*random***, rawcover=**<*number*>|<*number*>{*TB*|*GB*|*MB*|*KB*|*B*|*tb*|*gb*|*mb*|*kb*|*b*} **rawpattern=***P(<0-16>)*|*0x<8 digit data pattern***>, seekpattern=** {*S*|*SR*|*LS*|*R*|*LB*|*LR*|*AB*|*AR*}**, seekpoint=**{*i*|*m*|*o*|<*number*>}**, method=**

AsyncIO+SyncIO, **fssub=***E(nable)* | *D(isable)*, **fssize=** *<number>{K | KB | M | MB | k | kb | m | mb}*, **fsiosize=***<number>{K | KB | B | k | kb | b}*, **fspattern=***<data_pattern>*, **dev=***<device_name>*

TABLE 3-4 `disktest` Command-Line Syntax

Argument	Description
partition=*<0-7>* *[“<(mount_point)>”]*	Specifies the partition number as follows: • *n*—is the partition number (slice number), usually *0-7* for SPARC and *0-16* for x86 • *mount_point*—is the mount point for the mounted partition that you plan to test For example: **partition=6”(/export)”**
rawsub= *E(nable)	D(isable)*	Enables or disables the Media subtest. For example: **rawsub=** *Enable*
rawrw= *Readonly	CompareRead	WriteRead*	Specifies the Media subtest Read, Compare, and Write mode: • Read only • Read twice, Compare (works only with SyncIO method) • Write, Read, Compare, Restore For example: **rawrw=***ReadOnly*
rawiosize= *<number>{...	KB	kb...}	random*	Specifies the media size to transfer. The block size can be specified in kilobytes. For example: 2K,...512K. For example: **rawiosize=**9
rawcover= *<number>	<number>{TB	GB	MB	KB	B	tb	gb	mb	kb	b}*	Specifies media coverage from 0-100 (percentage) of the partition. Media Coverage can also be specified in units: TB, GB, MB, KB and B. For example: **rawcover=***40* OR **rawcover=***4GB*

TABLE 3-4 `disktest` Command-Line Syntax *(Continued)*

Argument	Description
`rawpattern=`*P(<0-16>)* | *0x<8 digit data pattern>*	`rawpattern` could be specified as a pre-defined pattern set, *P(0-16)*, or an 8 digit pattern could be specified as: *0xaa55aa55+0xff00ff00+0x*. The following is a description of the supported pre-defined patterns: • P0 – Low Frequency Pattern • P1 – Low Transition Density Pattern • P2 – High Transition Density Pattern • P3 – Compliant Jitter Pattern • P4 – Compliant Jitter: RPAT • P5 – Compliant Jitter: CRPAT • P6 – Compliant Jitter: JTPAT • P7 – Compliant Jitter: CJTPAT • P8 – Compliant Jitter: SPAT • P9 – Compliant Jitter: CSPAT • P10 – 8 Bit Cable Pattern • P11 – 16 Bit Cable Pattern • P12 – 8 Bit Xtalk Pattern • P13 – 16 Bit Xtalk Pattern • P14 – MFM Pattern • P15 – Generic Test Patterns • P16 - SATA Test Patterns For example: **`rawpattern=`***P1*
`seekpattern=` *{S | SR | LS | R | LB | LR | AB | AR}*	`seekpattern` could be specified to select the type of seek test to run on the disk drive. `disktest` supports the following pattern types: • `S` – Sequential • `SR` – Sequential Reverse • `LS` – Low Power Sequential • `R` – Random • `LB` – Low Power Butterfly • `LR` – Low Power Reverse Butterfly • `AB` – Actuator Butterfly • `AR` – Actuator Reverse Butterfly For exmaple: `seekpattern=S`

TABLE 3-4 `disktest` Command-Line Syntax *(Continued)*

Argument	Description
seekpoint=*{i | m | o | <number>}*	Specify the seek-point for the I/O. This could be specified either in terms of the range - inner, middle and outer. Or in terms of absolute seek location. The absolute location is specied by a number followed by any of the following units {TB | GB | MB | KB | B | tb | gb | mb | kb | b}. For example: a) `seekpoint=`*I*, start the I/O from block 1. b) `seekpoint=`*M*, start the I/O from middle offset of the partition.
method=*AsyncIO+SyncIO*	Specifies the Media access method. You can choose to use either or both methods . If you use both access methods together, you must insert a '+' between the two: AsyncIO: Runs the asynchronous I/O test, using the async read/write feature of the Solaris disk driver SyncIO: Runs the synchronous I/O test. For example: **method=***AsyncIO*
fssub=*E(nable) | D(isable)*	Enables or disables the File System subtest. File system subtest runs on a mounted partition with a file system.
fspattern=<*data_pattern*>	Specifies the file system data pattern as sequential or random or one of the patterns selected from the list. *{seq(uential) | 0x0(0000000) | 0xf(ffffffff) | 0xa (5a5a5a5) | 0x5(a5a5a5a) | ran(dom) | 0xd(b6db6db)}* For example: a) **fspattern=***0xa* a) **fspattern=***seq*
fssize= <*number*>*{K | KB | M | MB | k | kb | m | mb}*	Indicates the file system subtest size in Megabytes or Kilobytes: • `K | k | KB | kb` – kilobytes • `M | m | MB | mb` – megabytes `512KB | 2MB | 8MB | 20MB | 100MB | 200MB`
fsiosize= <*number*>*{K | KB | B | k | kb | b}*	Indicates the size of the file system subtest I/O transfer in bytes or Kilobytes: • B | b – bytes • `K | k | KB | kb` – Kilobytes `512B | 1024B | 10KB | 40KB | 80KB`
dev=*device_name*	Specifies the name of the disk to be tested. For example: c0t3d0.

The following example shows how to run `disktest` on a partition "0" (which is mounted under "/") for the disk device `c0t0d0`. The media subtest is enabled in ReadOnly mode using SyncIO method. The coverage specified is 30% with 512 KB transfer size. The File System subtest is disabled.

```
# /opt/SUNWvts/bin/disktest -f -o partition=0"(/)", rawsub=Enable,
rawrw=ReadOnly, method=SyncIO, rawcover=30, rawiosize=512KB,
fssub=Disable, dev=c0t0d0
```

Note – 64-bit tests are located in the `/bin/64` directory, or the relative path in which you installed SunVTS. If a test is not present in this directory, then it might be available as a 32-bit test only. For more information, see "32-Bit and 64-Bit Tests" on page 5.

CHAPTER 4

Fast Frame Buffer Test (`ffbtest`)

`ffbtest` verifies the functionality of the fast frame buffer (FFB).

`ffbtest` can detect and adapt to the video modes of single- and double-buffer versions of the FFB. All `ffbtest` tests can run in several screen resolutions such as standard, stereo, and high resolution. In stereo mode, all tests write into the right and left eyes unless you specify otherwise. This test also supports FFB, FFB2, and FFB2+ fast frame buffer configurations. Use the `ffbconfig -prconf` command to display the configuration of the frame buffer you want to test.

You can interrupt `ffbtest` using Control-C.

Test accuracy is checked using a checksum algorithm. Possible locations of failing pixels are identified, as well as the likely failing FRU.

Caution – *Do not* run any other application or screen saver program that uses the FFB accelerator port while running `ffbtest`. These programs cause SunVTS to return incorrect errors. To ensure you are not running any wrong programs perform the following procedures.

Turn off the screen saver:

```
# /usr/openwin/bin/xset s off
```

Turn off the Energy Power features:

```
# /usr/openwin/bin/xset -dpms
```

Turn off the Frame Buffer Power features:

```
# /usr/openwin/bin/xset -fbpm
```

Edit the `autopm` parameter in the `/etc/power.conf` file as follows:

```
# Copyright (c) 1996 - 1999 by Sun Microsystems, Inc.
# All rights reserved.
#
#pragma ident"@(#)power.conf1.1499/10/20 SMI"
#
# Power Management Configuration File
#
# NOTE: The entry below is only used when no windowing environment
# is running.  When running windowing environment, monitor power
# management is controlled by the window system.

device-dependency /dev/fb /dev/kbd

device-thresholds /dev/kbd  always-on

# Auto-ShutdownIdle(min)Start/Finish(hh:mm)Behavior
autoshutdown30    9:00 9:00unconfigured

autopm        disable
```

`ffbtest` Test Requirements

Disable all screen savers before testing any graphics device. Type **`xset s off`** at a UNIX prompt to disable the Solaris screen saver. Disable the Power Management™ software if it is running.

For full instructions on testing frame buffers, see The *SunVTS 5.1 Test Reference Manual*.

`ffbtest` requires approximately 7 MB of disk space in the `/tmp` directory to extract its working files. If this space is not available, the diagnostic will fail and report warning and error messages, indicating a lack of disk space.

To start SunVTS with `vtsui`, but without `vtsk`, you must add the host name to `xhost` as: **`xhost +`** *<hostname>*.

ffbtest Options

To reach the dialog box below, right-click on the test name in the System Map and select Test Parameter Options. If you do not see this test in the System Map, you might need to expand the collapsed groups, or your system may not include the device appropriate to this test. Refer to the *SunVTS User's Guide* for more details.

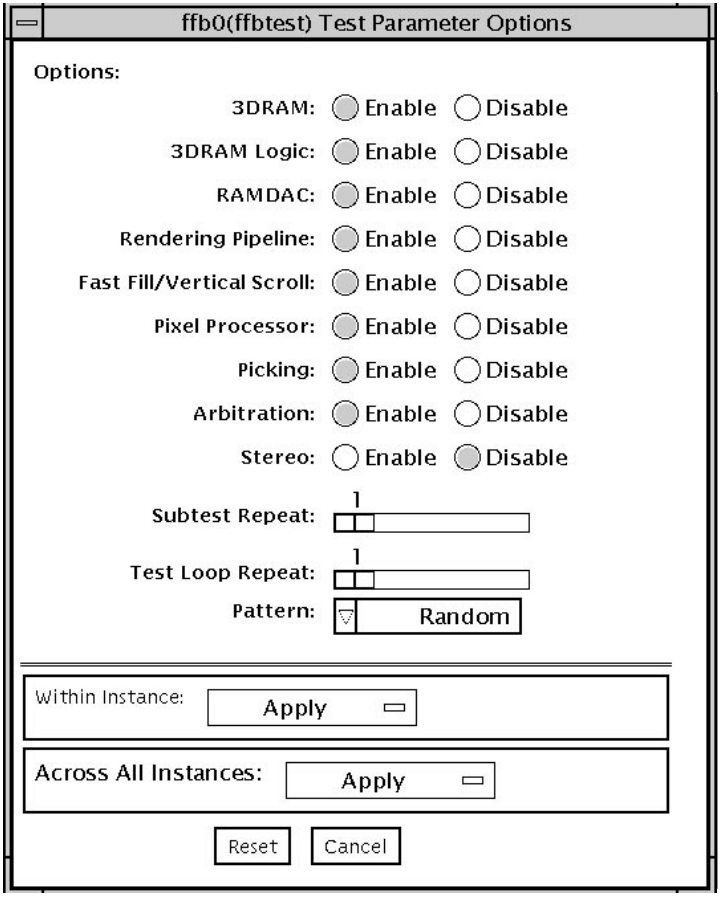

FIGURE 4-1 `ffbtest` Test Parameter Options Dialog Box

By default, all tests are enabled except the stereo test.

TABLE 4-1 `ffbtest` Options

`ffbtest` **Options**	**Description**
3DRAM test	The 3DRAM test thoroughly tests the video memory in the FFB using 512-bit reads and writes. 3DRAM makes a full screen pass, consisting of a write and a read to each pixel location, for each access mode on the list below. You can use either random data or specify data at the command line. A second pass is made with the one's complement of the data used in the first pass so that each memory location is tested with both a zero and a one. Notice that some passes are skipped on the single-buffered FFB. Errors in this subtest are attributes to the 3DRAM. A failing chip is indicated by (X, Y) locations and device-specific "U" numbers: • DFB8R, DFB8G, DFB8B, DFB8X—Buffer A • DFB24—Buffer A • DFB32—Buffer A • SFB8R, SFB8G, SFB8B, SFB8X—Buffer A • SFB8R, SFB8G, SFB8B, SFB8X—Buffer B (double buffer only) • SFB32—Buffer A • SFB32—Buffer B (double buffer only) • SFB32—Buffer C (double buffer only) • SFB64—Buffers A and C (double buffer only) • SFB64—Buffers B and C (double buffer only)

TABLE 4-1 `ffbtest` Options *(Continued)*

`ffbtest` **Options**	**Description**
3DRAM Logic test	3DRAM Logic provides logical functionality to the FFB. The following services are tested: • Compare Controls—Match AB • Compare Controls—Magnitude AB • Compare Controls—Match C (double buffer only) • Compare Controls—Magnitude C (double buffer only) • Match Mask—AB • Magnitude Mask—AB • Match Mask—C (double buffer only) • Magnitude Mask—C (double buffer only) • Raster Operations—RGB • Raster Operations—X • Raster Operations—YZ (double buffer only) • Plane Mask—RGB • Plane Mask—X • Plane Mask—Y • Plane Mask—Z • Group Enable—R, G, B, X • Group Enable—Y, Z (double buffer only)
3DRAM Logic test (Continued)	`ffbtest` tests each function separately with a series of SFB64 writes. A total of 16 writes are made for each different test case with Y coordinate values varying from 0 to 30 in increments of 2 pixels. This dotted column organization provides page thrashing and block flashing in all screen resolutions. For each operation, all possible combinations are tested. For example, `ROP RGB new== old` has three possible values: `new < old`, `new == old`, *and* `new > old`. `ffbtest` tests each of these cases. Five passes of the functions are made. Each pass writes into a different FFB address space: SFB32-A, SFB32-B, SFB32-C, SFB64-AC, and SFB64-BC. Note that the passes that write into the SFB32 address spaces are writing two pixels at a time because the tests use SFB64 writes. For FFB2+ boards, additional testing is performed on the new stencil and passin capabilities if the board is DBZ. Care is taken to ensure that all 3DRAM chips are tested. Errors in this subtest are attributed to the 3DRAM.

TABLE 4-1 `ffbtest` Options *(Continued)*

`ffbtest` **Options**	**Description**
RAMDAC Test	RAMDAC registers are tested using simple read/write patterns to determine if there are any bad bits. This includes all LUTs. `ffbtest` ensures that data is actually being read from the RAMDAC and not being supplied by the driver. Next, the RAMDAC Signature Register captures the pixels going to the screen. This test determines that all of the different data paths within the RAMDAC are functioning properly. The following modes are tested: • 24-bit true color from A • 24-bit true linear color from A • 24-bit direct color from A • 24-bit true color from B (double buffer only) • 24-bit true linear color from B • 24-bit direct color from B (double buffer only) • 8-bit pseudo color (from each plane in RGB) from A • 8-bit pseudo color (from each plane in RGB) from B (double buffer only) • 8-bit non-linear grayscale (from each plane in RGB) from A • 8-bit non-linear grayscale (from each plane in RGB) from B (double buffer only) • 8-bit linear grayscale (from each plane in XRGB) from A • 8-bit linear grayscale (from each plane in XRGB) from B (double buffer only) • 8-bit overlay pseudo color (from buffer A, X plane)
RAMDAC test (Continued)	This test displays a total of 11 different types of windows on the screen for the single-buffered configuration; 22 for double-buffered. A cursor is also displayed on the screen. RAMDAC on FFB2+ board supports three modes (SEP8, SEP4, and Combined). This test detects the RAMDAC type and tests the original and additional features like increased number of CLUTs, increased WLUT size, additional overlay WLUT. Errors in this test are attributed to the RAMDAC.

TABLE 4-1 `ffbtest` Options *(Continued)*

`ffbtest` **Options**	**Description**
Rendering Pipeline test	Rendering Pipeline uses the rendering pipeline tests developed for the FFB stand-alone diagnostics. Each primitive is tested thoroughly with the following sources and configurations: • Dots • Anti-aliased dots • Lines using all four line drawing primitives • Triangles • Polygons • Rectangles • Fonts Errors in this test are attributed to the FBC.
Fast Fill/Vertical Scroll test	The Fast Fill/Vertical Scroll primitives are separated from the Rendering Pipeline tests because of their dependence on screen type. There are three different tests, one for each screen type. Each test uses both block and page mode fistfuls. Errors in this test are attributed to the FBC.
Pixel Process test	Pixel Processor, a subtest, exercises the following options selected by the FFB's Pixel Processor Control (PPC) register: • Auxiliary clipping (additive and subtractive) • Depth cueing • Alpha blend • Viewport clip (2D and 3D) • Area pattern (transparent and opaque) Errors in this test are attributed to the FBC.
Picking test	The Picking test exercises the pick detect login of the 3DRAM. `ffbtest` defines a pick detect window and checks that writes to the window are picked, and writes outside the window are not picked. The test is repeated once for each 3DRAM. Errors in this test are attributed to the 3DRAM.

TABLE 4-1 `ffbtest` Options *(Continued)*

`ffbtest` **Options**	**Description**
Arbitration test	The Arbitration subtest continuously renders an object into the accelerator port while performing reads and writes through the direct port. For single-buffered configurations, a picture is rendered into the RGB planes while another process does DFB reads and writes in the X plane. For doubled buffered configuration, a picture is rendered into all 32 planes of the B buffer while the other does 32-bit DFB reads and writes in the A plane. This subtest simulates conditions in the real world, where rendering processes and windows operations run concurrently. Errors in this test are attributed to the FBC.
Stereo Test	The Stereo test displays an object in stereo mode with different images for the right and left eye. The user can verify proper operation by looking at the screen with stereo glasses and following the displayed instructions. If the monitor type is not 1280x1024 at 76MHz, this test prints a warning message and does not run. To prevent this message from being displayed or written to the SunVTS information log, disable the stereo test in the test option menu. Only Sony P4 and N2 monitors support stereo resolutions. This test temporarily switches the monitor into stereo mode, renders a stereo image, performs a signature analysis on the stereo image (using the RAMDAC signature capture register), and after displaying the image for five seconds, restores the monitor to its previous resolution. Errors in this test are attributed to the RAMDAC. Note — If vertical lines are displayed on the console when running SunVTS, this could be caused by the `ffbtest` stereo test. There is a time-critical period in the FFB hardware when trying to change the screen resolution from standard to stereo and back to standard. When the system is heavily loaded or running all of the SunVTS tests, the FFB device driver may get interrupted while changing screen resolution. If this occurs, FB ASIC and RAMDAC get out of synchronization, resulting in an unusual display on the FFB screen. This problem could also cause a system hang condition. To avoid this type of display problem, disable the `ffbtest` stereo test when other SunVTS tests are enabled. This test is disabled by default because it is only needed when a stereo monitor and stereo glasses are present.

`ffbtest` Test Modes

Due to the nature of graphic tests, reading from or writing to the frame buffer during graphic tests will disturb user operation. This test is only available in offline Functional test and Stress mode.

TABLE 4-2 `ffbtest` Supported Test Modes

Test Mode	Description
Functional (Offline)	The `ffbtest` verifies both the single- (SFB) and double-buffered (DBZ) fast frame buffer boards.
Stress mode	Stress mode exercises the frame buffer as much as possible. The Random test generator, constructed as part of the verification effort, is used. Starting from a known seed, random primitives with random attributes are generated. The primitives are checked to ensure that they were rendered in the same way as on a known good system. The test is repeated ten times, with each random picture overlaying the previous one. Stress mode is not available on FFB2+ boards.

ffbtest Command-Line Syntax

/opt/SUNWvts/bin/ffbtest *standard_arguments* **-o dev=***device_name,* **S=***subtest_number,***F=#***_of_subtest_loops,***B=#***_of_test_loops***,P=***test_pattern*

TABLE 4-3 ffbtest Command-Line Syntax

Argument	Description
dev=*device_name*	*device_name* is the relative path name of the device being tested with respect to /dev/fbs; The default is ffb0.
S=*subtest_number*	*subtest_number* is the test number of the subtest to be run. Select from the subtests below. You can run multiple subtests by adding the subtest numbers. For example, n=0x3 runs both test 1 and test 2; n=0x180 runs both test 0x080 and test 0x0100. Note that you do not need the leading zeros. • n=0x00001 3DRAM • n=0x00002 3DRAM Logic • n=0x00004 RAMDAC • n=0x00008 Rendering Pipeline • n=0x00010 FastFill/Vertical Scroll • n=0x00020 Pixel Processor • n=0x00040 Picking • n=0x00080 Arbitration • n=0x00100 Stereo More than one test can be selected by ORing subtest numbers. For example: n = 0x00009 selects 3DRAM and Rendering Pipeline tests. A hex number must be preceded by 0x, decimal numbers are also acceptable.
F=#*_of_subtest_loops*	Specifies the number of times to repeat each subtest. The default is 1.
B=#*_of_test_loops*	Specifies the number of times to repeat a test loop before passing. The default is 1.
P=*test_pattern*	Specifies the test pattern number. The default is r, for random patterns. You may also choose 0 for 0x0000000, 3 for 0x3333333, 5 for 0x5555555, or 9 for 0x9999999.

Note – 64-bit tests are located in the `sparcv9` subdirectory: `/opt/SUNWvts/bin/sparcv9/`*testname*. If a test is not present in this directory, then it may only be available as a 32-bit test. For more information refer to the *SunVTS 5.1 Test Referenct Manual*.

Note – Errors returned by `ffbtest` are nonspecific. It is not possible to determine which component caused a failure. In all error conditions, the field replaceable unit (FRU) is the entire FFB.

CHAPTER 5

Sun XVR-2200 and XVR-2400 Graphics Accelerator Test (`graphicstest`)

`graphicstest` verifies the functionality of XVR-2200 and XVR-2400 frame buffers. Test accuracy is checked using direct image comparison against compressed images. Failed pixel locations are printed as error messages.

Caution – Do not run any other application or screen saver program that uses the framebuffer's accelerator port while running graphicstest. This combination causes SunVTS to return incorrect errors.

Test Requirements

Disable all screen savers before testing any graphics device. To disable the Solaris screen saver, type the following at a UNIX prompt:

```
# xset s off
```

To turn Display Power Management off, type the following at a UNIX prompt:

```
# xset -dpms
```

The display resolution must be 1280x1024. To change resolution, go to a UNIX prompt and type:

```
# fbconfig -res 1280x1024x76
```

For full instructions on testing frame buffers, see "Testing Frame Buffers" on page 11.

Test Preparation

Perform the steps in this section prior to performing `graphicstest` to ensure the test runs as smoothly as possible.

1. You must run `graphicstest` through the SunVTS user interface. The window system (such as CDE or GNOME) must be running on the XVR-2200 or XVR-2400, and the window system must be setup to run in 24-bit mode (the default depth for XVR-2200 and XVR-2400).
2. Turn Power Management off, if it is enabled. See "Test Requirements" on page 47.
3. Verify that no other program is running that might modify the screen during the test.
4. Verify you have permission to lock the X server. `graphicstest` is designed to lock the X server during testing to prevent screen changes.
5. Verify that the CDE login window is not displayed during testing.
6. Verify that the window system is running on one frame buffer only.

`graphicstest` Options

To reach the dialog box below, right-click on the test name in the System Map and select Test Parameter Options. Because graphics test can test multiple types of frame buffers, the test name that is displayed will correspond to the particular framebuffer being tested. If you do not see this test in the System Map, you might need to expand the collapsed groups, or your system may not include the device appropriate to this test. Refer to the SunVTS User's Guide.

By default, all `graphicstest` options are enabled.

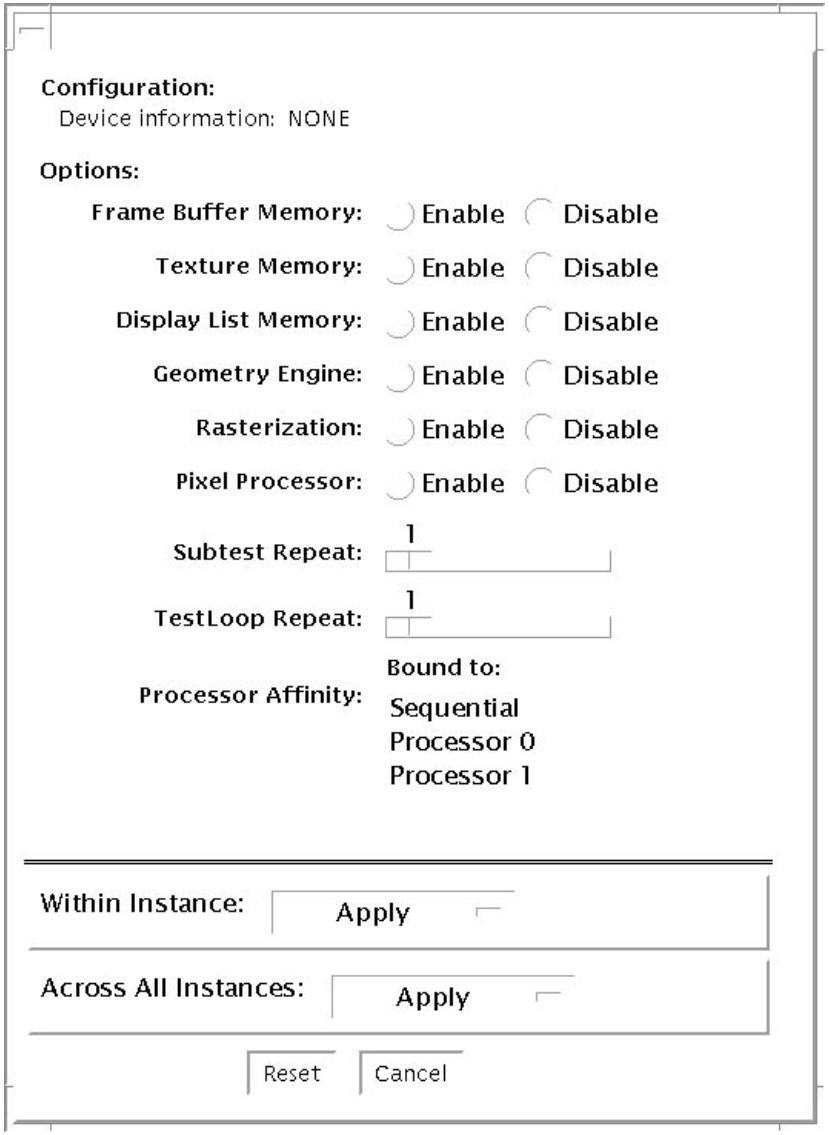

FIGURE 5-1 `graphicstest jfb0` Test Parameter Options Dialog Box

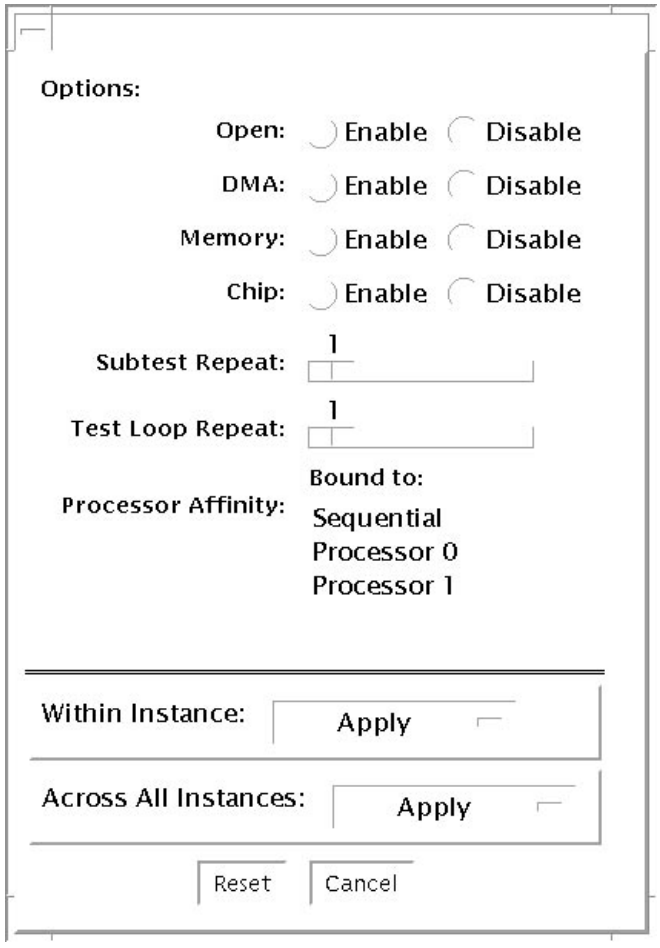

FIGURE 5-2 `graphicstest kfb0` Test Parameter Options Dialog Box

TABLE 5-1 `graphicstest` Options

Options	Description
Open test	The open test verifies that the graphics device can be correctly opened. First the device is opened with an open system call. Next the register and framebuffer regions are mapped into the virtual address space of the test process. Next a subset of the read/write resisters are written to and read back from to verify that the register region was correctly mapped. Next selected areas of the framebuffer are written to and read back from to see if the the framebuffer was correctly mapped. This test takes little time and no progress is displayed.
Memory test	The memory tests verifies that all the memory on the graphics card is working. The test opens the device, allocates all the device memory, and then writes to it using DMA or programmed I/O. The test then reads back the values from memory and checks to see if the correct values were read back.
DMA test	The DMA test verifies that the DMA engine on the device is functioning. The test writes data to an area of the frame buffer using DMA, then reads back the same area using programmed I/O and checks if the values are correct. The second part of the test writes data into the framebuffer and reads it back using the DMA engine and checks to see if the correct values were read back.
Chip Test	The chip test exercises the graphics chip on the board to see if it is functioning correctly. The chip test exercises the main data paths through the graphics chip. Commands are sent to the graphics board to initiate a graphics program. Once the program has run the results are read from the frame buffer to verify if they are correct.

`graphicstest` Test Modes

Due to the nature of graphic tests, reading data from, or writing data to the frame buffer during graphic tests will disturb user operation. For this reason, `graphicstest` is only available in offline Functional test mode.

TABLE 5-2 `graphicstest` Supported Test Modes

Test Mode	Description
Functional	Performs the entire set of tests offline.

graphicstest Command-Line Syntax

/opt/SUNWvts/bin/sparcv9/graphicstest *standard_arguments* -o [[dev=*device_name*] [open=*Enable/Disable*] [dma=*Enable/Disable*] [mem=*Enable/Disable*] [chip=*Enable/Disable*] [B=*n*] [F=*n*] [S=*value*]]

TABLE 5-3 graphicstest Command-Line Syntax

Argument	Description
dev=*device_name*	*device_name* is the relative path name of the device being tested with respect to /dev/fbs. There is no default.
open=*Enable/Disable*	Enables or disables the open test. Default is enable.
dma=*Enable/Disable*	Enables or disables the dma test. Default is enable.
mem=*Enable/Disable*	Enables or disables the memory test. Default is enable.
chip=*Enable/Disable*	Enables or disables the chip test. Default is enable.
B=*n*	Defines times to repeat each test loop Default is one.
F=*n*	Defines the number of times to repeat each subtest. Default is one.
S=*value*	Sets a bit mask to select which subtests will be used. • Bit 1, test open • Bit 2, test dma • Bit 3, test mem • Bit 4, test chip For example -o S=3 would run the open and dma tests only. The default is to run all tests

Note – 64-bit tests are located in the /bin/64 directory, or the relative path in which you installed SunVTS. If a test is not present in this directory, then it might be available as a 32-bit test only. For more information, see "32-Bit and 64-Bit Tests" on page 5.

CHAPTER 6

Infiniband Host Channel Adapter Test (`ibhcatest`)

`ibhcatest` is comprised of multiple iRISC CPU cores, two 4x IB ports and integrated SerDes components. In addition, the `ibhcatest` external associated components include FLASH ROM and DDR memory. `ibhcatest` provides high speed interconnect through PCI interface to external IB fabric. Supported platforms include: two 1U and two 2U x86 AMD Opteron entry-level servers, Sun Fire V2xx, V4xx and E series high end servers.

Note – `ibhcatest` supports the same set of options for both SPARC and x86 platforms.

`ibhcatest` provides a mechanism to exercise and verify the proper operation of the Tavor chip and its associated components, such as DDR memory, Flash PROM and internal IB packet transmit/receive circuitry. The goal is to isolate single fault to the identifiable component(s).

`ibhcatest` supports three execution test modes in SunVTS: Connection, Exclusive and Functional. In Connection mode, the test will provide a basic sanity check. This basic sanity test is done by querying for the Tavor firmware / hardware revision and running internal loopback.

The internal loopback test is run at least once depending on the amount of time each pass takes. In Functional mode all subtests are executed according to the options selected. In Exclusive mode all subtests are executed in sequence.

Tavor supports an internal loopback mechanism which is very similar to the actual operation. The main difference is that data does not go through the integrated SerDes and the 4x IB port circuitry. Also on the receiving side, data does not get verified by the CRC algorithm. Otherwise, all other components of Tavor that involve in transmitting and receiving data packets are being exercised.

Tavor based HCA is designed to use a single, 256 MB DDR memory for data storage at run time. This data storage is intended to be used and shared by three interdependent clients: Tavor driver, firmware, and hardware. During driver initialization, predetermined data structures and data are laid out in the memory.

With no exclusive atomic access from the driver side, subsequent writes to any memory location that contain real data can cause undesirable results like a system crash. Furthermore, the data allocation size is fixed, writing to the remaining free memory does not add any value in terms of finding faults.

With these constraints in mind, the memory subtest is limited to read only operations to cover the entire DDR memory. There is no checking for data corruption and no mechanism for triggering single/double bit type of errors through writing to memory. The resulting benefit of read operations is a secondary effect that occurs in the generation of high volume PCI activities from the memory accesses. Thus the test becomes a good exerciser to bring out bus related problems.

ibhcatest Subtests

TABLE 6-1 ibhcatest Subtests

Subtest	Description
Internal Loopback Test	The HCA supports internal loopback for packets transmitted between QPs that are assigned to the same HCA port. When transmitting a packet, if it is destined to a DLID that is equivalent to the Port LID with the LMC bits masked out or the packet DLID is a multicast LID, the packet goes on the loopback path. In this latter case, the packet also is transmitted to the fabric. When a packet is looped back, it must pass the SL2VL mapping. If the mapping yields 15 or a nonoperational VL, the packet is discarded. In the inbound direction, the ICRC and VCRC checks are "blindly" passed for looped back packets. Note that internal loopback is supported only for packets that are transmitted and received on the same port. Packets that are transmitted on one port and received on another port are transmitted to the fabric. The fabric should direct them to the destination port. This subtest uses interfaces from the Tavor driver to perform loopback testing. It is a push button type of test. Information such as data pattern for data packets, port number, CQ polling, retries between iteration, and the number of iterations for each `ioctl` call are passed to the driver. Once finished, status regarding the number passes completed is returned. If the number of passes does not match the number of iterations, a failure has occurred. This might happen when the number of retries is exhausted, and the last failing buffer in the retry series is returned as result. SunVTS then determines exactly what failed in the buffer and reports the failure. The options for this subtest are as follows: • `lb`=*Enabled* | *Disabled*: Turn on/off the loopback test • `tlbport`=*1+2*: Loopback test on Port 1 and/or 2, default is 1+2 • `data`=*Pattern*: Specific data pattern (default patterns 0xa5a5a5a5) • `cq`=*Time*: Number of CQ polling time value (in microseconds per iteration); default: 55000; max: 1000000. • `loop`=*Number*: Number of loopback iterations for each pass; default: 200; max: 1000 • `warn`=*Enabled* | *Disabled*: When enabled, prints a warning message
DDR READ Test	This subtetst is comprised of two test modes, Sequential and Random . The start and end address offsets are determined dynamically by obtaining them from the firmware. In Sequential mode, length and starting offset are instructed by the test option `rdoffset` and `rdsz`. Then the test goes through and sequentially reads data from each memory address until all memory locations are covered or the end address is reached. Each read is accomplished by an `ioctl` call to the driver. The test returns pass or fail based on the completion status of each `ioctl` call. In the `random test` method, this subtest reads the number of `rdsz` times in a randomly generated address bound by the start and end address offset. The options for this subtest are as follows: • `ddr`=*Enabled* | *Disabled*: Turn off/off the DDR Memory test • `rdoffset`=*Offset*: Starting offset of DDR Memory read, default is 0x0, in hexadecimal • `rdsz`=*Size*: Read number of byte of DDR Memory from Offset to Max address location, default: 0x2000; max: 256 MB, in hexadecimal

Caution – In SunVTS environment `ibhcatest` and `nettest` are mutually exclusive, `nettest` has higher priority if the IB port interface is plumbed up when SunVTS is invoked. These two tests can not be run at the same time at the command line, if both of these tests are invoked at the command line, `ibhcatest` exits gracefully if the IB port interface is plumbed up. The commands to bring down the IB daemon (`ibd` [`IPoIB`]) are as follows:

```
# ifconfig ibdXX down
# ifconfig ibdXX unplumb
```

Where *XX* is instance number of the interface.

`ibhcatest` Options

To reach the dialog box below, right-click on the test name in the System Map and select Test Parameter Options. Because graphics test can test multiple types of frame buffers, the test name that is displayed will correspond to the particular framebuffer

being tested. If you do not see this test in the System Map, you might need to expand the collapsed groups, or your system may not include the device appropriate to this test. Refer to the SunVTS User's Guide.

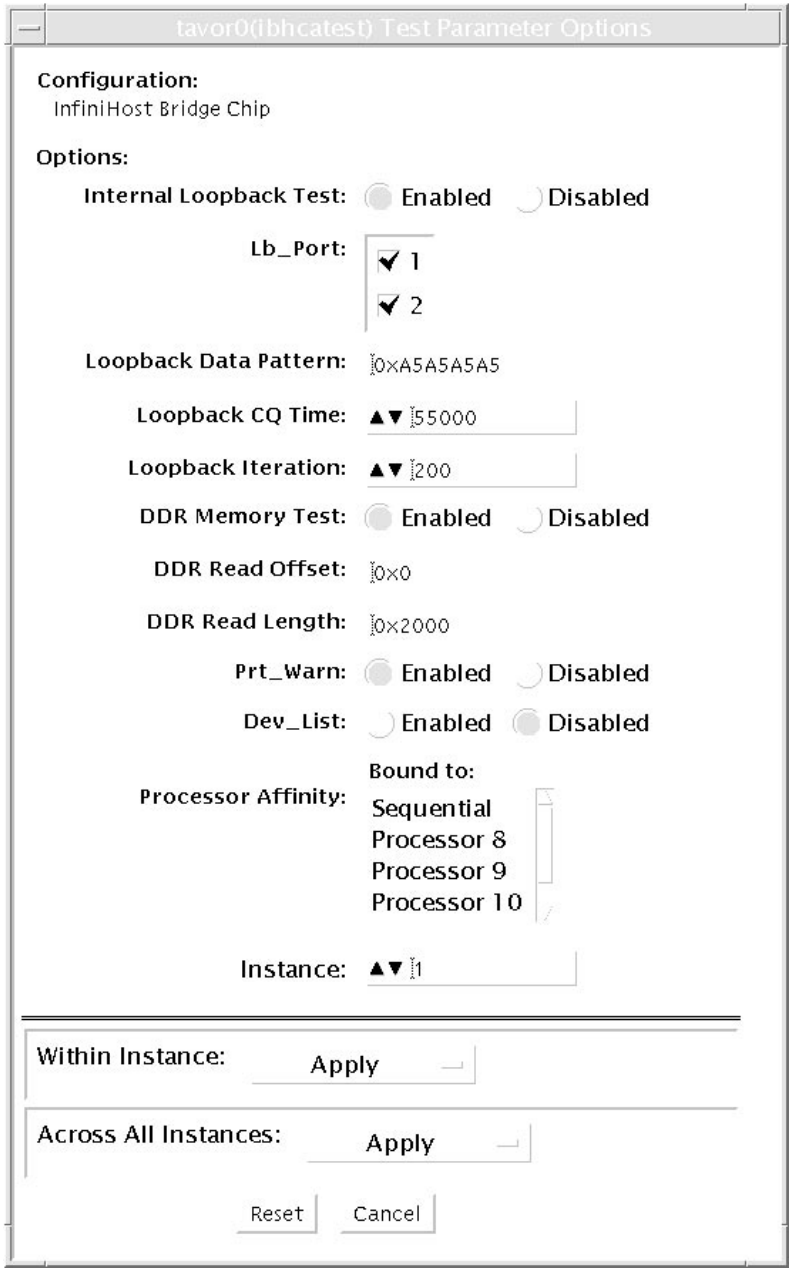

FIGURE 6-1 `ibhcatest` Test Parameter Options Dialog Box

TABLE 6-2 `ibhcatest` Options

Option	Description
`lb=`*Enabled* | *Disabled*	Turn on/off the loopback test
`tlbport=`*1*+*2*	Loopback test on Port 1 and/or 2, default is 1+2
`data=`*Pattern*	Specific data pattern (default patterns 0xa5a5a5a5)
`cq=`*Time*	Number of CQ polling time value (in microseconds per iteration), default: 55000; max: 1000000
`loop=`*Number*	Number of loopback iterations for each pass; default: 200; max: 1000
`ddr=`*Enabled* | *Disabled*	Turn on/off the DDR Memory test
`rdoffset=`*Offset*	Starting offset of DDR Memory read, default is 0x0, in hexadecimal
`rdsz=`*Size*	Read number of byte of DDR Memory from Offset to Max address location, default: 0x2000; max: 256 MB, in hexadecimal
`warn=`*Enabled* | *Disabled*	Print a warning message when enabled
`list`	Print device list, no testing when set

ibhcatest Test Modes

TABLE 6-3 ibhcatest Supported Test Modes

Test Mode	Description
Connection	Provides a basic sanity check by querying for the Tavor firmware / hardware revision and running internal loopback. The internal loopback test runs at least once depending on the amount of time each pass takes.
Exclusive	Executes all subtests sequentially.
Functional	Executes all subtests according to what is selected.

ibhcatest Command Line Syntax

ibhcatest [-scruvdtlxnf] [-p *n*] [-i *n*] [-w *n*] [-o [dev=*text*] [lb=*Enabled* | *Disabled*] [tlbport=*1+2*] [data=*Pattern*] [cq=*Time*] [loop=*Number*] [ddr=*Enabled* | *Disabled*] [rdoffset=*Offset*] [rdsz=*Len*] [warn=*Enabled* | *Disabled*] [list]]

Example:

```
# ibhcatest -p 0 -svf -o lb=Enabled, tlbport=1+2, data=0xA5A5A5A5,
cq=55000, loop=200, ddr=Enabled, rdoffset=0x0, rdsz=0x2000, warn=
Enabled, dev=tavor1
```

TABLE 6-4 ibhcatest Command Line Syntax

Option	Description
lb=*Enabled* | *Disabled*	Turn on/off the loopback test
tlbport=*1+2*	Loopback test on Port 1 and/or 2, default is 1+2
data=*Pattern*	Specific data pattern (default patterns 0xa5a5a5a5)
cq=*Time*	Number of CQ polling time value (in microseconds per iteration), default: 55000; max: 1000000.
loop=*Number*	Number of loopback iterations for each pass; default: 200; max: 1000
ddr=*Enabled* | *Disabled*	Turn on/off the DDR Memory test

TABLE 6-4 `ibhcatest` Command Line Syntax *(Continued)*

Option	Description
`rdoffset=`*Offset*	Starting offset of DDR Memory read, default is 0x0, in hexadecimal
`rdsz=`*Size*	Read number of byte of DDR Memory from Offset to Max address location, default: 0x2000; max: 256 MB, in hexadecimal
`warn=`*Enabled* | *Disabled*	Print a warning message when enabled
`list`	Print device list, no testing when set

Note – 64-bit tests are located in the `/bin/64` directory, or the relative path in which you installed SunVTS. If a test is not present in this directory, then it might be available as a 32-bit test only. For more information, see "32-Bit and 64-Bit Tests" on page 5.

CHAPTER 7

Sun™ XVR-1200 Graphics Accelerator Test (`jfbtest`)

`jfbtest` verifies the proper functioning of the Sun™ XVR-1200 graphics accelerator.

`jfbtest` can detect and adapt to many video modes of the Sun XVR-1200 graphics accelerator. All tests can run at a resolution of 1024x768 or higher.

You can interrupt `jfbtest` using Control-C.

Test accuracy is checked using direct image comparison against compressed images. Failed pixel locations are printed as error messages.

Caution – Do not run any other application or screen saver program that uses the Sun XVR-1200 graphics accelerator port while running `jfbtest`. This combination causes SunVTS to return incorrect errors.

`jfbtest` Test Requirements

Disable all screen savers before testing any graphics device. To disable the Solaris screen saver, type the following at a UNIX prompt:

```
# xset s off
```

To turn Power Management off, type the following at a UNIX prompt:

```
# xset -dpms
```

The display resolution must be 1024x768 or higher (the standard resolution). To change resolution, go to a UNIX prompt and type:

```
# fbconfig -res 1280x1024x76
```

For full instructions on testing frame buffers, see the Testing Frame Buffers section of the *SunVTS 5.1 Test Reference Manual*.

Preparation for `jfbtest`

You should complete a few steps in advance to ensure that `jfbtest` runs as smoothly as possible.

If you are running `jfbtest` in a window system (such as CDE):

- Turn Power Management off, if it is enabled. The following is an alternate way to turn Power Management off. Change `allowFBPM=1` to `allowFBPM=0` in `/platform/sun4u/kernal/drv/jfb.conf` file.
- Make sure that no other program is running that might modify the screen during the test.
- Make sure you have permission to lock the X server. `jfbtest` is designed to lock the X server during testing to prevent screen changes.
- The CDE login window should not be displayed during testing.
- Check that the window system is only running on one Sun XVR-1200 graphics accelerator.

If you are not running `jfbtest` in a window system:

- Turn Power Management off, if it is enabled. The following is an alternate way to turn Power Management off. Change `allowFBPM=1` to `allowFBPM=0` in `/platform/sun4u/kernal/drv/jfb.conf` file.
- Make sure that no other program is running that might modify the screen during the test.
- Make sure the Sun XVR-1200 graphics accelerator being tested is not the console device. Console messages may modify the screen.

`jfbtest` Options

To reach the dialog box below, right-click on the test name in the System Map and select Test Parameter Options. If you do not see this test in the System Map, you might need to expand the collapsed groups, or your system may not include the device appropriate to this test. Refer to the *SunVTS User's Guide* for more details.

By default, all `jfbtest` options are enabled.

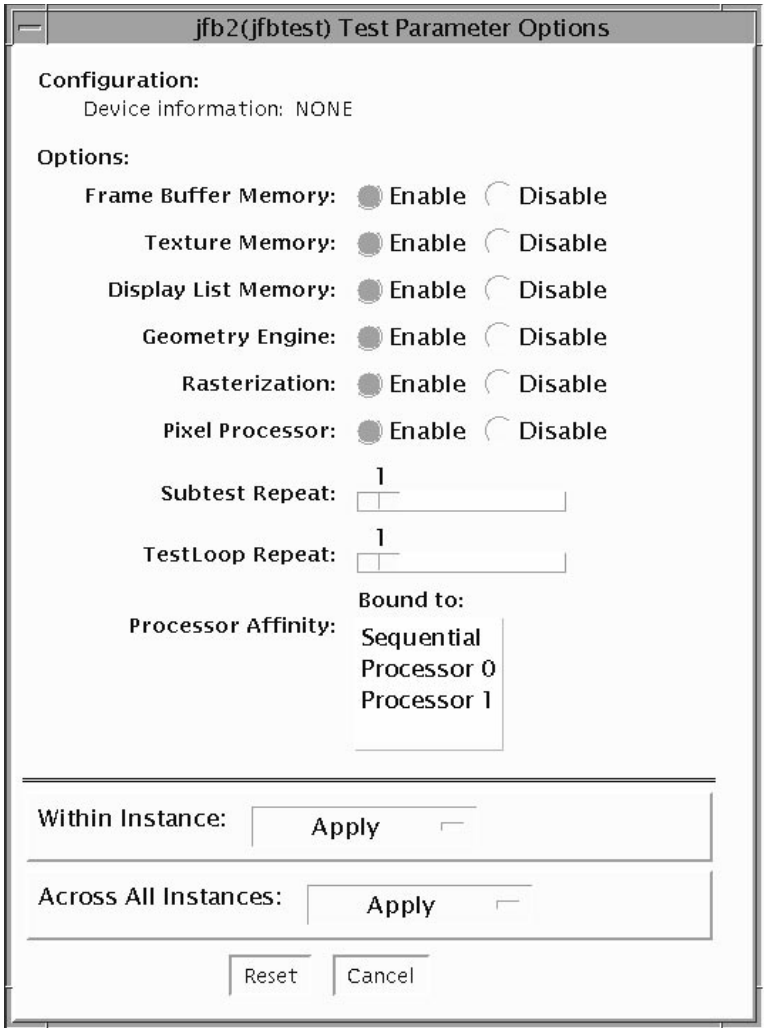

FIGURE 7-1 `jfbtest` Test Parameter Options Dialog Box

TABLE 7-1 `jfbtest` Options

`jfbtest` **Options**	**Description**
Frame Buffer Memory test	Thoroughly tests the Sun XVR-1200 video memory by using read and write requests. Tests for shorts or failed connections on the data bus by writing the following values to every address: • `0xFFFFFFFF` • `0xFFFF0000` • `0x0000FFFF` • `0xFF00FF00` • `0x00FF00FF` • `0xF0F0F0F0` • `0x0F0F0F0F` • `0xCCCCCCCC` • `0x33333333` • `0xAAAAAAAA` • `0x55555555` Tests for shorts or failed connections on the address bus by writing the offset of each memory location to each location and reading them back. This may also catch speed-related problems due to the volume of read/writes. Errors in the test are reported as an error in a particular address, not attributed to a specific chip. To help distinguish bit-related errors, the errors are summarized to list which bits had at least one error in the test. This test shows on the screen as random pixels.
Texture Memory test	This test is identical in process to the Frame Buffer Memory test (above). Since this test produces no visible effect, rectangles are drawn in rows across the screen to show progress.
Display List Memory test	This test is identical in process to the Frame Buffer Memory and Texture Memory tests (above), and is applied to direct burst memory. This test takes little time and no progress is displayed.
Geometry Engine test	Loads diagnostic microcode into the geometry engine and confirms that the processor operates correctly. This is a pass/fail test. This test takes little time and no progress is displayed.

TABLE 7-1 `jfbtest` Options

`jfbtest` **Options**	**Description**
Rasterization test	Renders many primitives with minimal fragment processing, to test the rasterization of the primitives. The primitives used are: • Dots • Anti-aliased dots • Lines using all for line-drawing primitives • Anti-aliased lines using all for line-drawing primitives • Triangles, Quads, and Polygons in point, line, and fill modes • Rectangles This tests for the following rasterization attributes: • Pixel coverage • Constant value registers for color, Z, and stencil • Interpolation of color, Z, and texture coordinates along lines and spans in polygons • Texture map sampling Resulting images are compared against stored images. Errors indicate which operation type and value was being tested, and the coordinate of the failed pixel.

TABLE 7-1 `jfbtest` Options

`jfbtest` **Options**	**Description**
Pixel Processor test	Tries the various pixel processing operators using a variety of fragment values. This tests the following fragment processing operations: • Depth Buffering • Blending • Alpha Test • Color Test • Color Clamp • Logic Operations • Color Matrix and Bias • Color Table • Control Planes • Fast Clear • Stencil • Scissor Clipping • Desktop Clipping • Mask Clipping • Write Masks • Window Origin • Fog • Pixel Texture • Accumulation Buffer • Pixel Buffers Resulting images are compared against stored images. Errors indicate which operation type and value was being tested and the coordinate of the failed pixel.

jfbtest Test Modes

Due to the nature of graphic tests, reading data from, or writing data to the frame buffer during graphic tests will disturb user operation. For this reason, jfbtest is only available in Functional test mode.

TABLE 7-2 jfbtest Supported Test Modes

Test Mode	Description
Functional	Runs the full set of tests.
Connection	Runs the full set of tests.

jfbtest Command-Line Syntax

/opt/SUNWvts/bin/jfbtest *standard_arguments* **-o dev=***device_name***, fbmem=***E(nable)/D(isable)***,texmem=***E/D***,dlmem=***E/D***,geomeng=***E/D***, rasterization=***E/D***,pixelproc=***E/D***,subtest_repeat=***number***, test_repeat=***number*

TABLE 7-3 jfbtest Command-Line Syntax

Argument	Description
dev=*device_name*	*device_name* is the relative path name of the device being tested with respect to /dev/fbs. There is no default.
fbmem=*E/D*	Enables or disables the Frame Buffer Memory test.
texmem=*E/D*	Enables or disables the Texture Memory test.
dlmem=*E/D*	Enables or disables the Display List Memory test.
geomeng=*E/D*	Enables or disables the Geometry Engine test.
rasterization=*E/D*	Enables or disables the Rasterization test.
pixelproc=*E/D*	Enables or disables the Pixel Processing test.
subtest_repeat=*number*	Defines the number of times to repeat each subtest. The default is 1.
test_repeat=*number*	Defines the number of times to repeat a test loop before passing. The default is 1.

Note – 64-bit tests are located in the `sparcv9` subdirectory: `/opt/SUNWvts/bin/sparcv9/`*testname, or the relative path to which you installed SunVTS*. If a test is not present in this directory, then it may only be available as a 32-bit test. For more information refer to the "32-Bit and 64-Bit Tests" section of the *SunVTS 5.1 Test Reference Manual* (816-5145-10).

CHAPTER **8**

Sun™ XVR-100 Graphics Accelerator Test (`pfbtest`)

`pfbtest` tests the PCI-based Sun™ XVR-100 graphics accelerator by performing the following subtests:

- Video Memory test
- RAMDAC test
- Accelerator Port test

Caution – *Do not* run any other application or screen saver program that uses the Sun XVR-100 graphics accelerator port while running `pfbtest`. This program causes SunVTS to return incorrect errors.

Caution – If pfb0b is set to display, an error similar to the following occurs:
Accelerator: signature err or in test Ramdac.
Display should always be set to pfb0a when running SunVTS.

Note – Disable all screen savers before testing any graphics device. Type **`xset s off`** at a UNIX prompt to disable the Solaris screen saver. Type **`xset -dpms`** (to turn off power management) or type **`xset s noblank`** (to turn off screen saver). Disable Power Management software if it is running.

Note – To start SunVTS with `vtsui`, but without `vtsk`, you must add the host name to `xhost` as: **`xhost +`** *hostname.*

For full instructions on testing frame buffers, refer to the Testing Frame Buffers section of the *SunVTS 5.1 Test Reference Manual*.

`pfbtest` Options

To reach the dialog box below, right-click on the test name in the System Map and select Test Parameter Options. If you do not see this test in the System Map, you might need to expand the collapsed groups, or your system may not include the device appropriate to this test. Refer to the *SunVTS User's Guide* for more details.

By default, all options are enabled except frame buffer locking.

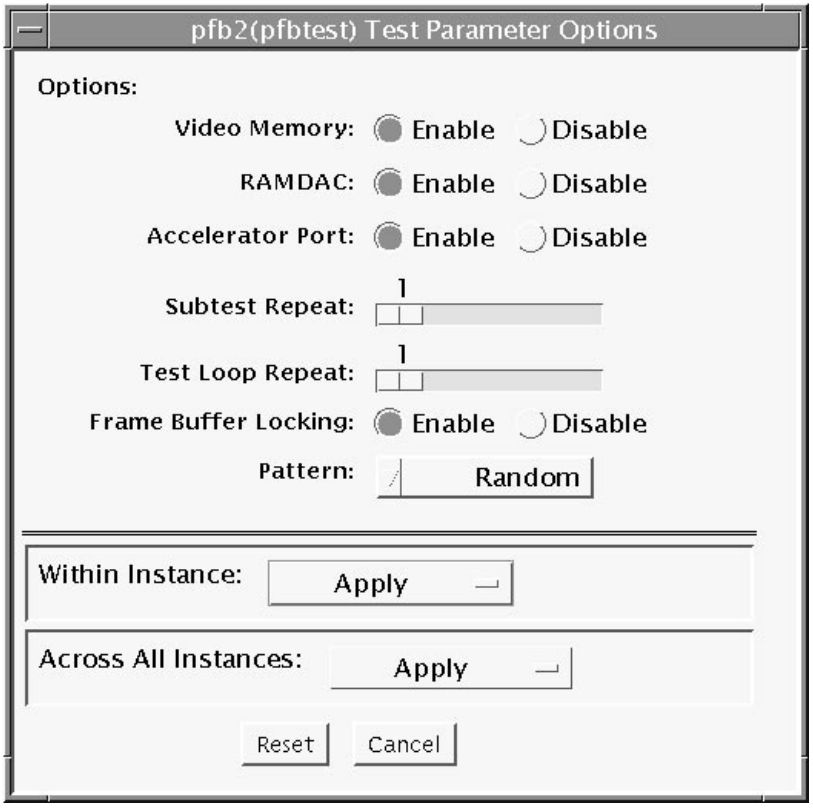

FIGURE 8-1 `pfbtest` Test Parameter Options Dialog Box

TABLE 8-1 `pfbtest` Options

`pfbtest` **Options**	**Description**
Video Memory test	Thoroughly tests the on-screen video memory (the memory part that is mapped on to the monitor) of the Sun XVR-100 graphics accelerator in 8-bit, 16-bit, 32-bit, 64-bit, and 64 byte (block) modes. Entire on-screen video memory is tested by testing 512 bit blocks at a time (8x8 pixel block). Each block is tested in two passes. Each pass consists of a data write and read. In the first pass, user specified data or random data is used, and in the second pass, one's complement of the data used in the first pass is used so that each on-screen video memory location (bit) is tested with a zero (electrical low state) and one (electrical high state).
RAMDAC test	Tests the RAMDAC in three phases. In the first phase the RAMDAC CLUT (Color LookUp Table) is tested using simple write/read patterns to determine if there are any bad bits in CLUT. The data patterns used are: • Random data • Complement of the random data (used as first data pattern) • The data pattern 0101 • The data pattern 10101 In the second phase, four different patterns are drawn on the screen. Each pattern stays on the screen for approximately 1/4 second. The four patterns are listed below. For each pattern, the signature is captured and compared with the signature obtained for the same pattern on a known good board. This test verifies that all the different data paths within the RAMDAC are functioning properly. Patterns drawn on screen: • Red ramp with cursor at top-left corner of the screen • Blue ramp with cursor at top-right corner of the screen • Green ramp with cursor at bottom-left of the screen • Grey ramp with cursor at bottom-right of the screen In the last (third) phase of the RAMDAC test the Vertical Retrace Interrupt is tested for approximately five seconds.
Accelerator Port test	Tests all of the following: • Data paths (sources: fixed color, host data, blit, fixed pattern) • Arithmetic and logic unit (ALU) • Primitives (destinations: line, rectangle) • Mono to color expansion logic Primitives are drawn using a combination of different data paths (allowed), ALU functions, and color comparator functions. A checksum is generated for each data combination and is compared with the checksum generated for the same data combination on a known good board.

TABLE 8-1 `pfbtest` Options *(Continued)*

`pfbtest` **Options**	**Description**
Frame Buffer Locking	This option is set to *disable* if the Sun XVR-100 graphics accelerator is not the console device. When the SunVTS GUI is brought up, Frame Buffer Locking is enabled by default if the Sun XVR-100 graphics accelerator is the console device. If the Sun XVR-100 graphics accelerator is not the console device, Frame Buffer Locking is disabled by default.

`pfbtest` Test Modes

Due to the nature of graphics tests, reading from or writing to the frame buffer during graphics tests will disturb user operation. This test is only available in the Functional test mode.

TABLE 8-2 `pfbtest` Supported Test Modes

Test Mode	**Description**
Functional	The `pfbtest` verifies the proper functioning of Sun XVR-100 graphics accelerator.

pfbtest Command-Line Syntax

/opt/SUNWvts/bin/pfbtest *standard_arguments* **-o dev=***device_name***, S=***subtest_number*,**F=**#*_of_subtest_loops*,**B=**#*_of_test_loops*,L=***disable*,**P=***test_pattern*

TABLE 8-3 pfbtest Command-Line Syntax

Argument	Description
dev=*device_name*	*device_name* is the relative path name of the device being tested with respect to /dev/fbs. The default is pfb0.
S=*subtest_number*	*subtest_number* is the test number of the subtest to be run. Select from the subtests below. You can run multiple subtests by adding the subtest numbers. For example, n=0x3 runs both test 1 and test 2; n=0x5 runs both test 1 and test 4. • n 0x1 VRAM • n 0x2 RAMDAC • n 0x4 Accelerator port test (Rendering Pipeline) More than one test can be selected by ORing subtest numbers. For example: n = 0x5 indicates VRAM and Rendering Pipeline tests. A hex number must be preceded by 0x, decimal numbers are also acceptable.
F=#*_of_subtest_loops*	Specifies the number of times to repeat each subtest. The default is 1.
B=#*_of_test_loops*	Specifies the number of times to repeat a test loop before passing; the default is 1.
L=*disable*	Disables the frame buffer lock. Disable the lock when the Sun XVR-100 graphics accelerator is not the console or when the server is not running on the Sun XVR-100 graphics accelerator under test.
P=*test_pattern*	Specifies the test pattern number. The default is r, for random patterns. You may also choose 0 for 0x0000000, 3 for 0x3333333, 5 for 0x5555555, or 9 for 0x9999999.

Note – 64-bit tests are located in the sparcv9 subdirectory /opt/SUNWvts/bin/sparcv9/*testname*, or the relative path to which you installed SunVTS. If a test is not present in this directory, then it may only be available as a 32-bit test. For more information refer to the "32-Bit and 64-Bit Tests" section of the *SunVTS 5.1 Test Reference Manual* (816-5145-10).

Note – Errors returned by `pfbtest` are nonspecific. It is not possible to determine which component caused a failure. In all error conditions, the field replaceable unit (FRU) is the entire Sun XVR-100 graphics accelerator.

CHAPTER 9

RAM Test (`ramtest`)

`ramtest` is designed to stress the memory modules (RAM) instead of the whole memory subsystem. The test is optimized to achieve large memory bandwidth on UltraSPARC III (USIII) and UltraSPARC II (USII) class of CPUs. `ramtest` has an integrated ECC error monitor which reports the ECC errors found during the test run.

This test is being added only for the Exclusive mode testing because of the high stress it puts on the memory and the system interconnect. `ramtest` assumes that no other application is running at the same time.

Note – `disktest` is supported on x86 platforms that use the Solaris Operating System.

Caution – This is an Exclusive mode test. No other application should be running during this test.

`ramtest` Options

To reach the dialog box below, right-click on the test name in the System Map and select Test Parameter Options. If you do not see this test in the System Map, you might need to expand the collapsed groups. Refer to the *SunVTS User's Guide* for more details.

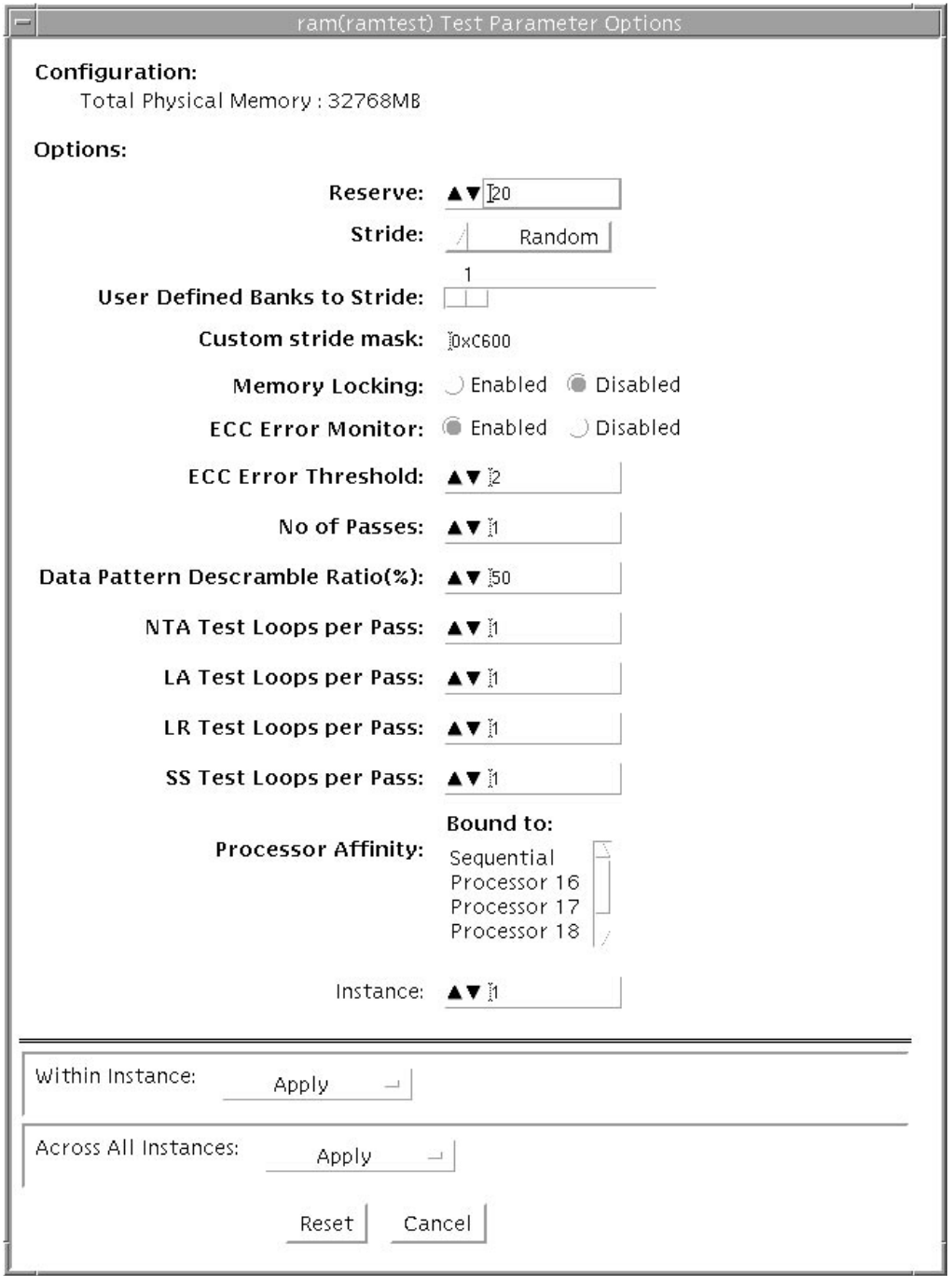

FIGURE 9-1 `ramtest` Test Parameter Options Dialog Box

The following table details the `ramtest` options:

TABLE 9-1 `ramtest` Options

`ramtest` **Options**	**Description**
Reserve	Reserve option represents the percentage of physical memory that is assumed to be in use by the OS or other processes. If you see excessive swapping while running ramtest, increase this percentage. The default is 20%; this means that ramtest allocates 80% of physical memory size for testing. Swapping decreases stress on memory and increases it on the system itself. For memory testing purposes, it is recommended to minimize swapping by tuning the reserve option. If for some reason the allocation or locking (in case Memory Locking is enabled) does not succeed, the amount of memory is reduced and the allocation process is repeated. Once the allocation succeeds, the amount of memory allocated is displayed in the messages.
Stride	By default this option is set to "Random". It can be set to "Column" or "Row" also. In case of random, either Row or Column are randomly selected for each pass. Value of stride defines the memory locations addressed consecutively in certain subtests, in a hardware dependent manner. All testable memory is still tested. Using different strides, checks coupling among different sets of memory cells; therefore random is the recommended value for this option unless both Column and Row are being explicitly used in different instances. For FA type of uses, stride may also be set to "UserDefined", in this case the test will stride the number of banks specified in the "userstride" option. Stride may be set to "Custom" in which case the stride values are randomly selected from the strides specified in the "stridemask" value. Stride may be set to “Custom” in which case the stride values are randomly selected from the strides specified in the “stridemask” value.
User-Defined Banks to Stride	Use this option to set the number of banks that the test should stride. One recommended choice is the interleave on the suspect bank, during FA. The value is currently limited to between 1 and 16. (This also means row striding is not possible while using this option).

TABLE 9-1 `ramtest` Options *(Continued)*

`ramtest` **Options**	**Description**
Stridemask	When stride=custom is selected, this value specifies the strides used. Each thread selects one of the stride values from stridemask by selecting one of the bits in the mask. The bits in the stridemask value represent the Least Significant Bit of the stride. Thus a value of 0x4000 calls for a stride of 16384 (using Bit 14 of the address). Mulitple bits can be set mixing row and column strides. Consult the Memory Controller section of the PRM for the CPU of the test system to discover how the memory reference address is divided between rows and columns in the DRAM. The value can be specified as a Decimal (NNN), Hexadecimal (0xNNN), or Octal (0NNN) value. The maximum value is 0x400000 (4194304). The default value is 0xC600 which represents strides using Bits 15, 14, 10, and 9.
Memory Locking	By default memory locking is "Disabled". To turn it on, set lock to "Enabled". This test uses ISM to lock the memory into the core, this gives 4 MB virtual pages and avoids swapping. Running without locking on the other hand, adds more randomness to the addressing sequence.
ECC Error Monitor	ECC Monitor is "Enabled" by default. The ECC error monitor runs as a separate thread in the test. When an ECC error is detected, the message is displayed on to the test output. The monitor can be turned off by setting this option to "Disabled". The ECC Monitor option is not supported on x86 platforms and an appropriate warning is displayed and the test proceeds based on other options.
ECC Error Threshold	This is the number of ECC errors after which the test will stop (if ECC monitor is running). When the threshold is reached, the test will exit with a non zero exit code. If set to zero, the test will still report all the errors but will not stop. The default of threshold is 2. The ECC Threshold option is not supported on x86 platforms and an appropriate warning is displayed and the test proceeds based on other options.
Number of Passes	This option specifies the number of passes, in the same instance. Increasing passes is recommended in case "lock" is enabled, this will save time spent on locking the memory every time a new process/instance is spawned by the VTS kernel. Note that this pass has no relation with the system passes in the VTS infrastructure, it will appear that `ramtest` is taking longer to complete system passes.

TABLE 9-1 `ramtest` Options *(Continued)*

`ramtest` **Options**	**Description**
NTA March Test	Specifies number of loops of NTA march(30N) test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. NTA march test attacks coupling and stuck at faults. NTA march is efficient at finding single, double, and some triple bit errors. Depending on the stride option, coupling faults between cells in adjacent columns, or rows that are targeted. Note that test time will be higher when row striding is selected because of greater page faults generated. For efficiency purposes, total memory is divided among available CPUs.
LA March Test	Specifies number of loops of LA march(22N) test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. LA march test attacks coupling and stuck-at-faults.
LR March Test	Specifies number of loops of LR march(14N) test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. LR march test attacks coupling and stuck-at-faults.
SS March Test	Specifies number of loops of SS march(22N) test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. SS march test attacks simple static faults. The SS March option is not supported on x86 platforms and an appropriate warning is displayed and the test proceeds based on other options.

ramtest Test Modes

TABLE 9-2 ramtest Supported Test Modes

Test Mode	Description
Exclusive	Generates enormous amount of memory traffic.

ramtest Command-Line Syntax

/opt/SUNWvts/bin/sparcv9/ramtest *standard_arguments* **[-o**

[reserve=<*Integer between 0 and 90*> **]**
[stride=<*Row* | *Column* | *Random* | *UserDefined* | *Custom*>
[userstride=<*1 - 16*> **]**
[stridemask=<*0x40 - 0x400000*>]
[lock=<*Enabled* | *Disabled*> **]**
[dratio=<*Integer between 0 and 100*> **]**
[eccmonitor=<*Enabled* | *Disabled*> **]**
[threshold=<*Integer i; 0 <= i <= MAX_INT* >**]**
[pass=<*32 bit integer*> **]**
[ntaloops=<*32 bit integer*>**]**
[laloops=<*32 bit Integer*> **]**
[lrloops=<*32 bit Integer*> **]**
[ssloops=<*32 bit Integer*>]]

TABLE 9-3 `ramtest` Command-Line Syntax

Argument	Description
`reserve`	This is used to specify the amount of memory that will not be allocated for testing. Reserve represents a percentage of the total physical memory in the system. When the test starts, it probes the total memory present in the system, then tries to allocate (100 - reserve)% of memory. If the allocation or locking does not succeed the amount of memory is reduced before the retry. Before starting the test, the amount of memory allocated for testing is displayed. Default value for reserve option is 20. For US IIIi platforms, default value is tuned to 25. It should be noted that on low memory systems, the reserve value should be kept higher to avoid excessive swapping. For 32-bit booted systems, the reserve value represents the percentage of 4 GB rather than the percentage of total physical memory.
`stride`	By default stride is set to "Random". It can be set to "Column" or "Row" also. In case of random, either Row or Column are randomly selected for each pass. Value of stride defines the memory locations addressed consecutively in certain subtests, in a hardware dependent manner. All testable memory is still tested. Using different stride checks coupling among a different set of memory cells, therefore random is the recommended value for this option unless both column and row are being explicitly used in different instances. For FA type of uses, stride may also be set to "UserDefined", in this case the test will stride the number of banks specified in the "userstride" option. Stride may be set to "Custom" in which case the stride values are randomly selected from the strides specified in the "stridemask" value.
`userstride`	Use this option to set number of banks the test should stride. One of the good choices could be the interleave on the suspect bank, during FA. the value is limited between 1 and 16 right now. (This also means row striding is not possible while using this option).
`stridemask`	When stride=custom is selected, this value specifies the strides used. Each thread selects one of the stride values from stridemask by selecting one of the bits in the mask. The bits in the stridemask value represent the Least Significant Bit of the stride. Thus a value of 0x4000 calls for a stride of 16384 (using Bit 14 of the address). Mulitple bits can be set mixing row and column strides. Consult the Memory Controller section of the PRM for the CPU of the test system to discover how the memory reference address is divided between rows and columns in the DRAM. The value can be specified as a Decimal (NNN), Hexadecimal (0xNNN), or Octal (0NNN) value. The maximum value is 0x400000 (4194304). The default value is 0xC600 which represents strides using Bits 15, 14, 10, and 9.

TABLE 9-3 `ramtest` Command-Line Syntax

Argument	Description

`lock`	By default memory locking is "Disabled". To turn it on set lock to "Enabled". The test uses ISM to lock the memory into the core, this gives 4 MB virtual pages and avoids swapping. Running without locking on the other hand, adds more randomness to the addressing sequence.
	It should be noted that on low memory systems, this option can be "Enabled" to avoid excessive swapping.
	In case the test is unable to lock the memory, the user should put the following lines in `/etc/system` and reboot the machine.
	`set shmsys:shminfo_shmmax=0xFFFFFFFFFFFFFFFF`
	`set shmsys:shminfo_shmmin=1`
	`set shmsys:shminfo_shmmni=100`
	`set shmsys:shminfo_shmseg=10`

`eccmonitor`	ECC Monitor is "Enabled" by default. The ECC error monitor runs as a separate thread in the test. When an ECC error is detected, the message is displayed on to the test output. The monitor can be turned off by setting this option to "Disabled". The ECC Monitor option is not supported on x86 platforms and an appropriate warning is displayed and the test proceeds based on other options.
`threshold`	This is the number of ECC errors after which the test will stop (if ECC monitor is running). When the threshold is reached the test will exit with a non zero exit code. If set to zero, the test will still report all the errors but will not stop. The default threshold is 2. The ECC Threshold option is not supported on x86 platforms and an appropriate warning is displayed and the test proceeds based on other options.
`pass`	This option specifies number of passes, in the same instance. Increasing pass is recommended in case "lock" is enabled, this will save time spent on locking the memory every time a new process/instance is spawned by the VTS kernel. Note that this pass has no relation with the system passes in the VTS infrastructure, it will appear that `ramtest` is taking longer to complete system passes.
`ntaloops`	Specifies number of loops of NTA march(30N) test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. NTA march test attacks stuck-at-faults, two cell coupling faults, and some three cell coupling faults.
`laloops`	Specifies number of loops of LA march(22N) test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. LA march test attacks coupling and stuck-at-faults.

TABLE 9-3 `ramtest` Command-Line Syntax

Argument	Description
`ntaloops`	Specifies number of loops of NTA march test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. NTA march test attacks coupling and stuck at faults.
`lrloops`	Specifies number of loops of LR march(14N) test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. LR march test attacks coupling and stuck-at-faults.
`dratio`	Descrambles ratio can be used to tune the algorithm used to generate data patterns in `ramtest`. Descramble ratio of 100 means that all the data patterns generated will be descrambled. Where as if descramble ratio is 0, the test will generate the data patterns tuned towards bus noise. Default value is 50, which means that half the data patterns are descrambled.
`ssloops`	Specifies number of loops of SS march(22N) test, per pass. Increasing the number of loops of any subtest increases the relative time spent on that subtest in each pass. This increase also increases the time taken to complete a pass. The SS March test attacks simple static faults. The SS March option is not supported on x86 platforms and an appropriate warning is displayed and the test proceeds based on other options.
`custom`	When `stride=custom` is selected, this value specifies the strides used. Each thread selects one of the stride values from stridemask by selecting one of the bits in the mask. The bits in the stridemask value represent the Least Significant Bit of the stride. Thus a value of 0x4000 calls for a stride of 16384 (using Bit 14 of the address). Mulitple bits can be set mixing row and column strides. Consult the Memory Controller section of the PRM for the CPU of the test system to discover how the memory reference address is divided between rows and columns in the DRAM. The value can be specified as a Decimal (NNN), Hexadecimal (0xNNN), or Octal (0NNN) value. The maximum value is 0x400000 (4194304). The default value is 0xC600 which represents strides using Bits 15, 14, 10, and 9.

Note – 32-bit tests are located in the bin subdirectory, `/opt/SUNWvts/bin/`*testname*.

Note – ECC errors returned by `ramtest` are actually detected by the operating system and are logged in the `/var/adm/messages` file. Please review this file for more detailed information regarding errors.

Note – 64-bit tests are located in the `/bin/64` directory, or the relative path in which you installed SunVTS. If a test is not present in this directory, then it might be available as a 32-bit test only. For more information, see “” on page 5.

CHAPTER 10

Sun™ XVR-4000 Graphics Accelerator Test (`zulutest`)

The `zulutest` does functional testing of the Sun™ XVR-4000 graphics accelerator device. zulutest detects and adapts to the video modes of Sun XVR-4000. All zulutest tests can be performed in several screen resolutions such as standard, Stereo, and high resolution. In Stereo mode, all tests write into the right and left eyes unless you specify otherwise. Use the fbconfig -dev device-name -prconf command to display the configuration of the frame buffer you want to test.

You can interrupt zulutest using Control-C. Turn off all other keyboard input if the CDE user interface is running on the unit being tested. Test accuracy is checked using a checksum algorithm. Possible locations of failing pixels are identified, in addition to the failing FRU.

`zulutest` is only available in 64-bit mode.

Caution – Do not run any 3D graphics applications screen lock or screen saver programs that uses the Sun XVR-4000 graphics accelerator port while running `zulutest`. This combination causes SunVTS to return incorrect errors.

`zulutest` Test Requirements

Disable all screen locks and screen savers before testing any graphics device. Type **`xset s off`** at a UNIX® prompt to disable the Solaris screen saver. Disable the Power Management software if it is running.

For full instructions on testing frame buffers, please refer to the Testing Frame Buffers of the *SunVTS 5.1 Test Reference Manual*.

To start SunVTS with the vtsui, and without the vtsk, you must add the host name to xhost as follows: xhost + host_name

Using `zulutest` Without X-Windows

If you perform `zulutest` on a system that was powered on without running X-Windows, you must bring up X-Windows on the Sun XVR-4000 Graphics Accelerator device under test and kill the X-Windows process before performing `zulutest`. Otherwise, the Convolve subtest will fail, and other subtests may also fail.

Note – You must enable multisampling with the `fbconfig` command before performing the following workaround. To perform `zulutest` with X-Windows (CDE) the following workaround is not necessary.

Workaround

To bring up X-Windows on the Sun XVR-4000 Graphics Accelerator device under test, enter the following command:

```
/usr/openwin/bin/Xsun -dev /dev/fbs/device_name &
```

It takes 30 to 45 seconds before Xsun comes up. To kill the Xsun process, enter the following command:

```
pkill -KILL Xsun
```

Once the Xsun process is killed, the `zulutest` can be performed without the incorrect subtest errors.

The Sun XVR-4000 Graphics Accelerator cannot perform video read back in Interlaced and Stereo modes because the Convolve subtest cannot keep up.

For `zulutest` to be able to perform the Convolve subtest, multisampling must be enabled.

`zulutest` Options

To reach the dialog box below, right-click on the test name in the System Map and select Test Parameter Options. If you do not see this test in the System Map, you might need to expand the collapsed groups, or your system may not include the device appropriate to this test. Refer to the *SunVTS User's Guide* for more details.

By default, all `zulutest` options are enabled except for the Stereo test.

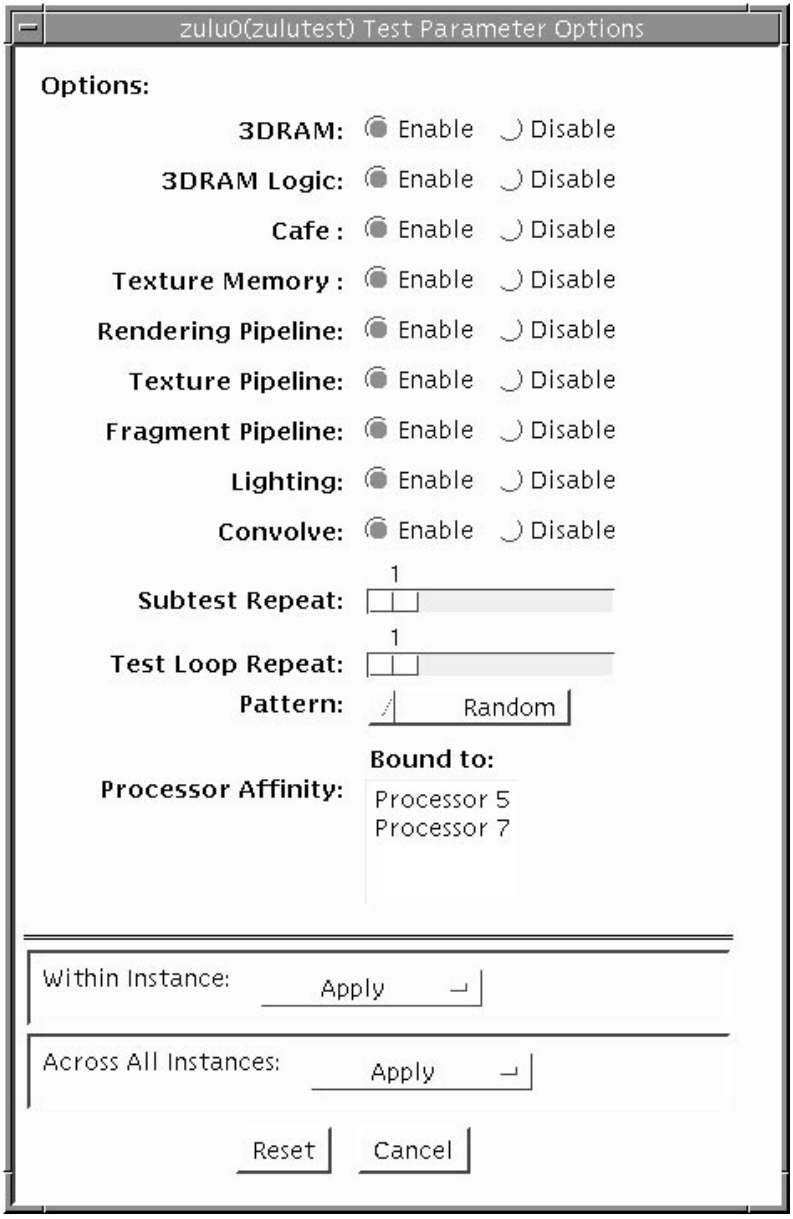

FIGURE 10-1 `zulutest` Test Parameter Options Dialog Box

TABLE 10-1 `zulutest` Options

`zulutest` **Options**	**Description**
3DRAM test	The 3DRAM test thoroughly tests the video memory in the Sun XVR-4000 graphics accelerator using 512-bit reads and writes. 3DRAM makes a full-screen pass, consisting of a write and a read to each pixel location, for each access mode on the list below. You can use either random data or specify data at the command line. A second pass is made with the one's complement of the data used in the first pass so that each memory location is tested with both a zero and a one. Errors in this subtest are attributed to the 3DRAM. A failing chip is indicated by (X, Y) locations and device-specific "U" numbers in the following access modes: • SFB Stencil 8 • SFB WID 16 • FB RGBAZ 64 - Buffer A • SFB RGBAZ 64 - Buffer B
3DRAM Logic test	The 3DRAM Logic test provides logical functionality to the Sun XVR-4000 graphics accelerator. The following services are tested: • Compare Controls—Match AB • Compare Controls—Magnitude AB • Compare Controls—Match C • Compare Controls—Magnitude C • Match Mask—AB • Magnitude Mask—AB • Match Mask—C • Magnitude Mask—C • Raster Operations—RGB • Raster Operations—X • Raster Operations—YZ • Plane Mask—RGB Each function is tested separately with a series of SFB64 writes. A total of 16 writes are made for each different test case with Y coordinate values varying from 0 to 30 in increments of 2 pixels. This dotted column organization provides page thrashing and block flashing in all screen resolutions. For each operation, all possible combinations are tested. For example, in `ROP RGB` *new==old* there are three possible values: *new < old, new == old, and new > old*. Each of these cases are tested. Errors in this subtest are attributed to the 3DRAM.

TABLE 10-1 `zulutest` Options *(Continued)*

`zulutest` **Options**	**Description**
Cafe test	This test will do non-destructive testing of the Cafe memory (RDRAM) and Cafe. Errors in this test are attributed to the Cafe and its memory.
Texture Memory test	Texture Memory test tests out all the of the Texture Memory by writing the data pattern selected (random, 0s, 1s, 5s, or 0xAs). By default, Random data is selected. The data is written using block writes and read back using block reads. Errors in this test are attributed to the Texture Memory and the Texture Memory subsystem.
Rendering Pipeline test	Each primitive is tested thoroughly by exercising the following: • Simple Triangles • 2d primitives • 3d Primitives (such as Triangles, 3d lines etc.) • Vertex Processor Errors in this test are attributed to the pipelines of the Sun XVR-4000 graphics accelerator and/or 3DRAM.
Texture Pipeline test	This test renders textured primitives to test: • 2d texture Minification filtering • 2d texture Magnification filtering • 3d texture Minification filtering • 3d texture Magnification filtering • Texture environment • Filter4 and sharpen filters • Anisotropic filter Errors in this test are attributed to the pipelines of the Sun XVR-4000 graphics accelerator and/or 3DRAM.
Fragment Processor test	Fragment Processor test, a subtest, exercises the fragment pipe of each pipeline of the XVR-4000's. Auxiliary clipping (additive and subtractive): • Depth cueing • Alpha blend • Viewport clip (2D and 3D) • Area pattern (transparent and opaque) Errors in this test are attributed to the FBC3 and/or 3DRAM.

TABLE 10-1 `zulutest` Options *(Continued)*

`zulutest` **Options**	**Description**
Lighting test	The Lighting test exercises Cafe and lighting microcode. This test lights an object with maximum number of lights that XVR-4000 can handle in hardware. A checksum is generated for the rendered image and compared with the checksum generated for the same image on a known good system. Errors in this test are attributed to the Cafe, Microcode, FBC3 and RD RAMs.
Convolve test	Convolve test tests the Convolve chips functionality (convolution filters, Color look up tables and Gamma look up tables) along with the video read back functionality of convolves and master chip. This sub test renders an image which is made up of lines drawn radial. Then a block in the center of the image is super sampled and video read back is initiated. Once the video read back data is available to the zulutest, zulutest will generate checksum and compares with the checksum generated on known good system. Errors in this subtest can be attributed to FBC3, 3DRAM, Convolve, Master.
Stereo test	Currently, this sub test is not active. Stereo test displays an object in Stereo mode with different images for the right and left eye. You can verify proper operation by looking at the screen with stereo glasses and following the instructions displayed in the Parameter Options dialog box. This test temporarily switches the monitor into Stereo mode, renders a Stereo image and after displaying the image for five seconds, restores the monitor to its previous resolution.

`zulutest` Test Modes

Due to the nature of graphic tests, reading data from, or writing data to the frame buffer during graphic tests will disturb user operation. For this reason, `zulutest` is only available in Functional test mode.

TABLE 10-2 `zulutest` Supported Test Modes

Test Mode	**Description**
Functional	Runs the full set of tests.

zulutest Command-Line Syntax

/opt/SUNWvts/bin/sparcv9/zulutest *standard_arguments* **-o dev=***device_name*,**S=***subtest_number*,**F=**#*_of_subtest_loops*,**B=**#*_of_test_loops*,**P=***test_pattern*

TABLE 10-3 zulutest Command-Line Syntax

Argument	Description
dev=*device_name*	*device_name* is the relative path name of the device being tested with respect to /dev/fbs; the default is zulu0.
S=*subtest_number*	*subtest_number* is the test number of the subtest to be run. Select from the subtests below. You can run multiple subtests by adding the subtest numbers together. For example, *n=0x3* runs both test 1 and test 2; *n=0x180* runs both test *0x080* and test *0x0100*. You do not need the leading zeros. • *n*—0x00001 Video Memory 3DRAM • *n*—0x00002 3DRAM Logic • *n*—0x00004 Cafe • *n*—0x00008 Texture Memory SDRAM • *n*—0x00010 Rendering Pipeline • *n*—0x00020 Texturing Pipeline • *n*—0x00040 Fragment Pipeline • *n*—0x00080 Lighting • *n*—0x00100 Convolve • *n*—0x00200 Stereo More than one subtest can be selected by ORing their subtest numbers. Example: n = 0x00011 indicates 3DRAM and Rendering Pipeline tests. A hex number must start with 0x, decimal numbers are also acceptable. [n = 0xff] If looping on a test, the verbose mode is disabled. F=n : Number of times to repeat each subtest [n = 1]. B=n : Number of times to repeat test loop before passing [n = 1]. P=*pattern* : test pattern – r for random, 0 for 0x00000000, 3 for 0x33333333, 5 for 0x55555555, or 9 for 0x9999999. [*pattern=r*]
F=#*_of_subtest_loops*	The number of times to repeat each subtest. The default is 1.
B=#*_of_test_loops*	The number of times to repeat a test loop before passing. The default is 1.
P=*test_pattern*	The test pattern number. The default is **r**, for random patterns. You may also choose 0 for 0x0000000, 3 for 0x3333333, 5 for 0x5555555, or 9 for 0x9999999.

Note – 64-bit tests are located in the `sparcv9` subdirectory `/opt/SUNWvts/bin/sparcv9/`*testname, or the relative path to which you installed SunVTS*. If the test is not present in this directory, then it may only be available as a 32-bit test. For more information refer to the "32-Bit and 64-Bit Tests" section of the *SunVTS 5.1 Test Reference Manual* (816-5145-10).

Note – Errors returned by `zulutest` are nonspecific: It is not possible to determine which component caused a failure. In all error conditions, the field replaceable unit (FRU) is the entire Sun XVR-4000 graphics accelerator.