sun microsystems

PROM User's Manual Addenda and Errata

Sun Microsystems, Inc. • 2550 Garcia Avenue • Mountain View, CA 94043 • 415-960-1300

Part No: 800-1736-10
Revision A of 24 April 1989

Contents

Tables

Figures

PROM User's Manual Addenda and Errata

The *PROM User's Manual* describes PROM monitor commands, self-tests and extended tests for Sun-2 through Sun-386i systems. This text adds to or modifies the information found there, for Sun-3 systems with the new, 2.8 version of the Boot PROM, and all Sun-3/400 series, the Sun-3/80 and SPARCsystem 330s.

New Monitor Command Feature

In the past, all numerical PROM monitor commands were entered in hexadecimal. If you have PROM version 2.8, or a Sun-3/400 series system, a Sun-3/80 or SPARCsystem 330, you may now enter decimal or ASCII values after the PROM monitor prompt (>). This feature is particularly useful when using the monitor `q` command to program the EEPROM, which sometimes requires that you convert letters and decimal numbers to hexadecimal values before you enter them.

To enter a decimal value after a PROM monitor command, simply precede the value with the "%" character:

```
>q 050 %20
```

To enter an ASCII character, simply precede it with the "@" character:

```
>q 022 @i
>q 023 @e
```

If the value you enter is not preceded by a % or @ character, the monitor program treats that the value as hexadecimal.

EEPROM Security Feature

Chapter 10 of the *PROM User's Manual* describes the Sun-3 and Sun-4 EEPROM. There is now a Security Mode Select Feature, located at EEPROM address 0x492. This feature provides a *non-secure* mode that permits the use of all PROM monitor commands.

It also provides a *command secure* mode that permits the use of PROM monitor commands (other than `c`, for continue, or `b`, for boot, without parameters) only when a password is entered. In the command secure mode, you may operate your workstation normally, including powering down, booting, terminating with the `L1-A` command, and re-booting. You may not, however, perform any unusual operations, such as booting a non-standard kernel, running diagnostics, or changing EEPROM or CPU board memory contents, without entering a password.

Finally, this feature provides a *fully secure* mode that does not permit use of the PROM monitor (other than the **c** command with no parameters) without entering a password. To power-up, re-boot, or perform any other PROM monitor operation, you must supply a password. This mode allows you to control access to the workstation by turning it off. The workstation does not automatically boot on power-up.

CAUTION **In fully secure mode, even a default boot cannot be completed unless the password is entered. Once the SunOS is halted, you cannot restore it until you enter the correct password after the prompt. If the password is unknown, the system CPU board must be serviced as a failed board.**

The Password

If you should attempt to enter a PROM monitor command such as q, for example, on a system that is set for one of the secure modes, your interaction might look like this:

```
>q 020                          (you want to look at EEPROM offset address 020)
>Mon Pass:                   (you enter the correct password)
>EEPROM 020: 00 ?        (the contents of location 020 are shown)
```

Or, if you enter an incorrect password, your interaction might look like this:

```
>q 020
>Mon Pass: (you enter the wrong password)
>Invalid
                (there is a slight delay)
>
```

You may now try again or enter an unprotected command.

To install or change a password, the system must be in non-secure mode, or you must know the existing password for a secure system. You then use the PROM monitor **q** command to enter the password in EEPROM offset location 490 for Sun-3 and Sun-4 (SPARC) systems, or 160 for the Sun386i.

NOTE *When you attempt to change the values stored in the EEPROM monitor password locations, you will be prompted with this message:*

```
Modifiying security location(s). Are you sure?(y/N)
```

If you enter **y** *for yes, the change you entered is written to the location shown. If you enter* **n** *for no, nothing is written to EEPROM, and the contents of the next location are displayed.*

To enter a monitor password, use the @ character, described at the beginning of this document, in order to enter the letters that make up the password. If you do not use the @ character, you must enter the hexadecimal equivalent of the letters. Following is an example of the way you would enter a password called *mypasswd* on a Sun-3 or Sun-4 system. You enter one letter per location, followed with : Return. To exit the command, you may enter any non-hexadecimal character, such as period, as shown.

```
>q 493
> EEPROM 493: 00? @m
Modifiying security location(s). Are you sure?(y/N) Yes
> EEPROM 494: 00? @y
Modifiying security location(s). Are you sure?(y/N) Yes
> EEPROM 495: 00? @p
Modifiying security location(s). Are you sure?(y/N) Yes
> EEPROM 496: 00? @a
Modifiying security location(s). Are you sure?(y/N) Yes
> EEPROM 497: 00? @s
Modifiying security location(s). Are you sure?(y/N) Yes
> EEPROM 498: 00? @s
Modifiying security location(s). Are you sure?(y/N) Yes
> EEPROM 499: 00? @w
Modifiying security location(s). Are you sure?(y/N) Yes
> EEPROM 49a: 00? @d
Modifiying security location(s). Are you sure?(y/N) Yes
> EEPROM 49b: 00? .
>
```

NOTE *If a password was already stored in locations 493-49a, hexadecimal values would appear in place of the zeroes in the example above.*

The password you enter must either fill the eight bytes (locations 493-49a) with a character or a zero.

Note that changes to the security mode and password do not take effect until the PROM monitor mode is re-entered.

It is recommended that the password is changed before the security mode is changed. For more information on using the EEPROM **q** command, refer to the *PROM User's Manual*, Sun PN 800-1736, or the `Monitor(8S)` section of the *SunOS Reference Manual.*

Changing Security Modes

The EEPROM offset location 492 (or 162 for a Sun386i) contains a value that determines the security mode. The table below shows the interpretation of values found in that location. The "0x" denotes a hexadecimal value.

0x1	command secure
0x5e	fully secure
all other values	non-secure

Because the PROM monitor password is stored as text, it is recommended that the `chmod` and `/etc/chown` commands be used so that the `/dev/eeprom` device file may be accessed by the super-user. To accomplish this, enter:

```
%su
Password: enter your super-user password
# cd /dev
# /etc/chown root eeprom
# chmod 600 eeprom
```

EEPROM Layout for PROM Security

Here is a table that shows the EEPROM offset locations for the Sun-3, Sun-4 and Sun386i systems.

Sun386i Offset	*Sun-3,-4 Offset*	*Field*	*Function*
0x160-1	0x490-1	bad_login	The bad login counter stores the number of invalid password attempts. The maximum value is 65535 and the counter does not roll over to 0.
0x162	0x492	secure	The values 1 and 0x5e correspond to command secure and fully secure, respectively. Any other value is non-secure.
0x163-a	0x493-a	password	If the password is shorter than 8 bytes, the password string is fenced with a null character.

3-D Logo EEPROM Parameter

If your system has a CG6 board, you may set EEPROM location 0x020 to 0x06 so that, upon power-up, the Sun logo appears to be three-dimensional. Refer to the *PROM User's Manual* monitor **q** command description for information on changing EEPROM values.

Sun-3 Extended Tests

Chapter 9 of the *PROM User's Manual* describes Sun-3 extended tests. For workstations with the boot PROM version 2.8, many of these tests are unavailable, and the user interface has changed. The new PROM version tests only the devices needed to boot the operating system. Therefore, when you invoke the extended tests from the monitor prompt,

```
>x
```

the menu of extended tests look something like this:

```
Extended Test Menu: (Enter 'q' to exit)

Cmd - Test

ie  -  Intel Ethernet Test
mk  -  Mouse/Keyboard Ports Test
rs  -  Serial Ports Test
```

NOTE *For CPU boards with the AMD AM7990 (Lance) Ethernet chip, the first choice will be*

```
ae  -  AMD Ethernet Test
```

New Boot Path Extended Tests

In order to invoke the disk and tape bootpath tests when a version 2.8 PROM is installed on a Sun-3 CPU board, you must enter an asterisk after the boot command, from the monitor prompt:

```
>b*device()
```

The extended test appropriate for the named *device* will then be executed, and any error messages displayed on the screen. *device* could be one of the following:

sd	*for SCSI disk*
st	*for SCSI tape*
xd	*for Xylogics 7053 Disk Controller*
xt	*for Xylogics tape*
xy	*for Xylogics 450/451 Disk Controller*

Sun-3/400 Series and Sun-3/80 PROM

For Sun-3/400 series and Sun-3/80 users, the text that follows is intended to replace Chapters 7 and 8 in the *PROM User's Manual*, Sun PN 800-1736-10. These chapters cover changes to PROM monitor commands and self-tests associated with the Sun-3/400 product. In addition, the information in Chapter 13 of the PROM manual, "Sun-4 Extended Test System", applies to the Sun-3/400 series workstation. If you have a Sun-3/400 series system, Chapter 9, "Sun-3 Extended Test System" does not apply.

The Power-Up Test Sequence

In order to perform the power-up tests, two assumptions must be met. The CPU (the non-PMMU portion of the MC68030 for the Sun-3/400 series system) must be functional and the ability to fetch instructions from the Boot PROM must be intact.

Powering up a Sun-3 workstation resets the CPU to *boot* state, which means that all instruction fetches are forced to the Boot PROMs. Execution of the minimum-confidence power-up tests begin immediately. These tests do not employ any memory until memory has been successfully checked.

The objective of the power-up test sequence is to determine whether or not the CPU board logic and main memory are functional. Following the successful completion of the power-up tests and subsequent workstation initialization, an attempt is made to boot the SunOS operating system, an EEPROM-specified program, or an operator-specified stand-alone program.

If hardware problems are detected during the verification process, the PROM monitor prompt should appear.

For Sun-3/400 series systems, if the Diagnostic Switch at the rear of the system is in NORM position, you will not be able to interact with the self-tests. You may read the LEDs on the CPU board edge (described later in this chapter) to determine whether or not a test is failing, and you will see a rotating diagonal symbol after the

```
Testing _ megabytes of memory...
```

message on the console during the memory tests. The quantity of memory checked during a power-up with the diagnostic switch on NORM is dependent on EEPROM programming. The EEPROM chapter in the *PROM User's Manual* and the `eeprom` command description in the *SunOS Reference Manual* explain how to set the parameter that controls the quantity of memory tested.

For the Sun-3/80, which has no diagnostic switch, an EEPROM parameter must be set in order to execute a diagnostic boot-up and view the self-test display on a terminal. Refer to the "Diagnostic Power-Up" section for more information on this setting.

If the workstation contains a large amount of main memory, self-tests may last as long as eight minutes. The table below compares the self-test duration of various Sun-3 systems.

Table 1 *Self-Test Execution Time Comparisons*

System Type	*Clock Rate in MHz*	*Memory Size in Megabytes*	*Self-test Duration in Minutes*
Sun-3/160	16 MHz	4 Meg	.5 Min
Sun-3/75	16 MHz	8 Meg	.75 Min
Sun-3/60	20 MHz	24 Meg	1.75 Min
Sun-3/80	16 MHz		11 Sec (est.)
Sun-3/260	25MHz	64 Meg	2.5 Min
Sun-3/400 series	33MHz	128 Meg	2.75 Min (est.)
SPARC system		128 Meg	8.0 Min (est.)

If the Diagnostic Switch is in the NORM position, (or the Sun-3/80 EEPROM parameter set), the power-up tests execute successfully and, if you do *not* terminate the default boot sequence, an attempt is made to down-load the SunOS operating system.

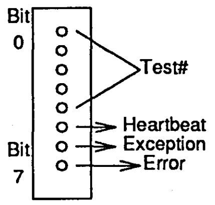
A display something like this appears on the *workstation's* screen to indicate that power-up tests are successful:

```
Selftest Completed Successfully.

       Sun Workstation, Model Sun-3/___ Series
       Type-X keyboard
       ROM Rev __, _ MB memory installed, Serial #_______
       Ethernet address __:__:__:__:__:__

Testing __ megabytes of memory...Completed.

Auto-boot in progress...
```

Diagnostic LEDs

One requirement of Sun-3 firmware is to assign a unique test number to most of the power-up tests and display that number in bits *zero* through *four* of the diagnostic LEDs as the test is running. Given that there are fewer test numbers than there are power-up tests, power-up tests that check the same part of hardware share a test number.

If one of these power-up tests should fail, bit *seven* of the diagnostic LEDs also lights up. Bit seven serves as an indicator that there is a hardware problem. The LED display permits the service person not only to conclude whether or not there is a problem, but to determine which type of power-up test is failing.



For the sake of completeness, LED *five* is the *heart beat* LED. After the power-up tests have been completed, but prior to invocation of the SunOS operating system or an EEPROM-specified program, LED 5 will blink on and off to indicate that the IU is actually executing instructions. LED *six* indicates whether or not the failure is an exception (i.e. unexpected trap or unexpected interrupt). The diagram to the left shows LED designations for Sun-3/400 series systems.

Sun-3/80 LED

Sun-3/80 systems have one green LED that blinks while a test is in progress, and turns OFF when there is an error condition. When the light stays ON, everything is functioning satisfactorily:

LED status	Visual	Condition
Blinking	Θ	Testing
OFF	O	Error
ON (Green)	●	Ok

The Boot Sequence

Following the initialization of the workstation, the default boot sequence is executed. There are two issues that must be considered here. One has to do with *what is to be down-loaded* while the other has to do with *where it is to be loaded from.*

Assuming *no* operator intervention, the position of the Diagnostic Switch (on all Sun-3's except the Sun-3/80) will determine *what* is to be booted. If the Diagnostic Switch is in the NORM position, the SunOS operating system is booted. Otherwise, if the Diagnostic Switch is in the DIAG position, the EEPROM-specified program is booted. Be aware that the PROM monitor program is invoked if no EEPROM-specified program is available.

If you are in the PROM monitor mode, you may specify *what* is to be booted and *where* it is to be booted from. See command **b** (boot) in the chapter titled *Sun-3 PROM Monitor Commands* for a description of how to boot user-specified programs from user-specified devices.

If you are interacting with a Sun-3 workstation through the console to reach the monitor program, you may enter `L1-a` (or Break on a terminal) IMMEDIATELY after the

```
Testing __ megabytes of memory...Completed.
```

message.

CAUTION **Do not use `L1-A` procedure once the automatic boot has started; the file systems may be damaged if disks are powered-on and the operating system has started to run.**

If the operating system is already booted, use the procedures described in Chapter 3 of the *PROM User's Manual* to start the monitor.

Once you are in the monitor mode (symbolized by the > prompt), you should do the following:

```
> g 0
panic: zero
Syncing disks... done
```

Press `L1-a` or [Break] again when the message above finishes.

Next you may see this message:

```
dumping to dev somevalue, offset somevalue

Abort at somevalue
>
```

The firmware also determines from what boot device the program will be loaded. If the Diagnostic Switch is in the NORM position and the content of EEPROM location 0x18 is equal to 0x12 (an arbitrarily chosen value), Sun-3 firmware will attempt to boot the SunOS operating system from the boot path specified in the EEPROM, beginning at location 0x19. If the boot path is missing or contains an error, the monitor program is invoked.

If the Diagnostic Switch is in the NORM position and the content of EEPROM location 0x18 is *not* equal to 0x12, Sun-3 firmware attempts to boot the SunOS operating system using the following boot device polling sequence:

1. Xylogics Disk (450-451).
2. SCSI Disk.
3. Ethernet.

If the Diagnostic Switch is in the DIAG position, the firmware assumes that both the path name of the file containing the to-be-loaded program and the boot device are specified in the EEPROM, beginning at EEPROM location 0x22. If either the file name or the boot device is not present or is in error, the monitor is invoked. If the Diagnostic Switch is in the DIAG position, you may connect a terminal to Serial A or B and interact with the self-tests and the Extended Test System, if required.

Diagnostic Power-Up

NOTE *There is no diagnostic switch on a Sun-3/80; an EEPROM setting is required to allow a diagnostic boot-up that displays test names and errors through Serial Port A. Enter the PROM monitor mode and use the* `q` *command to write* **12** *to location 0x70b. Any other value causes tests to run without error reporting to the terminal. Refer to "Displaying and Modifying Memory" for information on use of the* `q` *command.*

If the Diagnostic Switch exists and is on DIAG, the self-test is executed as it is when the switch is "off" or on NORM, except that all of memory is tested. In addition, self-test status information is directed only to serial ports A and B, using the MMU (Memory Management Unit) bypass until all hardware required for the Video Monitor has been successfully tested.

Any hardware failures during the selftests will invoke scope loops to permit troubleshooting the failure. An RS-232 terminal with its characteristics set to 9600 Baud, 8 data bits, 1 stop bit and no parity should be connected to Serial Port A of the CPU board if you wish to view self-test status and interact with the system. If you use Serial Port B you must set the terminal baud rate to 1200.

Limited interaction with the self-test program is possible in this mode and the following characters will invoke the following actions when entered from the terminal connected to serial port A during self test. Each of these commands is documented below.

ESCAPE Key — Sun-3/400 Series and Sun-3/80

Pressing this key any time during the self-test sequence prior to the display of the `Selftest Passed` message causes the self-test sequence to abort and a warning message is displayed. The necessary memory sizing and other setup is done, and then, for the Sun-3/400 series, the program displays the PROM Monitor menu. For the Sun-3/80, this message is displayed:

```
<Warning: selftests aborted by user>

<Initializing Main Memory 0x0000000C Megabytes Initialized>

Type a Character within 10 seconds to enter the Menu Tests...(e for e
mode)
EEPROM: Using RS232 A port.
Selftest Completed.
```

Note that this abort can be done if the test is running normally (without any errors) or if the test is presently looping on any error encountered.

Control-q Key — Sun-3/400 Series Only

Holding down the [Control] key while pressing the `q` key any time during the execution of a particular Sun-3/400 Series self-test causes that one self-test to stop and execution of the next self-test to begin. For example, if the [Control-q] sequence was entered during the `MEMORY WRITE/WRITE/READ TEST`, that test would terminate and the next test (`MEMORY ADDRESS TEST`) would begin. Note that this action can be performed if the test is running normally (without any errors) or if the test is presently looping on any error encountered.

Control-l Key — Sun-3/400 Series and Sun-3/80

Holding down the [Control] key while pressing the `l` key any time during a self-test causes that particular self-test to terminate and control to be transferred to the `TEST LOOP MENU`. From within this menu any particular self-test can be executed and looped directly, without executing the self-tests that normally come before it. Once a test is invoked from the `TEST LOOP MENU`, a test control flag is set so that once that test has finished it will go back to the beginning of that one particular test and start it all over again. In order to proceed to the next self-test after a particular self-test has been invoked from this menu, enter the [Control-q] sequence or the [Esc] key to go to next test or to the Monitor (as described above). See the section on the Test Loop Menu for more details.

More Interactive Self-Test Commands

b Press the **b** (a mnemonic for *burn-in*) key, prior to the display of the `...Completed` or `Selftest Finished` message, to execute the power-up test sequence indefinitely. This option is useful during the manufacturing burn-in stage.

For the Sun-3/80, when the last selftest is finished the message tests`Automatically`continuing will be displayed and the self-test will be restarted again, using the System Enable Register read test as the first test. The SCC and terminal I/O tests are bypassed in burn-in mode since operator interaction is required. This looping sequence will continue until an ESCAPE is entered, a reset is done, or the "*b*" key pressed again to turn burn-in mode off. The burn-in mode can be toggled by successive pressing of the "*b*" key. Note that this burn-in mode key can be processed during a test that is running normally (no errors), if the test is presently looping on an error encountered, or at the end of selftest when the "Selftest Finished" is displayed and an operator input is requested.

s Press the **s** key prior to the display of the `...Completed` message to *re-start* the power-up test sequence.

Space Bar

If one of the power-up tests fails, it will continue to re-execute forever unless interrupted. Press the space bar to terminate the failed test and execute the next power-up test.

`Control-m`

Holding down the Control key while pressing **m** causes this test to end and sets a flag that allows execution of only the quick memory tests.

`Control-b`

Holding down the Control key while entering **b** causes this test to end and sets a test control flag so that all the tests that require the Bus Error circuitry to work will be skipped. This test is also the equivalent of a "Control-m". It assumes that 8MB of ECC memory are present on the board, because the test is executed before memory has been sized. This test displays no error messages. The Bus Error dependent tests skipped are:

Bus Error Register Test
The Level 1 to Level 3 Interrupt tests
P4 Video RAM Board Tests.

A Successful Sun-3/80 Self-Test

For a Sun-3/80, After the selftest have completed the final status of the selftest sequence will be displayed, as shown by the following example:

```
        END OF SELFTEST # 0x00000005 (SELFTEST FAILED)
        #PASSED = 0x00000004, #FAILED = 0x00000001

<Initializing Main Memory 0x0000000C Megabytes Initialized>

Type a Character within 10 seconds to enter the Menu Tests...(e for echo mode)
```

Test Loop Menu — Sun-3/400 Series and Sun-3/80

The Test Loop Menu is a menu appears when you press `Control-l` (loop) key either during Sun-3/400 series or Sun-3/80 self-test execution. Test. The menu allows the operator to transfer self-test control directly to a specified self-test and loop that test continually. This option is intended to support debugging a specific failing test or section of hardware without running all of the previous self-tests.

Once in the Test Loop Menu the following display will be shown on the terminal:

```
                   <<< TEST LOOP MENU >>>
                  (a)System Enable Register Test
                  (b)I/O Mapper RAM Write/Write/Read Test
                  (b)I/O Mapper RAM Address Test
                  (c)I/O Mapper RAM 3-Pattern Test
                  (t)ECC Memory Forced CE Test
                  (u)ECC Memory Forced UE Test
                  <<< Press ,Space for more, <Esc> to quit or select an option >>>

                                    or, for the Sun-3/80

                   <<< TEST LOOP MENU >>>
                  (a)System Enable Register Test
                  (b)I/O Mapper RAM Write/Write/Read Test
                  (b)I/O Mapper RAM Address Test
                  (c)I/O Mapper RAM 3-Pattern Test
                  (w)P4 Enable plane Write/Write Read Test
                  (x)P4 Color Plane Write/Write/Read Test
                  <<< Press ,Space for more, <Esc> to quit or select an option >>>
```

At this point you may either press the space bar to see more self-tests, or enter the key code for the test you want to execute in loop mode. Once a test is selected, that test is invoked and executed continually, regardless of whether the test passed or failed. From this point you can use any of the special control keys to continue the selftest if desired: `control-q` to continue the self-test in non-loop mode or `Esc` to abort all self-tests and go to the PROM Monitor.

Successful Diagnostic Boot Display

For a Sun-3/400 series or Sun-3/80 system, upon self-test completion a status message that looks something like this is displayed:

```
END OF SELFTEST #0x00000005 (SELFTEST FAILED)

#PASSED=0x00000004,#FAILED=0X00000001
```

NOTE *There is no diagnostic switch on a Sun-3/80; an EEPROM setting is required to allow a diagnostic boot-up. Enter the PROM monitor mode and use the* `q` *command to write* **12** *to location 0x70b. Refer to "Displaying and Modifying Memory" for information on use of the* `q` *command.*

During a diagnostic boot, after self-test has completed successfully, you will be prompted to enter the extended tests, as shown below.

```
Selftest passed.

Optional Menu Tests

Type a character within 10 seconds to enter Menu Tests...(e for echo mode)
```

If you press a terminal key during this time the Extended Test Menu is displayed on the terminal screen. See the "Sun-3 Extended Test Sequence" chapter in this document for details.

If you do not assert control of the system by pressing a terminal key within the 10 seconds delay time the Boot PROM program will next display the Sun logo and message on the console.

Remote Testing

If you press **e** on a *dumb terminal* keyboard, during the ten second period, all subsequent output will appear on *both* the console video monitor *and* a terminal attached to Serial Port A or B.

The purpose of this feature is to enable an individual at a remote site, using a terminal attached to a non-local machine by way of a telephone line with modems at each end, and the individual on-site to observe the system's behavior simultaneously. Once a system is in this mode, the local and non-local parties can communicate by employing the # command. Specifically, if the person at the remote site would like to send a message to the person on-site, he or she would simply type the # before typing the actual message. In order to terminate the message, the person at the remote site would enter a second #. At this point, the on-site person would be able to respond by prefixing and terminating their response with the #.

The # option will be very useful in the situation where an on-site customer has contacted a remote repair depot regarding a potential hardware problem. The repair person at the remote repair depot could initiate diagnostics on the non-local machine and allow both parties to observe the output. Beyond that, the repair person and the customer could communicate by way of the # command.

Diagnostic Self-test Sequence

The following text lists diagnostic mode self-tests for each Sun-3 system. If the diagnostic switch is enabled, the name of each test appears on the terminal until all self-tests are complete. The tests differ slightly according to the system architecture. The examples that follow represent each Sun-3 workstation.

Figure 1 *Sun-3/75, 3/140, 3/150, 3/160, and 3/110 Diagnostic Boot Sequence*

```
Boot PROM Selftest

  PROM Checksum Test
  DVMA Reg Test
  Context Reg Test
  Segment Map Wr/Rd Test
  Segment Map Address Test
  Page Map Test
  Memory Path Data Test
  NXM Bus Error Test
  Interrupt Test
  TOD Clock Interrupt Test
  MMU Access Bit Test
  MMU Access/Modify Bit Test
  MMU Invalid Page Test
  MMU Protected Page Test
  Parity Test
  Memory Size = 0x00000xxx Megabytes
  Memory Test (testing xxxxxxxx MBytes)

Selftest passed

Optional Menu Tests

Type character within 10 seconds to enter menu tests... (e for echo mode)
```

Figure 2 *Sun-3/50 and Sun-3/60 Diagnostic Boot Sequence*

```
Boot PROM Selftest

  PROM Checksum Test
  Context Reg Test
  Segment Map Wr/Rd Test
  Segment Map Address Test
  Page Map Test
  Memory Path Data Test
  NXM Bus Error Test
  Interrupt Test
  TOD Clock Interrupt Test
  MMU Access Bit Test
  MMU Access/Modify Bit Test
  MMU Invalid Page Test
  MMU Protected Page Test
  Parity Tests
  Memory Size = 0x00000xxx Megabytes
  Memory Test (testing xxxxxxxx MBytes)

Selftest passed

Optional Menu Tests

Type a character within 10 seconds to enter Menu Tests... (e for echo mode)
```

Figure 3 *Sun-3/260 and Sun-3/280 Diagnostic Boot Sequence*

```
Boot PROM Selftest

  PROM Checksum Test
  DVMA Reg Test
  Context Reg Test
  Segment Map Wr/Rd Test
  Segment Map Address Test
  Page Map Test
  Memory Path Data Test
  NXM Bus Error Test
  Interrupt Test
  TOD Clock Interrupt Test
  MMU Access Bit Test
  MMU Access/Modify Bit Test
  MMU Invalid Page Test
  MMU Protected Page Test
  ECC Error Tests
  Cache Data 3 Pat Test
  Cache Tags 3 Pat Test
  Memory Size = 0x00000xxx Megabytes
  Memory Test (testing xxxxxxxx MBytes)

Selftest passed

Optional Menu Tests

Type character in next 10 seconds to enter menu tests... (e for echo mode)
```

Figure 4 *Sun-3/400 Series Diagnostic Boot Sequence*

```
*System_name* Boot Prom Self-Test
(Hit key for character echo, space bar for next test, control-m, control-b)
System Enable Register Read Test {pass 0x00000001}
PROM Checksum Test
I/O Mapper RAM Write/Write/Read Test {pass 0x00000001}
I/O Mapper RAM Address Test {pass 0x00000001}
I/O Mapper RAM 3-Pattern Test {pass 0x00000001}
Bus Error Register Test {pass 0x00000001}
Level 1 Interrupt Test {pass 0x00000001}
Level 2 Interrupt Test {pass 0x00000001}
Level 3 Interrupt Test {pass 0x00000001}
TOD Clock Interrupt Test {pass 0x00000001}
<Sizing Main Memory>
<Memory Size = 0x00000010>
Memory Write/Write/Read Test {pass 0x00000001}
Memory Address Test {pass 0x00000001}
Memory 3-Pattern Test {pass 0x00000001}
Memory Read Byte Alignment Test {pass 0x00000001}
Memory Write Byte Alignment Test {pass 0x00000001}
Parity Memory No Error Test {pass 0x00000001}
Parity Memory Forced Error Test {pass 0x00000001}
ECC Memory No Error Test {pass 0x00000001}
ECC Memory Forced CE Test {pass 0x00000001}
ECC Memory Forced UE Test {pass 0x00000001}
Central Cache Tag RAM Write/Write/Read Test {pass 0x00000001}
Central Cache Tag RAM Address Test {pass 0x00000001}
Central Cache Tag RAM 3-Pattern Test {pass 0x00000001}
Central Cache Data RAM Write/Write/Read Test {pass 0x00000001}
Central Cache Data RAM Address Test {pass 0x00000001}
Central Cache Data RAM 3-Pattern Test {pass 0x00000001}
Central Cache Data RAM Read Byte Alignment Test {pass 0x00000001}
Central Cache Data RAM Write Byte Alignment Test {pass 0x00000001}
Central Cache Read Hit Test {pass 0x00000001}
Central Cache Invalid Read Miss Test {pass 0x00000001}
Central Cache Valid Read Miss Test {pass 0x00000001}
Central Cache Write Hit Test {pass 0x00000001}
Central Cache Write Miss, No Writeback Test {pass 0x00000001}
Cache Write Miss, Writeback Test {pass 0x00000001}
Central Cache Line Cross Invalid Read Miss Test {pass 0x00000001}
Central Cache Line Cross Write Miss Writeback Test {pass 0x00000001}
Central Cache Writeback Timeout Test {pass 0x00000001}
Block Copy (Source=Cache Miss,Dest=Cache Miss) Test {pass 0x00000001}
Block Copy (Source=Cache Miss,Dest=Cache Hit) Test {pass 0x00000001}
Block Copy (Source=Cache Hit,Dest=Cache Miss) Test {pass 0x00000001}
Block Copy (Source=Cache Hit,Dest=Cache Hit) Test {pass 0x00000001}
```

```
Memory Write/Write/Read Test (Central Cache on) {pass 0x00000001}
IOC Tag RAM Write/Write/Read Test {pass 0x00000001}
IOC Tag RAM Address Test {pass 0x00000001}
IOC Tag RAM 3-Pattern Test {pass 0x00000001}
IOC Data RAM Write/Write/Read Test {pass 0x00000001}
IOC Data RAM Address Test {pass 0x00000001}
IOC Data RAM 3-Pattern Test {pass 0x00000001}
IOC Data RAM Read Byte Alignment Test {pass 0x00000001}
IOC Data RAM Write Byte Alignment Test {pass 0x00000001}
VME Loopback Test {pass 0x00000001}
VME Loopback and DVMA Test {pass 0x00000001} (not for Sun-3/460)
IOC Read Hit Test {pass 0x00000001}
IOC Invalid Read Miss Test {pass 0x00000001}
IOC Write Hit Test {pass 0x00000001}
IOC Write Miss, No Writeback Test {pass 0x00000001}
IOC Write Miss, Writeback Test {pass 0x00000001}
IOC Read Miss, Writeback Test {pass 0x00000001}
IOC Valid Write Hit (Central Cache Match,Unmod) Test {pass 0x00000001}
IOC Invalid Write Miss (Central Cache Match,Unmod) Test {pass 0x00000001}
IOC Invalid Read Miss (Central Cache Match,Unmod) Test {pass 0x00000001}
IOC Invalid Read Miss (Central Cache Match,Modified) Test {pass 0x00000001}
IOC Valid Read Miss (Central Cache Match),Writeback Test {pass 0x00000001}
IOC Flush (Valid, Modified) Test {pass 0x00000001}
IOC Flush (Valid, Not Modified) Test {pass 0x00000001}
IOC Flush (Not Valid, Not Modified) Test {pass 0x00000001}
IO Mapper Invalid Page (IO.DT) Test {pass 0x00000001}
IOC Write Miss, Writeback (Write Protect) Test {pass 0x00000001}
IOC Invalid Read Miss (IO Mapper IO.EN = 0) Test {pass 0x00000001}
IOC Write Miss (IO Mapper IO.EN = 0) Test {pass 0x00000001}
IOC Random Data Block Write Test {pass 0x00000001}
IOC Random Data Block Read (Central Cache off) Test {pass 0x00000001}
IOC Random Data Block Read (Central Cache on) Test {pass 0x00000001}
<P4 Low Resolution Color Video RAM Board Detected>
P4 Overlay Frame Buffer Write/Write/Read Test {pass 0x00000001}
P4 Overlay Frame Buffer Address Test {pass 0x00000001}
P4 Overlay Frame Buffer 3-Pattern Test {pass 0x00000001}
P4 Overlay Frame Buffer March Test {pass 0x00000001}
P4 Overlay Frame Buffer Read Byte Alignment Test {pass 0x00000001}
P4 Overlay Frame Buffer Write Byte Alignment Test {pass 0x00000001}
P4 Enable Plane Write/Write/Read Test {pass 0x00000001}
P4 Color Plane Write/Write/Read Test {pass 0x00000001}

Selftest passed

Type character within 10 seconds to enter Extended Tests System (e for echo mode)
```

Figure 5 *Sun-3/80 Diagnostic Boot Sequence*

```
Sun-3/80 Boot PROM Selftest Revision xx
  (Press <Esc> to abort tests or <Cntrl-L> for the Loop Menu)
  System Enable Register Read Test {pass 0x00000001}  hh:mm:ss
  PROM Checksum Test {pass 0x00000001}  hh:mm:ss
  Clock/Calendar Device {pass 0x00000001}  hh:mm:ss
  I/O Mapper RAM Write/Write/Read Test {pass 0x00000001}  hh:mm:ss
  I/O Mapper RAM Address Test {pass 0x00000001}  hh:mm:ss
  I/O Mapper RAM 3-Pattern Test {pass 0x00000001}  hh:mm:ss
  <Memory Size = 0x00000010>  hh:mm:ss
  Memory Address Test {pass 0x00000001}  hh:mm:ss
  Memory Read Byte Alignment Test {pass 0x00000001}  hh:mm:ss
  Memory Write Byte Alignment Test {pass 0x00000001}  hh:mm:ss
  I/O Mapper RAM Write/Write/Read Test {pass 0x00000001}  hh:mm:ss
  I/O Mapper RAM Address Test {pass 0x00000001}  hh:mm:ss
  I/O Mapper RAM 3-Pattern Test {pass 0x00000001}  hh:mm:ss
  Bus Error Register Test {pass 0x00000001}  hh:mm:ss
  Level 1 Interrupt Test {pass 0x00000001}  hh:mm:ss
  Level 2 Interrupt Test {pass 0x00000001}  hh:mm:ss
  Level 3 Interrupt Test {pass 0x00000001}  hh:mm:ss
  Configuration Memory Check {pass 0x00000001}  hh:mm:ss
  LANCE Controller Check {pass 0x00000001}  hh:mm:ss
  EPS SCSI Check {pass 0x00000001}  hh:mm:ss
  Floppy Controller Check {pass 0x00000001}  hh:mm:ss
  Printer Controller Check {pass 0x00000001}  hh:mm:ss
  <P4 Low Resolution Color Video RAM Board Detected>
  P4 Overlay Frame Buffer Write/Write/Read Test {pass 0x00000001} hh:mm:ss
  P4 Overlay Frame Buffer Address Test {pass 0x00000001}  hh:mm:ss
  P4 Overlay Frame Buffer 3-Pattern Test {pass 0x00000001}  hh:mm:ss
  P4 Overlay Frame Buffer March Test {pass 0x00000001}  hh:mm:ss
  P4 Overlay Frame Buffer Read Byte Alignment Test {pass 0x00000001} hh:mm:ss
  P4 Overlay Frame Buffer Write Byte Alignment Test {pass 0x00000001} hh:mm:ss
  P4 Enable Plane Write/Write/Read Test {pass 0x00000001}  hh:mm:ss
  P4 Color Plane Write/Write/Read Test {pass 0x00000001}  hh:mm:ss

  END OF SELFTEST #0x00000005 (SELFTEST PASSED/FAILED)
  #PASSED = 0x00000004, #FAILED = 0x00000001

  Initializing Main Memory n Megabytes Initialized

  Type a character within 10 seconds to enter menu tests.
```

Refer to the *PROM User's Manual* for descriptions of the self-tests named for workstations other than the Sun-3/400 series and the Sun-3/80.

Here is a summary of the LED displays for the Sun-3/400 series and Sun-3/80 during the power-up self-tests. Many of the tests are the same for both systems; the test description will indicate which tests is system-specific, In case of error in a Sun-3/400 series test, the LED in bit position 7 also lights. The text that follows this chart shows both the normal test and error displays. As indicated, many tests share the same LED display.

Sun-3/80 LED

LED status	Visual	Condition
Blinking	⊖	Testing
OFF	○	Error
ON (Green)	●	Ok

The Sun-3/80 has one green LED only, which lights steadily during normal function, blinks during a test, and turns off when there is an error. The messages that appear on a terminal attached to Serial Port A describe the specific test in progress and the nature of any errors found.

The following table explains the Sun-3/400 series LED code.

LED Display ● = ON, ○ = OFF 7 6 5 4 3 2 1 0	*SelfTest being Performed.*
● ● ● ● ● ● ● ●	A reset will set LEDs to this state.
○ ○ ○ ○ ○ ○ ○ ●	Keyboard/Mouse SCC Write/Read Test
○ ○ ○ ○ ○ ○ ● ●	System Enable Register Read Test
○ ○ ○ ○ ○ ● ○ ○	PROM Checksum Test
○ ○ ○ ○ ○ ● ○ ●	I/O Mapper RAM Test(s)
○ ○ ○ ○ ○ ● ● ○	Bus Error Register Test
○ ○ ○ ○ ○ ● ● ●	Interrupt Test(s)
○ ○ ○ ○ ● ○ ○ ○	ECC Memory Sizing and Test(s)
○ ○ ○ ○ ● ○ ○ ●	Parity Memory Sizing and Test(s)
○ ○ ○ ○ ● ○ ● ○	ECC Memory Forced Error Test(s)
○ ○ ○ ○ ● ○ ● ●	Central Cache Tag RAM Test(s)
○ ○ ○ ○ ● ● ○ ○	Central Cache Data RAM Test(s)
○ ○ ○ ○ ● ● ○ ●	Central Cache Hit/Miss Test(s)
○ ○ ○ ○ ● ● ● ○	Block Copy Test(s)
○ ○ ○ ○ ● ● ● ●	Memory Write/Write/Read Test(Central Cache on)
○ ○ ○ ● ○ ○ ○ ○	IOC Tag RAM Test(s)
○ ○ ○ ● ○ ○ ○ ●	IOC Data RAM Test(s)
○ ○ ○ ● ○ ○ ● ○	VME Loopback Test
○ ○ ○ ● ○ ○ ● ●	VME Loopback and DVMA (not for 3/460)
○ ○ ○ ● ○ ● ○ ○	IOC Read/Write/Flush Test(s)
○ ○ ○ ● ○ ● ○ ●	P4 Overlay Frame Buffer Test(s)

Self-Test Descriptions

The following paragraphs describe each individual test, and the messages and indications generated if it fails. Some tests do not apply to both the Sun-3/400 series and Sun-3/80; refer to the Diagnostic Boot Sequence display examples on previous pages for lists of applicable self-tests.

LED Register Test

The first Sun-3/400 series test is indicated by a slow loop through the Diagnostic LEDs, from LED 7 through LED 0. It is intended to determine if the CPU is able to fetch instructions correctly from the Diagnostic PROM and transfer data across the data bus to the LED status register.

NOTE *If all the LED s on a Sun-3/400 series remain lighted, you could have a low voltage or board seating problem, or the system could contain the wrong Boot PROM.*

NOTE *The diagrams that follow depict the LED states for each self-test. Note that the LEDs are shown here for convenience in reading as binary numbers; the least significant bit is on the right. In most systems, the LEDs are read with bit 0 on the left for desktop installations and with bit 0 on top for deskside installations.*

The tests are described here in the order that they are executed.

UART SCC (Z8530) Port A,B Write/Read Test

This test checks the ability of the Sun-3/400 series workstation CPU to communicate with port A and B of the Zilog Z8530 Serial Communications Chip (SCC). It performs a write/read test of port A and B SCC chip internal registers WR12/RR12. The purpose of this test is to test the path between the CPU and the SCC so that all subsequent tests may display the test name and error status to a terminal attached to Serial Port A.

The test enters a scope loop on a data compare error, but no error messages are displayed on the terminal during this test.

The diagram below summarizes the LED states for a Sun-3/400 series CPU board.

Hexadecimal Value Of LEDs	*Visual Representation*	*Condition*
0x00	bit 7 oooooooo bit 0	okay
0x80	●ooooooo	error

Keyboard/Mouse SCC (Z8530) Port A,B Write/Read Test

This test checks the ability of the CPU to communicate with port A and B of the Zilog Z8530 Serial Communications Chip (SCC). It performs a write/read test of port A and B SCC chip internal registers WR12/RR12. Note that this test is for the SCC used for the keyboard and mouse.

The test enters a scope loop on a data compare error, but no error messages are displayed on the terminal during this test because this is a test of the path to the SCC chip.

The diagram below summarizes the LED states for a Sun-3/400 series workstation.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
1	**0x01**	**bit 7 ooooooo● bit 0**	**okay**
1	**0x81**	**●oooooo●**	**error**

System Enable Register Read Test

This test reads the System Enable Register and verifies that all bits read are zero, ignoring the Diagnostics Switch bit.

The test enters a scope loop on data compare errors, with the following terminal message:

```
exp xxxxxxxx, obs xxxxxxxx, xor xxxxxxxx
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
3	**0x03**	**bit 7 oooooo●● bit 0**	**okay**
3	**0x83**	**●ooooo●●**	**error**

PROM Checksum Test

The checksum of all locations in both PROMs with the exception of the last word is calculated and compared to an expected value that is stored in the last word of the second PROM. If this test fails, the PROMs should be replaced.

The test enters a scope loop on data compare errors, with the following terminal message:

Checksum error: Exp xxxxxxxx, Obs xxxxxxxx

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
4	**0x04**	**bit 7 ○○○○○●○○ bit 0**	**okay**
4	**0x84**	**●○○○○●○○**	**error**

I/O Mapper Write/Write/Read Test

This test verifies the address and data paths to the IO Mapper RAM, as well as address bit and data bit uniqueness.

For each test address of IO Mapper RAM:
(base+0x0,base+0x01,base+0x02,base+0x04,base+0x08,...,base+iomapper_size)

For each data pattern at each test address:
(0x0,0x1,0x2,0x4,0x10,0x20,...0x80000000)
The test does the following:

- Writes test data to the test address.
- Writes inverted test data to test address + 0x04.
- Reads back data from the test address and compares.

Upon error, the test loops through the steps shown above with constant test data and a constant test address.

This test enters a scope loop on data compare errors, with the following terminal message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
5	**0x05**	**bit 7 ○○○○○●○● bit 0**	**okay**
5	**0x85**	**●○○○○●○●**	**error**

I/O Mapper RAM Address Test

This test writes the complete I/O Mapper RAM address space with the longword address as the data, then reads back the entire address space and verifies that no addresses are overwritten. This is a test for addressing uniqueness of the I/O Mapper RAM.

This test enters a scope loop on data compare errors, with the following terminal message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display shown for the I/O Mapper Write/Write/Read Test applies to this test also.

I/O Mapper RAM 3-Pattern Test

This test writes the complete I/O Mapper RAM address space with a repeated three-long-word pattern sequence, then reads back the entire address space and verifies the data.

This test enters a scope loop on data compare errors, with the following terminal message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display shown for the I/O Mapper Write/Write/Read Test applies to this test also.

Bus Error Register Test

This test does the following:

1. Verifies that attempting to read an invalid memory address causes a bus error, with appropriate data written to the BUSERR register.
2. Verifies that attempting to read an FPA device address while the ENABLE-FPA bit in the System Enable Register is off causes a bus error, with appropriate data written to the BUSERR register.

This test enters a scope loop upon error, with one or more of the following terminal error messages:

```
error1: No Bus Error Reading Invalid Memory Address, read addr = 0xyyyyyyyy.
error2: Bus error reg TIMEOUT bit not set.
error3: No Bus Error Reading Disabled FPA, read addr = 0xyyyyyyyy.
error4: Bus error reg FPAENERR bit not set.
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
6	**0x06**	**bit 7 ooooo●●o bit 0**	**okay**
6	**0x86**	**●oooo●●o**	**error**

Level 1 Interrupt Test

This test forces a Level 1 soft interrupt to verify that an autovector Level 1 interrupt will occur.

This test enters a scope loop upon error, with the following error message:

```
error: No level 1 interrupt occurred.
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
7	**0x07**	**bit 7 ooooo●●● bit 0**	**okay**
7	**0x87**	**●oooo●●●**	**error**

Level 2 Interrupt Test

This test forces a level 2 soft interrupt to verify that an autovector Level 1 interrupt will occur.

The test enters a scope loop upon error, with the following terminal error message:

```
error: No level 2 interrupt occurred.
```

The LED display for this test is the same as that for the Level 1 Interrupt test.

Level 3 Interrupt Test

This test forces a level 3 soft interrupt to verify that an autovector Level 3 interrupt will occur.

The test enters a scope loop upon error, with the following error message:

```
error: No level 3 interrupt occurred.
```

The LED display for this test is the same as that for the Level 1 Interrupt test.

TOD Clock Interrupt Test

This test enables the TOD CLock to interrupt and then verifies that the interrupt occurs.

This test enters a scope loop upon error, with the following error message:

```
error: No TOD interrupt occurred.
```

The LED display for this test is the same as that for the Level 1 Interrupt test.

Memory Write/Write/Read Test

This test verifies the address and data paths to Main Memory, as well as address bit and data bit uniqueness.

For each test address of main memory:
(0x0,0x1,0x2,0x4,0x8,0x10,0x20,...size_of_mem_installed)

For each data pattern at each test address:
(0x0,0x1,0x2,0x4,0x10,0x20,...0x80000000)
The test does the following:

- Writes test data to test address.
- Writes inverted test data to test address + 0x04.
- Reads back data from test address and compares.

Upon error, the test loops through the steps shown above with constant test data and a constant test address.

This test enters a scope loop on data compare errors, with the following message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
8	**0x08**	**bit 7 oooo●ooo bit 0**	**okay**
8	**0x88**	**●ooo●ooo**	**error**

Memory Address Test

This test writes the complete Main Memory address space with the longword address as the data, then reads back the entire address space and verifies that no addresses are overwritten. This is a test for addressing uniqueness of the Main Memory RAM.

This test enters a scope loop on data compare errors, with the following terminal message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display is that same as that shown for the Memory Write/Write/Read Test.

Memory 3-Pattern Test

This test writes the complete Main Memory address space with a repeated three long word pattern sequence, then reads back the entire address space and verifies the data.

This test enters a scope loop on data compare errors, with the following terminal message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display is that same as that shown for the Memory Write/Write/Read Test.

Memory Read Byte Alignment Test

This test verifies that byte, word, and long word read operations produce the appropriate byte-aligned data.

The test enters a scope loop on data compare errors, with the following message:

```
byte misalignment: addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display is that same as that shown for the Memory Write/Write/Read Test.

Memory Write Byte Alignment Test

This test verifies that byte, word, and long word read operations produce the appropriate byte-aligned data for various byte-aligned write operations.

This test enters a scope loop on data compare errors, with the following terminal message:

```
byte misalignment: addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display is that same as that shown for the Memory Write/Write/Read Test.

Parity Memory No Error Test

This test will execute only if a Parity Memory board was detected when main memory was sized.

The test does the following:

1. Determines if Parity memory is installed. If not, it bypasses the test.
2. Initializes all Parity memory with zero to initialize the check bits.
3. For each test address 0x00000000,0x00000001,0x00000002,...,0x00040000, 0x00800000,0x00800001,0x00800002,...0x00840000, or up to total Parity memory installed (16MB max):

   For each data pattern 0x00000001,0x00000003,0x00000007,...0xffffffff at one specific test address:

   The test:
   (a) Writes 0x0 to Memory Error Address register to clear previous interru
   (b) Enables Parity checking on Parity board.
   (c) Enables Level 7 interrupts in Memory Error Control Register.
   (d) Enables system interrupts in System Interrupt Register.
   (e) Verifies that Memory Error Control register was properly set.
   (f) Writes longword test data to test address.
   (g) Reads longword from test address.
   (h) Verifies that no Level 7 interrupt occurred.
   (i) Verifies that Memory Error Control register was not modified.

Upon error, the test loops through steps a - i with a constant test address and constant test data.

Possible error messages for this test are:

```
error1: Memory Error Control Reg not written:
exp=0x00000000, obs=0x00000000, testaddr=0x00000000, testdata=0x00000000
error2: Unexpected parity error (level 7) interrupt:
testaddr=0x00000000, testdata=0x00000000
Memory Error Control Reg: obs=0x00000000
Parity Memory Error Reg:  obs=0x00000000
Memory Error Address Reg: obs=0x00000000
error3: Bad Memory Error Control Reg after memory write:
exp=0x00000000, obs=0x00000000, testaddr=0x00000000, testdata=0x00000000
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
9	**0x09**	**bit 7 oooo•oo• bit 0**	**okay**
9	**0x89**	**•ooo•oo•**	**error**

Parity Memory Forced Error Test

This test will execute only if a Parity Memory board was detected when main memory was sized.

For each test address 0x00000000,0x00000001,0x00000002,...,0x00040000, 0x00800000,0x00800001,0x00800002,...0x00840000 or up to total Parity memory installed (16MB max):

For each data pattern 0x1,0x2,0x4,...0x00000100 at one specific test address:

The test does the following:

Writes 0x0 to Memory Error Address register to clear previous interrupt.
Enables Parity memory board.
Writes longword 0x0 to test address, test address + 4 (good parity).
Enables Parity checking and Parity Test on Parity board.
Enables Level 7 interrupts in Memory Error Control Register.
Enables system interrupts in system interrupt Register.
Writes longword test data to test address.
Reads longword from test address.
Verifies that Level 7 interrupt occurred.
Verifies that Memory Error Address register captured test address.
Verifies that Parity Memory Error register was set correctly
Verifies that Memory Error Control register was set correctly.

Upon error, the test loops through the steps shown above with a constant test address and constant test data.

Possible error messages for this test are:

```
error1: no parity error (level 7) interrupt occurred when bad parity forced.
addr = 0x00000000, wr/rd data = 0x00000000

error2: Memory Error Address Reg: exp=0x00000000, obs=0x00000000
addr = 0x00000000, wr/rd data = 0x00000000

error3: Parity Memory Error Reg: exp=0x00000000, obs=0x00000000
addr = 0x00000000, wr/rd data = 0x00000000

error4: Bad Memory Error Control Reg after memory write:
exp=0x00000000, obs=0x00000000, testaddr=0x00000000, testdata=0x00000000
```

The LED display shown for the Parity Memory No Error Test applies to this test also.

ECC Memory No Error Test

This test does the following:

- Checks if ECC memory is installed. If not, it bypasses this test.
- Determines the starting address and size of total ECC memory.
- Initializes all ECC memory with zero to initialize ECC check bits.
- For each test address 0x00000000,0x00000001,0x00000002,...,0x00040000, 0x00800000,0x00800001,0x00800002,...0x00840000, 0x01000000,0x01000001,0x01000002,...0x01040000, 0x01800000,0x01800001,0x01800002,...0x01840000, ...,

0x08800000,0x08800001,0x08800002,...0x00880000, or up to total memory installed (144MB max) and for each data pattern 0x00000001,0x00000003,0x00000007,...0xffffffff at one specific test address, the test does the following:

(a) Enables ECC checking on all ECC boards.
(b) Enables Level 7 interrupts in the Memory Error Control Reg.
(c) Enables system interrupts in the System Interrupt Register.
(d) Verify that Memory Error Control register was properly set.
(e) Reads a longword from test address.
(f) Verifies that no Level 7 interrupt occurred.
(g) Verifies that the Memory Error Control register was not modified.

Upon error, the test loops through steps a - g with a constant test address and constant test data.

Possible error messages for this test are:

```
error1: Memory Error Control Reg not written:
exp=0x00000000, obs=0x00000000, testaddr=0x00000000, testdata=0x00000000

error2: Unexpected ECC error (level 7) interrupt:
testaddr=0x00000000, testdata=0x00000000
Memory Error Control Reg: obs=0x00000000
Memory Error Address Reg: obs=0x00000000
Syndrome Reg (Board 0):   obs=0x00000000

error3: Bad Memory Error Control Reg after memory write:
exp=0x00000000, obs=0x00000000, testaddr=0x00000000, testdata=0x00000000
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
10	0x0a	bit 7oooo●o●o bit 0	okay
10	0x8a	●ooo●o●o	error

ECC Memory Forced CE Test

This test does the following:

- Gets the starting address and size of total ECC memory.
- For each test address 0x00000000,0x00800000,0x01000000,...0x07800000, (starting at ECC memory base), or up to total ECC memory installed, the test:

  (a) Resets syndrome register.
  (b) Enables CE, level 7 interrupts in the Memory Error Control Register.
  (c) Verifies that Memory Error Control register was properly set.
  (d) Writes data to memory.
  (e) Writes check bits to the Diagnostic registers.
  (f) Enables ECC checking, DM1 on memory board.
  (g) Enables system interrupts in the System Interrupt Register.
  (h) Reads a longword from test address.
  (i) Verifies that a Level 7 interrupt occurred.
  (j) Verifies that the Memory Error Control register was modified correctly.
  (k) Verifies that the CE bit was set in the Memory board syndrome register.
  (l) Verifies that the syndrome code in the syndrome register was correctly set.

Upon error, the test loops through steps a - l.

Possible error messages for this test are:

```
error1: Memory Error Control Reg not written:
exp=0x00000000, obs=0x00000000, testaddr=0x00000000

error2: No ECC error (level 7) interrupt occurred after CE forced:
testaddr=0x00000000

error3: Bad Memory Error Control Reg after CE forced:
exp=0x00000000, obs=0x00000000, testaddr=0x00000000

error4: CE bit in ECC syndrome reg not set after CE forced:
testaddr=0x00000000

error5: Bad ECC Syndrome Reg Syndrome code (b31:24) after CE forced:
testaddr=0x00000000, exp=0x00000000, obs 0x00000000
```

The LED display for this test is the same as that for the ECC Memory No Error Test.

ECC Memory Forced UE Test

This test does the following:

- Determines the starting address and size of total ECC memory.
- For each test address 0x00000000,0x00800000,0x01000000,...0x07800000, (starting at ECC memory base), or up to total ECC memory installed, the test:

  (a) Resets the Syndrome Register.
  (b) Enables Level 7 interrupts in the Memory Error Control Register.
  (c) Verifies that the Memory Error Control register was properly set.
  (d) Writes data to memory.
  (e) Writes check bits to the Diagnostic registers.
  (f) Enables ECC checking, DM1 on the memory board.
  (g) Enables system interrupts in the System Interrupt register.
  (h) Reads a longword from test address.
  (i) Verifies that a Level 7 interrupt occurred.
  (j) Verifies that the Memory Error Control register was correctly modified.
  (k) Verify that the syndrome code in Syndrome register was correctly set.

Upon error, the test loops through steps a -k.

Possible error messages for this test are:

```
error1: Memory Error Control Reg not written:
exp=0x00000000, obs=0x00000000, testaddr=0x00000000.

error2: No ECC error (level 7) interrupt occurred after UE forced:
testaddr=0x00000000

error3: Bad Memory Error Control Reg after UE forced:
exp=0x00000000, obs=0x00000000, testaddr=0x00000000

error5: Bad ECC Syndrome Reg Syndrome code (b31:24) after UE forced:
testaddr=0x00000000, exp=0x00000000, obs 0x00000000
```

The LED display for this test is the same as that for the ECC Memory No Error Test.

Central Cache Tag RAM Write/Write/Read Test

This test verifies the address and data paths to the Central Cache Tag RAM, as well as address bit and data bit uniqueness.

For each test address of Central Cache TAG RAM:
(base+0x0,base+0x10,base+0x40,base+0x80,...,base+cache_size)

For each data pattern at each test address:
(0x00000000,0x00010000,0x00020000,0x00040000,...0x80000000)

The test does the following:

(a) Writes test data to test address.
(b) Writes inverted test data to test address + 0x10 (next tag).
(c) Reads back data from test address and compare.
NOTE: Only bits 16:31 of data read back are valid!

Upon error, the test loops through steps a - c with constant test data and a constant test address.

This test enters a scope loop on data compare errors, with the following terminal message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
11	**0x0b**	**bit 7 oooo●o●● bit 0**	**okay**
11	**0x8b**	**●ooo●o●●**	**error**

Central Cache Tag RAM Address Test

This test writes the complete Central Cache Tag RAM address space with the longword address as the data, then reads back the entire address space and verifies that no addresses are overwritten. Only the most significant 16 bits (b31:16) are verified, as the least significant 16 (b15:00) are invalid for reads. This is a test for addressing uniqueness of the Central Cache Tag RAM.

This test enters a scope loop on data compare errors, with the following message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display for this test is the same as that for the Central Cache Tag RAM Write/Write/Read Test.

Central Cache Tag RAM 3-Pattern Test

This test writes the complete Central Cache Tag RAM address space with a repeated three-long-word pattern sequence, then reads back the entire address space and verifies the data. Only the most significant 16 bits (b31:16) are verified, because the least significant 16 (b15:00) are invalid for reads.

This test enters a scope loop on data compare errors, with the following terminal message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display for this test is the same as that for the Central Cache Tag RAM Write/Write/Read Test.

Central Cache Data RAM Write/Write/Read Test

This test verifies the address and data paths to the Central Cache Data RAM, as well as address bit and data bit uniqueness.

For each test address of Central Cache Data RAM:
(base+0x0,base+0x01,base+0x02,base+0x04,base+0x08,...,base+cache_size)

For each data pattern at each test address:
(0x0,0x1,0x2,0x4,0x10,0x20.....0x80000000)
The test does the following:

(a) Writes test data to test address.
(b) Writes inverted test data to test address + 0x04.
(c) Reads back data from test address and compares.

Upon error, the test loops through steps a - c with constant test data and a constant test address.

This test enters a scope loop on data compare errors, with the following message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The diagram below summarizes the LED states.

Test Number	***Hexadecimal Value Of LEDs***	***Visual Representation***	***Condition***
11	**0x0b**	**bit 7 ○○○○●○●● bit 0**	**okay**
11	**0x8b**	**●○○○●○●●**	**error**

Central Cache Data RAM Address Test

This test writes the complete Central Cache Data RAM address space with the longword address as data, then reads back the entire address space and verifies that no addresses are overwritten. This is a test for addressing uniqueness of the Central Cache Data RAM.

This test enters a scope loop on data compare errors, with the following message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
12	**0x0c**	**bit 7 oooo●●oo bit 0**	**okay**
12	**0x8c**	●ooo●●oo	**error**

Central Cache Data RAM 3-Pattern Test

This test writes the complete Central Cache Data RAM address space with a repeated three-long- word pattern sequence, then reads back the entire address space and verifies the data.

This test enters a scope loop on data compare errors, with the following message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display for this test is the same as that shown for the Central Cache Data RAM Address Test.

Central Cache Data RAM Read Byte Alignment Test

This test verifies that byte, word, and long word read operations produce the appropriate byte-aligned data.

The test enters a scope loop on data compare errors, with the following message:

```
byte misalignment: addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display for this test is the same as that shown for the Central Cache Data RAM Address Test.

Central Cache Data RAM Write Byte Alignment Test

This test verifies that byte, word, and long word read operations produce the appropriate byte-aligned data for various byte-aligned write operations.

This test enters a scope loop on data compare errors, with the following terminal message:

```
byte misalignment: addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display for this test is the same as that shown for the Central Cache Data RAM Address Test.

Central Cache Read Hit Test

This test verifies that a data operand read from system memory with the memory block address in the cache valid causes a read from the cache and not from system memory.

The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.
2. Clears Central Cache tags, data, and the first 64 Kbytes of system memory.
3. For each test address 0x0,0x1,0x2,0x4.....0x08000000 (128MB), or up to maximum memory installed, the test:

   (a) Writes 00-0f as data into memory for cache line of interest.
   (b) Writes ff-f0 as data into cache data RAM for cache line of interest.
   (c) Writes test address, valid bit ON in cache tag RAM.
   (d) Turns on central cache.
   (e) Reads from test address.
   (f) Turns off central cache.
   (g) Verifies that the data read is from cache, not system memory.
   (h) Verifies cache tag is still correct (unmodified).
   (i) Verifies cache data is still correct (unmodified).

Upon error, the test loops through steps a - i with a constant test address.

Possible error messages for this test are:

```
error1: Bad read data on cache valid read:
testaddr=0x00000000,readaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad cache tag ram after cache valid read:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad cache data ram after cache valid read:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
13	**0x0d**	**bit 7 oooo••o• bit 0**	**okay**
13	**0x8d**	**•ooo••o•**	**error**

Central Cache Invalid Read Miss Test

This test verifies that doing a data operand read from system memory with the memory block address in the cache invalid causes a read from system memory and not the cache. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.
2. Clears Central Cache tags, data, and the first 64 Kbytes of system memory.
3. For each test address 0x0,0x1,0x2,0x4....0x08000000 (128MB), or up to maximum memory installed, the test:

   (a) Writes 00-0f as data into memory for cache line of interest.
   (b) Writes ff-f0 as data into cache data RAM for cache line of interest.
   (c) Writes test address, valid bit OFF in cache tag RAM.
   (d) Turns on central cache.
   (e) Reads from test address.
   (f) Turns off central cache.
   (g) Verifies that the data read is from system memory, not cache.
   (h) Verifies cache tag is updated correctly.
   (i) Verifies cache data is updated correctly.

Upon error, the test loop through steps a - i with a constant test address. Possible error messages for this test are:

```
error1: Bad read data on cache invalid read:
testaddr=0x00000000, readaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad cache tag ram after cache invalid read:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad cache data ram after cache invalid read:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display for this test is the same as that shown for the Central Cache Read Hit Test.

Central Cache Valid Read Miss Test

This test verifies that a data operand read from system memory with a valid modulo 64 Kbyte memory block address in the cache causes a read from system memory and not the cache.

The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.
2. Clears Central Cache tags, data, and the first 64 Kbytes of system memory.
3. For each test address 0x0,0x1,0x2,0x4....0x08000000 (128MB), or up to maximum memory installed, the test:

   (a) Writes 00-0f as data into the memory line for test address (x).
   (b) Writes 50-5f as data into the memory line for addr x+64KB.
   (c) Writes ff-f0 as data into the cache data RAM for addr x.
   (d) Writes the test address, valid bit ON in cache tag RAM.
   (e) Turns on central cache.
   (f) Reads from address x+64KB.
   (g) Turns off central cache.
   (h) Verifies that the data read is from memory, not cache.
   (i) Verifies that the cache tag is updated correctly
   (j) Verifies that cache data is updated correctly.

Upon error, the test loops through steps a - j with a constant test address. Possible error messages for this test are:

```
error1: Bad system memory after cache valid write hit:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad cache tag ram after cache valid write hit:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad cache data ram after cache valid write hit:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the Central Cache Read Hit Test.

Central Cache Write Hit Test

This test verifies that doing a data operand write to system memory with the memory block address in the cache valid causes a write to the cache and not to system memory.

The test does the following:

1. Turns off EN_CACHE, EN_IOCACHE, EN_DVMA, EN_VME_LOOP in the System Enable Register.
2. Clears Central Cache tags, data, and the first 64 Kbytes of system memory.
3. For each test address 0x0,0x1,0x2,0x4....0x08000000 (128MB), or up to maximum memory installed, the test:

   (a) Writes the longword test address as data into memory.
   (b) Writes longword zero into cache data RAM.
   (c) Writes the address, valid bit ON in cache tag RAM.
   (d) Turns on central cache.
   (e) Writes the test address with inverted longword address as data.
   (f) Turns off central cache.
   (g) Verifies that the system memory location was unmodified, since the write should have gone only to the cache (no write-through).
   (h) Verifies that the cache tag was updated correctly (valid,dirty,addr bits).
   (i) Verifies that cache data was written correctly.

Upon error, the test loops through steps a - i with a constant test address. Possible error messages for this test are:

```
error1: Bad system memory after cache valid write hit:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad cache tag ram after cache valid write hit:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad cache data ram after cache valid write hit:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that shown for the Central Cache Read Hit Test.

Central Cache Write Miss, No Writeback Test

This test verifies that a data operand write to system memory address (X+64KB) with a valid, not dirty memory block address (X) in the cache causes a write to the cache and not to system memory, and that no writeback to system memory occurs.

The test does the following:

1. Turns off EN_CACHE, EN_IOCACHE, EN_DVMA, EN_VME_LOOP in the System Enable Register.
2. Clears Central Cache tags, data, and the first 64 Kbytes of system memory.
3. For each test address 0x0,0x1,0x2,0x4....0x08000000 (128MB), or up to maximum memory installed, the test:

   (a) Writes the longword address as data into memory.
   (b) Writes the inverted longword address as data into cache data RAM.
   (c) Writes the correct address, valid bit ON in cache tag ram.
   (d) Turns on central cache.
   (e) Writes to (address+64KB) with the inverted longword address as data.
   (f) Turns off central cache.
   (g) Verifies that the system memory location was unmodified, since the write should have gone only to the cache (no write-through).
   (h) Verifies that the cache tag was updated correctly (valid,dirty,addr bits).
   (i) Verifies that cache data was written correctly.

   Upon error: loop through steps (a) --> (i) with constant test address. Possible error messages for this test are:

```
error1: Bad system memory after cache valid write miss:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad cache tag ram after cache valid write miss:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad cache data ram after cache valid write miss:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that shown for the Central Cache Read Hit Test.

Central Cache Write Miss, Writeback Test

This test verifies that a data operand write to system memory address (X+64KB) with a valid and dirty memory block address (X) in the cache causes a write to the cache and also a writeback to system memory. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.
2. Clears Central Cache tags, data, and the first 64 Kbytes of system memory.
3. For each test address 0x0,0x1,0x2,0x4....0x08000000 (128MB), or up to maximum memory installed, the test:

   (a) Writes the longword address as data into memory.
   (b) Writes inverted longword address as data into cache data RAM.
   (c) Writes address, valid bit ON, dirty bit ON in cache tag RAM.
   (d) Turns on central cache.
   (e) Writes to (address+64KB) with inverted longword address as data.
   (f) Turns off central cache.
   (g) Verifies that system memory location was modified, i.e. a writeback of data initially in cache ram occurred.
   (h) Verifies cache tag was updated correctly (valid,dirty,addr bits).
   (i) Verifies cache data was written correctly.

   Upon error, the test loops through steps a - i with a constant test address.

   Possible error messages for this test are:

```
error1: Bad read data on line cross cache invalid read:
testaddr=0x00000000,readaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad cache tag ram after line cross cache invalid read:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad cache data ram after line cross cache invalid read:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that shown for the Central Cache Read Hit Test.

Central Cache Line Cross Invalid Read Miss Test

This test verifies that doing a longword data operand read from system memory across a 16-byte line boundary with invalid memory block addresses in the cache causes a read from system memory and not the cache, and that both 16-byte lines are copied into the cache. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.
2. Clears Central Cache tags, data, and the first 64 Kbytes of system memory.
3. For each test address 0x0,0x1,0x2,0x4....0x08000000 (128MB), or up to maximum memory installed, the test:
   (a) Writes the longword address as data into memory for the two 16-byte lines starting at the address of interest (8 longword writes).
   (b) Writes the inverted longword address as data into cache data RAM for the two 16-byte lines starting at the address of interest (8 longword writes).
   (c) Writes the two correct addresses, valid bit OFF in cache tag RAM.
   (d) Turns on central cache.
   (e) Reads a longword from (address + 0x0e), to access data across the line boundary.
   (f) Turns off central cache.
   (g) Verifies that the data read is the same as the address, and therefore, is not read from cache, but rather from system memory.
   (h) Verifies that the cache tag is updated correctly (both entries).
   (i) Verifies that cache data is updated correctly (all 32 bytes).

   Upon error, the test loops through steps a - i with a constant test address.

   Possible error messages for this test are:

```
error1: Bad read data on line cross cache invalid read:
testaddr=0x00000000,readaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad cache tag ram after line cross cache invalid read:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad cache data ram after line cross cache invalid read:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that shown for the Central Cache Read Hit Test.

Central Cache Line Cross Write Miss Writeback Test

This test Verifies that a longword data operand write to system memory (X+64KB) across a 16-byte line boundary with a valid and dirty memory block address (X) in the cache causes a write to the cache with the new data and a writeback to memory with the old data in the cache. It verifies that both 16-byte lines are correct in memory as well as cache. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.
2. Clears Central Cache tags, data, and the first 64 Kbytes of system memory.
3. For each test address 0x0,0x1,0x2,0x4....0x08000000 (128MB), or up to maximum memory installed, the test:
   (a) Writes the longword address as data into memory for the two 16-byte lines at address X and for two lines at address (X+64KB).
   (b) Writes the inverted longword address as data into cache data RAM for the two 16-byte lines starting at the address of interest.
   (c) Writes the two correct addresses, valid bit ON and dirty bit ON in cache tag RAM.
   (d) Turns on central cache.
   (e) Writes a longword to (address+64KB+0x0e), to access data across the line boundary and to cause a writeback.
   (f) Turns off central cache.
   (g) Verifies that memory was modified through writeback of data previously in cache (all 32 bytes).
   (h) Verifies cache tag is updated correctly (both entries).
   (i) Verifies cache data is updated correctly (all 32 bytes).

   Upon error: loop through steps (a) --> (i) with constant test address. Possible error messages for this test are:

```
error1: Bad memory data on line cross cache write miss writeback:
testaddr=0x00000000,memmaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad cache tag ram after line cross cache write miss writeback:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad cache data ram after line cross cache write miss writeback:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that shown for the Central Cache Read Hit Test.

Central Cache Writeback Timeout Test

This test verifies that a data operand write to system memory address 0x0 with a valid and dirty memory block address 0x20000000 (a nonexistent memory address) in the cache causes a write to the cache and a writeback timeout due to attempting to writeback to a nonexistent memory address. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.
2. Clears Central Cache tags, data, and the first 64 Kbytes of system memory.
3. For each test address 0x0,0x1,0x2,0x4....0x08000000 (128MB), or up to maximum memory installed, the test:
   (a) Write line for addr x (test addr) with 00-0f data.
   (b) Write cache data ram line for addr x with ff-f0 data.
   (c) Write address (x+0x9000000), VALID, DIRTY in cache tag ram.
   Maximum addressable memory for Sun-3/400 is 144 MB = 0x9000000.
   (d) Turn on central cache.
   (e) Write to address x with longword 0x50515253 as data.
   (f) Turn off central cache.
   (g) Verify WBTIMOUT bit Memory Error Control Reg was set.
   (h) Verify Memory Error Addr Reg captured invalid writeback addr.
   (i) Verify that entire line at addr x was not modified.
   (j) Verify cache tag was updated correctly (valid,dirty,addr bits).
   (k) Verify cache data was written correctly.

   Upon error: loop continuously through steps a - k. Possible error messages for this test are:

```
error1: No level 7 interrupt occurred.
testaddr=0x00000000,tagramaddr=0x00000000,dataramaddr=0x00000000

error2: Memory Error Control Reg Asynch Timeout bit not set.
testaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad Mem Err Addr Reg after cache invalid writeback:
testaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that shown for the Central Cache Read Hit Test.

Block Copy (Source=Cache Miss,Dest=Cache Miss) Test

This test verifies that doing a block copy read from memory followed by a block copy write to memory with all Central Cache entries invalidated causes the proper blocks of data to be transferred in memory. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.
2. Clears Central Cache tags, data, and the first 64 Kbytes of system memory.
3. For each test address 0x0,0x1,0x2,0x4....0x08000000 (128MB), or up to maximum memory installed, the test:
   (a) Writes the line for addr x (test addr) in memory with 00-0f data.
   (b) Writes the line for addr x+0x10 in memory with ff-f0 data.
   (c) Writes the line for addr x+0x20 in memory with all zero data.
   (d) Does a block copy read from addr x.
   (e) Does a block copy write to addr x+0x10.
   (f) Verifies that the line data for addr x was unmodified.
   (g) Verifies that the line data for addr x+0x10 was modified correctly.
   (h) Verifies that the line data for addr x+0x20 was unmodified.

   Upon error, the test loops through steps a - h with a constant test address. Possible error messages for this test are:

```
error1: Bad system memory where no write expected:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad system memory where block write expected:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad system memory where no write expected:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
13	**0x0d**	**bit 7 oooo●●o● bit 0**	**okay**
13	**0x8d**	**●ooo●●o●**	**error**

Block Copy (Source=Cache Miss,Dest=Cache Hit) Test

This test verifies that doing a block copy read from memory followed by a block copy write to memory with a valid and clean Central Cache entry for the destination causes the proper blocks of data to be transferred in memory and the Central Cache block entry invalidated. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.
2. Clears Central Cache tags, data, and the first 64 Kbytes of system memory.
3. For each test address 0x0,0x1,0x2,0x4....0x08000000 (128MB), or up to maximum memory installed, the test:
   (a) Writes the line for addr x (test addr) in memory with 00-0f data.
   (b) Writes the line for addr x+0x10 in memory with ff-f0 data.
   (c) Writes the line for addr x+0x20 in memory with all zero data.
   (d) Writes the Central Cache line for addr x+0x10 with 50-5f data.
   (e) Writes the Central Cache tag for addr x+0x10 valid, clean.
   (f) Turns on Central Cache.
   (g) Does a block copy read from addr x.
   (h) Does a block copy write to addr x+0x10.
   (i) Turns off Central Cache.
   (j) Verifies that the line data for addr x was unmodified.
   (k) Verifies that the line data for addr x+0x10 was modified correctly.
   (l) Verifies that the line data for addr x+0x20 was unmodified.
   (m) Verifies that the Central Cache tag for addr x+0x10 was invalidated.

   Upon error, the test loops through steps a - m with a constant test address. Possible error messages for this test are:

```
error1: Bad system memory where no write expected:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad system memory where block write expected:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad system memory where no write expected:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error4: Central Cache tag not invalidated:
testaddr=0x00000000,tagaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that shown for the Block Copy (Source=Cache Miss,Dest=Cache Miss) Test.

Block Copy (Source=Cache Hit,Dest=Cache Miss) Test

This test verifies that doing a block copy read from memory followed by a block copy write to memory with a valid and clean Central Cache entry for the source causes the proper block of data to be transferred from Central Cache to memory and that the Central Cache block entry for the source is unchanged. The test does the following:

Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.

Clears Central Cache tags, data, and the first 64 Kbytes of system memory.

For each test address 0x0,0x10,0x20,0x40,...,0x08000000 (128MB), or up to maximum memory installed, the test:
(a) Writes the line for addr x (test addr) in memory with 00-0f data.
(b) Writes the line for addr x+0x10 in memory with 50-5f data.
(c) Writes the line for addr x+0x20 in memory with all zero data.
(d) Writes the Central cache line for addr x with ff-f0 data.
(e) Writes the Central cache line for addr x+0x10 with af-a0 data.
(f) Writes the Central cache tag for addr x valid, clean.
(g) Turns on Central Cache.
(h) Does a block copy read from addr x.
(i) Does a block copy write to addr x+0x10.
(j) Turns off Central Cache.
(k) Verifies that the line data for addr x was unmodified.
(l) Verifies that the line data for addr x+0x10 was modified correctly.
(m) Verifies that the line data for addr x+0x20 was unmodified.
(n) Verifies that the Central Cache tag for line addr x was unchanged.

Upon error, the test loops through steps a - n with a constant test address. Possible error messages for this test are:

```
error1: Bad system memory where no write expected:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad system memory where block write expected:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad system memory where no write expected:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error4: Central Cache tag UNEXPECTEDLY invalidated:
testaddr=0x00000000,tagaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
14	0x0e	bit 7 oooo●●●o bit 0	okay
14	0x8e	●ooo●●●o	error

Block Copy (Source=Cache Hit,Dest=Cache Hit) Test

This test verifies that doing a block copy read from memory followed by a block copy write to memory with the a valid and clean Central Cache entry for the source and destination causes the proper block of data to be transferred from Central Cache to memory and that the Central Cache block entry for the destination address is invalidated. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.
2. Clears Central Cache tags, data, and the first 64 Kbytes of system memory.
3. For each test address 0x0,0x10,0x20,0x40,...,0x08000000 (128MB), or up to maximum memory installed, the test:
   (a) Writes the line for addr x (test addr) in memory with 00-0f data.
   (b) Writes the line for addr x+0x10 in memory with 50-5f data.
   (c) Writes the line for addr x+0x20 in memory with all zero data.
   (d) Writes the Central cache line for addr x with ff-f0 data.
   (e) Writes the Central cache line for addr x+0x10 with af-a0 data.
   (f) Writes the Central cache tag for addr x,x+0x10 valid, clean.
   (g) Turns on Central Cache.
   (h) Does a block copy read from addr x.
   (i) Does a block copy write to addr x+0x10.
   (j) Turns off Central Cache.
   (k) Verifies that the line data for addr x was unmodified.
   (l) Verifies that the line data for addr x+0x10 was modified correctly.
   (m) Verifies that the line data for addr x+0x20 was unmodified.
   (n) Verifies that the Central Cache tag for addr x is unchanged.
   (o) Verifies that the Central Cache tag for addr x+0x10 was invalidated.

   Possible error messages for this test are:

```
error1: Bad system memory where no write expected:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad system memory where block write expected:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad system memory where no write expected:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error4: Central Cache tag UNEXPECTEDLY invalidated:
testaddr=0x00000000,tagaddr=0x00000000,exp=0x00000000,obs=0x00000000

error5: Central Cache tag not invalidated:
testaddr=0x00000000,tagaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display for this test is the same as that for the Block Copy (Source=Cache Hit,Dest=Cache Miss) Test.

Memory Write/Write/Read Read Test (Central Cache on)

This test verifies the address and data paths to Main Memory, as well as address bit and data bit uniqueness.

For each test address of main memory:
(0x0,0x1,0x2,0x4,0x8,0x10,0x20....size_of_mem_installed)

For each data pattern at each test address:
(0x0,0x1,0x2,0x4,0x10,0x20....0x80000000)
The test does the following:
(a) Writes test data to test address.
(b) Writes inverted test data to test address + 0x04.
(c) Reads back data from test address and compares.

CAUTION **THIS RAM TEST IS BEING DONE WITH CENTRAL CACHE ENABLED!**

Upon error: loop through steps a - c with constant test data and a constant test address. This test enters a scope loop on data compare errors, with the following message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
15	**0x0f**	**bit 7 oooo●●●● bit 0**	**okay**
15	**0x8f**	**●ooo●●●●**	**error**

IOC Tag RAM Write/Write/Read Test

This test performs three passes of write/write/read testing on the entire IOC Tag RAM. Each pass consists of writing a longword test pattern to the address under test, then the 1's complement of that pattern to the next long word, and finally, reading the original long word back and comparing it with what was written. The test address is then incremented by one longword and the process is repeated until the end of RAM is reached. The first pass is done with a test pattern of 0x5a972c5a, the second pass is done with a test pattern of 0x5a5a972c, and the third pass is done with a test pattern of 0x2c5a5a97.

This test enters a scope loop on data compare errors, with the following terminal message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
16	**0x10**	**bit 7 ooo●oooo bit 0**	**okay**
16	**0x90**	**●oo●oooo**	**error**

IOC Tag RAM Address Test

This test writes the complete I/O Cache Tag RAM address space with the longword address as the data, then reads back the entire address space and verifies that no addresses are overwritten. This is a test for addressing uniqueness of the I/O Cache Tag RAM.

This test enters a scope loop on data compare errors, with the following message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display is the same as that shown for the IOC Tag RAM Write/Write/Read Test.

IOC Tag RAM 3-Pattern Test

This test writes the complete I/O Cache Tag RAM address space with a repeated three long word pattern sequence, then reads back the entire address space and verifies the data.

This test enters a scope loop on data compare errors, with the following message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display is the same as that shown for the IOC Tag RAM Write/Write/Read Test.

IOC Data RAM Write/Write/Read Test

This test performs 3 passes of write/write/read testing on the entire IOC Data RAM. Each pass consists of writing a longword test pattern to the address under test, then the 1's complement of that pattern to the next long word, and finally, reading the original long word back and comparing it with what was written. The test address is then incremented by one longword and the process is repeated until the end of RAM is reached. The first pass is done with a test pattern of 0x5a972c5a, the second pass is done with a test pattern of 0x5a5a972c, and the third pass is done with a test pattern of 0x2c5a5a97.

This test enters a scope loop on data compare errors, with the following terminal message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
17	0x11	bit 7 ooo●ooo● bit 0	okay
17	0x91	●oo●ooo●	error

IOC Data RAM Address Test

This test writes the complete I/O Cache Data RAM address space with the long-word address as the data, then reads back the entire address space and verifies that no addresses are overwritten. This is a test for addressing uniqueness of the I/O Cache Data RAM.

This test enters a scope loop on data compare errors, with the following message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display is the same as that shown for the IOC Data RAM Write/Write/Read Test.

IOC Data RAM 3-Pattern Test

This test writes the complete I/O Cache Data RAM address space with a repeated three-long-word pattern sequence, then reads back the entire address space and verifies the data.

This test enters a scope loop on data compare errors, with the following message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display is the same as that shown for the IOC Data RAM Write/Write/Read Test.

IOC Data RAM Read Byte Alignment Test

This test verifies that byte, word, and long-word read operations produce the appropriate byte-aligned data. The test enters a scope loop on data compare errors, with the following terminal message:

```
byte misalignment: addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display is the same as that shown for the IOC Data RAM Write/Write/Read Test.

IOC Data RAM Write Byte Alignment Test

This test verifies that byte, word, and long word read operations produce the appropriate byte-aligned data for various byte-aligned write operations. The test enters a scope loop on data compare errors, with the following terminal message:

```
byte misalignment: addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display is the same as that shown for the IOC Data RAM Write/Write/Read Test.

VME Loopback Test

This test verifies that the VME loopback function works for writes and reads. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.
2. For each test address (0x0,0x4,0x8,0x10,0x20,0x00040000....0x00080000) and for each data pattern (0x0,0x1,0x2,,...,0x80000000) at each test address, the test:
   (a) Turns on VME-loopback in System Enable Register.
   (b) Writes the address, using DMVA offset.
   (c) Reads the address, using DVMA offset.
   (d) Reads the address again, using DVMA offset.
   (e) Turns off VME-loopback in System Enable Register.
   (f) Verifies that the data read is the same as the data that was written.

Upon error: loop through steps a - f with a constant address and data. Possible error messages for this test are:

```
error1: testaddr=0x00000000,testdata=0x00000000,exp=0x00000000,obs=0x00000000
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
18	**0x12**	**bit 7 ooo●oo●o bit 0**	**okay**
18	**0x92**	**●oo●oo●o**	**error**

VME Loopback and DVMA Test

This test is not used on a Sun-3/460 workstation.

This test verifies that the VME loopback function works for DVMA writes and reads.

The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in System Enable Register.
2. For each test address (0x0,0x4,0x8,0x10,0x20,0x00040000....0x00080000) and for each data pattern (0x0,0x1,0x2....0x80000000) at each test address, the test does the following:
   (a) Writes IO Mapper entry for test address.
   (b) Turns on DVMA, VME-loopback in System Enable Register.
   (c) Writes the address, using DMVA offset.
   (d) Reads the address, using DVMA offset.
   (e) Turns off DVMA in System Enable Register.
   (f) Reads the address again, using DVMA offset.
   (g) Turns off VME-loopback in System Enable Register.
   (h) Verifies that the data read is the same as the data that was written.

   Upon error, the test loops through steps a - h with constant addresses and data. Possible error messages for this test are:

```
error1: testaddr=0x00000000,testdata=0x00000000,exp=0x00000000,obs=0x00000000
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
19	**0x13**	**bit 7 ooo●oo●● bit 0**	**okay**
19	**0x93**	**●oo●oo●●**	**error**

IOC Read Hit Test

This test verifies that a data operand read from system memory with a valid memory block address in the IO Cache causes a read from the IO Cache and not from system memory. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and 1st 64KB of system memory.
4. For each address longword aligned address within each of 128 pages (0x0,0x4,0x8,0x10....0x1000,0x2000,0x2004....0x3000....0x000f1000). the test does the following:

   (a) Write the longword address as data into memory.
   (b) Write the inverted longword address as data into cache data ram.
   (c) Write the address, valid bit ON in cache tag ram.
   (d) Write IO Mapper entry for test address.
   (e) Turn on IO Cache, DVMA, VME-loopback in System Enable Register.
   (f) Read the address, using DVMA offset.
   (g) Turn off IO Cache, DVMA in System Enable Register.
   (h) Read the address again, using DVMA offset.
   (i) Turn off VME-loopback in System Enable Register.
   (j) Verify that the data read is the inverse of the address, and
   therefore, is not read from system memory, but rather from cache.
   (k) Verify cache tag is still correct (valid,unmodified).
   (l) Verify cache data is still correct (unmodified).

   Upon error, the test loops through steps a - l with a constant test address. Possible error messages for this test are:

```
error1: Bad read data on cache valid read:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad IOC tag ram after cache valid read:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad IOC data ram after cache valid read:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The diagram below summarizes the LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
20	**0x14**	**bit 7 ooo●o●oo bit 0**	**okay**
20	**0x94**	**●oo●o●oo**	**error**

IOC Invalid Read Miss Test

This test verifies that a data operand read from system memory with an invalid memory block address in the IO Cache causes a read from system memory and not from the IO Cache. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and 1st 64KB of system memory.
4. For each address longword aligned address within each of 128 pages (0x0,0x4,0x8,0x10....0x1000,0x2000,0x2004....0x3000....0x000f1000), the test does the following:
   (a) Writes 00-0f as data into memory for cache line of interest.
   (b) Writes ff-f0 as data into cache data ram for cache line of interest.
   (c) Writes the address, valid bit OFF in cache tag ram.
   (d) Writes IO Mapper entry for test address.
   (e) Turns on IO Cache, DVMA, VME-loopback in System Enable Register.
   (f) Reads the address, using DVMA offset.
   (g) Turns off IO Cache, DVMA in System Enable Register.
   (h) Reads the address again, using DVMA offset.
   (i) Turns off VME-loopback in System Enable Register.
   (j) Verifies that the data read is the same as the address, and therefore is not read from io cache, but rather from system memory.
   (k) Verifies cache tag is updated correctly.
   (l) Verifies cache data is updated correctly (entire line).

   Upon error: loop through steps a - l with a constant test address. Possible error messages for this test are:

```
error1: Bad read data on cache invalid read:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad IOC tag ram after cache invalid read:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad IOC data ram after cache invalid read:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Write Hit Test

This test verifies that a data operand write to system memory with a valid memory block address in the IO Cache causes a write to the IO Cache and not to system memory. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each address longword aligned address within each of 128 pages (0x0,0x4,0x8,0x10,....0x1000,0x2000,0x2004,....0x3000,...0x000f1000). the test does the following:
   (a) Write 00-0f (16 bytes) into memory for cache line of interest.
   (b) Write all zeros into cache data ram for cache line of interest.
   (c) Write the address, valid bit ON in cache tag ram.
   (d) Write IO Mapper entry for test address.
   (e) Turn on IO Cache, DVMA, VME-loopback in System Enable Register.
   (f) Write address (with DVMA offset) with inverse address as data.
   (g) Turn off IO Cache, DVMA, VME-loopback in System Enable Register.
   (h) Verify that system memory location was unmodified, since the write should have gone only to the cache (no write-through).
   (i) Verify cache tag was updated correctly (valid,dirty,addr bits).
   (j) Verify cache data was written correctly.

   Upon error: loop through steps a - j with a constant test address. Possible error messages for this test are:

```
error1: Bad system memory after cache valid write hit:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad IOC tag ram after cache valid write hit:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad IOC data ram after cache valid write hit:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Write Miss, No Writeback Test

This test verifies that a data operand write to system memory with an invalid memory block address in the IO cache causes a write to IO cache and not to system memory. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each address longword aligned address within each of 128 pages (0x0,0x4,0x8,0x10,...0x1000,0x2000,0x2004,...0x3000,...0x000f1000), the test does the following:

   (a) Writes 00-0f (16 bytes) into memory for cache line of interest.
   (b) Writes all zeros into cache data ram for cache line of interest.
   (c) Writes the address, valid bit OFF in cache tag ram.
   (d) Writes IO Mapper entry for test address.
   (e) Turns on IO Cache, DVMA, VME-loopback in System Enable Register.
   (f) Writes address (with DVMA offset) with inverse address as data.
   (g) Turns off IO Cache, DVMA, VME-loopback in System Enable Register.
   (h) Verifies that system memory location was unmodified, since the write should have gone only to the cache (no write-through).
   (i) Verifies cache tag was updated correctly (valid,dirty,addr bits).
   (j) Verifies cache data was written correctly.

   Upon error: loop through steps a - j with a constant test address. Possible error messages for this test are:

```
error1: Bad system memory after cache write miss:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad IOC tag ram after cache write miss:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad IOC data ram after cache write miss:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Write Miss, Writeback Test

This test verifies that a data operand write to system memory with a different, valid and dirty memory block address in the IO cache causes a writeback to system memory.

The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears the Central Cache tags, data, and the first 64KB of system memory.
4. For each address longword aligned address within each of 128 pages (0x0,0x4,0x8,0x10,...0x1000,0x2000,0x2004,...0x3000,...0x000f1000) the test:

(a) Writes the line for memory address x (writeback addr) with ff-f0 data.
(b) Writes the line for memory address x+0x10 with zero data.
(c) Writes the line in data cache for addr x with 00-0f data.
(d) Writes the address (x), VALID, DIRTY in cache tag RAM.
(e) Writes an IO Mapper entry for the test address.
(f) Turns on IO Cache, DVMA, VME-loopback in the System Enable Reg.
(g) Writes the address (with DVMA offset) with inverse address as data.
(h) Turns off the IO Cache, DVMA, VME-loopback in System Enable Reg.
(i) Verifies that the line for addr x was written back to memory.
(j) Verifies that system memory for location (x+0x10) was unmodified.
(k) Verifies that the cache tag was updated correctly (valid,dirty,addr bits).
(l) Verifies that cache data was written correctly.

Upon error, the test loops through steps a - l with a constant test address. Possible error messages for this test are:

```
error1: Bad system memory where writeback should have occurred:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad system memory where NO writeback should have occurred:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad IOC tag ram after cache write miss:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error4: Bad IOC data ram after cache write miss:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Read Miss, Writeback Test

This test Verifies that a data operand read from system memory with a different valid and dirty memory block address in the IO cache causes a writeback to system memory. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each address longword aligned address within each of 128 pages (0x0,0x4,0x8,0x10,...0x1000,0x2000,0x2004,...0x3000,...0x000f1000).

   the test does the following:

   (a) Writes the line for memory address x (writeback addr) with ff-f0 data.
   (b) Write the line for memory address x+0x10 with 50-5f data.
   (c) Writes the line in data cache for addr x with 00-0f data.
   (d) Writes the address (x), VALID, DIRTY in cache tag ram.
   (e) Writes the IO Mapper entry for the test address.
   (f) Turns on IO Cache, DVMA, VME-loopback in System Enable Register.
   (g) Reads from address x+0x10 (with DVMA offset).
   (h) Turns off IO Cache, DVMA in System Enable Register.
   (i) Reads from address x+0x10 (with DVMA offset).
   (j) Turns off VME-loopback in System Enable Register.
   (k) Verifies data read in was from memory, not cache.
   (l) Verifies that the line for addr x was written back to memory.
   (m) Verifies that the system memory line for addr (x+0x10) was unmodified.
   (n) Verifies that the cache tag was updated correctly (valid,addr bits).
   (o) Verifies that cache data was written correctly.

   Upon error, the test loops through steps a - o with a constant test address. Possible error messages for this test are:

```
error1: Bad read data on cache valid read miss:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad system memory where writeback should have occurred:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad system memory where NO writeback should have occurred:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error4: Bad IOC tag ram after cache valid read miss:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error5: Bad IOC data ram after cache valid read miss:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Valid Write Hit (Central Cache Match,Unmod) Test

This test Verifies that a data operand write to system memory with a valid memory block address in the IO Cache *and* in the Central Cache causes a write to the IO Cache and not to system memory, and that the entry in the Central Cache is invalidated. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA,EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each longword aligned address within each of 128 pages (0x0,0x4,0x8,0x10,...0x1000,0x2000,0x2004,...0x3000,...0x000f1000).

   the test does the following:

   (a) Writes the line for memory address x with 00-0f data.
   (b) Writes the line in IO cache for addr x with all zero data.
   (c) Writes the address (x), VALID in IO cache tag ram.
   (d) Writes the address (x), VALID in Central cache tag ram.
   (e) Writes the IO Mapper entry for the test address.
   (f) Turns on Central Cache, IO Cache, DVMA, VME-loopback in the System Enable register.
   (g) Writes to the test address (with DVMA offset) with the inverted address as data.
   (h) Turns off the Central Cache, IO Cache, DVMA, and VME-loopback in the System Enable register.
   (i) Verifies that the system memory line for addr x was unmodified.
   (j) Verifies that the IO cache tag was updated correctly (valid,dirty, address bits).
   (k) Verifies that the IO cache data was written correctly.
   (l) Verifies that the Central Cache tag was updated correctly (invalid).

Upon error, the test loops through steps a - l with a constant test address. Possible error messages for this test are:

```
error1: Bad system memory after cache valid write hit:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad IOC tag ram after cache valid write hit:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad IOC data ram after cache valid write hit:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error4: Bad Central Cache tag ram after cache valid write hit:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Invalid Write Miss (Central Cache Match,Unmod) Test

This test verifies that a data operand write to system memory with a memory block address that is INVALID in in the IO Cache and VALID in the Central Cache causes a write to IO Cache and not to system memory, and that the entry in the Central Cache is invalidated. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA, and EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each longword aligned address within each of 128 pages (0x0,0x4,0x8,0x10,...0x1000,0x2000,0x2004,...0x3000,...0x000f1000).

   the test does the following:

   (a) Writes the system memory line for addr x with 00-0f data.
   (b) Writes the line in IO cache for addr x with all zero data.
   (c) Writes the address (x), INVALID in IO cache tag ram.
   (d) Writes the address (x), VALID in Central cache tag ram.
   (e) Writes the IO Mapper entry for the test address.
   (f) Turns on Central Cache, IO Cache, DVMA, VME-loopback in the System Enable register.
   (g) Writes to the test address (with DVMA offset) with the inverted address as data.
   (h) Turns off Central Cache, IO Cache, DVMA, VME-loopback in the System Enable register.
   (i) Verifies that the system memory line for addr x was unmodified.
   (j) Verifies that the IO cache tag was updated correctly (valid,dirty,address bits).
   (k) Verifies that IO cache data was written correctly.
   (l) Verifies that Central Cache tag was updated correctly (invalid).

   Upon error, the test loops through steps a - l with a constant test address. Possible error messages for this test are:

```
error1: Bad system memory after cache invalid write miss:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad IOC tag ram after cache invalid write miss:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad IOC data ram after cache invalid write miss:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error4: Bad Central Cache tag ram after cache invalid write miss:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Invalid Read Miss (Central Cache Match,Unmod) Test

This test verifies that a data operand read from system memory with the memory block address that is INVALID in the IO Cache and VALID in the Central Cache causes a read from Central Cache and not from IO cache or system memory. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA, and EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each longword aligned address within each of 128 pages (0x0,0x4,0x8,0x10,...0x1000,0x2000,0x2004,...0x3000,...0x000f1000).

   the test does the following:

   (a) Writes the system memory line for addr x with ff-f0 data.
   (b) Writes the line in IO cache for addr x with all zero data.
   (c) Writes the address (x), INVALID in IO cache tag ram.
   (d) Writes the line in Central cache for addr x with 00-0f data.
   (e) Writes the address (x), VALID in Central cache tag ram.
   (f) Writes the IO Mapper entry for test address.
   (g) Turns on Central Cache, IO Cache, DVMA, VME-loopback in the System Enable register.
   (h) Reads from the test address (with DVMA offset).
   (i) Turns off Central Cache, IO Cache, DVMA in the System Enable register.
   (j) Reads from the test address again (with DVMA offset).
   (k) Turns off VME-loopback in the System Enable register.
   (l) Verifies that the data read in is correct (came from Central cache).
   (m) Verifies that the system memory line for addr x was unmodified.
   (n) Verifies that the IO cache tag was updated correctly (valid, clean, address bits).
   (o) Verifies that the IO cache data was written correctly.
   (p) Verifies that the Central cache tag was unchanged.

Upon error, the test loops through steps a - p with a constant test address. Possible error messages for this test are:

```
error1: Bad read data on cache invalid read:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad system memory after cache invalid read miss:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad IOC tag ram after cache invalid read miss:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error4: Bad IOC data ram after cache valid read miss:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error5: Bad Central Cache tag ram after cache invalid read miss:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Invalid Read Miss (Central Cache Match,Modified) Test

This test verifies that a data operand read from system memory with a memory block address that is INVALID in the IO Cache; VALID in the Central Cache; and MODIFIED causes a read from Central Cache and not from IO cache or system memory. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA, and EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each longword aligned address within each of 128 pages (0x0,0x4,0x8,0x10,...0x1000,0x2000,0x2004,...0x3000,...0x000f1000), the test does the following:

   (a) Writes the system memory line for addr x with ff-f0 data.
   (b) Writes the line in IO cache for addr x with all zero data.
   (c) Writes the address (x), INVALID in IO cache tag ram.
   (d) Writes the line in Central cache for addr x with 00-0f data.
   (e) Writes the address (x), VALID, DIRTY in Central cache tag ram.
   (f) Writes the IO Mapper entry for the test address.
   (g) Turns on Central Cache, IO Cache, DVMA and VME-loopback in the System Enable register.
   (h) Reads from the test address (with DVMA offset).
   (i) Turns off Central Cache, IO Cache, DVMA in the System Enable register.
   (j) Reads from the test address (with DVMA offset).
   (k) Turns off VME-loopback in System Enable register.
   (l) Verifies that the data read in is correct (came from Central cache).
   (m) Verifies that the system memory line for addr x was unmodified.
   (n) Verifies that the IO cache tag was updated correctly (valid,clean, address bits).
   (o) Verifies that IO cache data was written correctly.
   (p) Verifies that the Central cache tag was unchanged.
   (q) Verifies that the Central cache data was unchanged.

   Upon error, the test loops through steps a - q with a constant test address. Possible error messages for this test are:

```
error1: Bad read data on cache invalid read:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad system memory after cache invalid read miss:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad IOC tag ram after cache invalid read miss:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error4: Bad IOC data ram after cache valid read miss:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error5: Bad Central Cache tag ram after cache invalid read miss:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Valid Read Miss (Central Cache Match), Writeback Test

This test verifies that a data operand read from system memory with a memory block address that is VALID,MODIFIED in the IO Cache and another that is VALID in the Central Cache causes a read from Central Cache and not from IO cache or system memory. Also, a writeback of the valid and modified data in the IOC should take place. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA, and EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each longword aligned address within each of 128 pages (0x0,0x4,0x8,0x10,...0x1000,0x2000,0x2004,...0x3000,...0x000f1000), the test does the following:

   (a) Writes the system memory line for addr x with ff-f0 data.
   (b) Writes the system memory line for addr x+0x10 with all zero data.
   (c) Writes the line in IO cache for addr x with 50-5f data.
   (d) Writes the line in Central cache for addr x with 00-0f data.
   (e) Writes the address (x+0x10),VALID,DIRTY in IO Cache tag RAM.
   (f) Writes the address (x), VALID in Central Cache tag RAM.
   (g) Writes the IO Mapper entry for test address.
   (h) Turns on Central Cache, IO Cache, DVMA and VME-loopback in the System Enable register.
   (i) Reads from test address x (with DVMA offset).
   (j) Turns off Central Cache, IO Cache and DVMA in the System Enable registe
   (k) Reads from test address x (with DVMA offset).
   (l) Turns off VME-loopback in the System Enable register.
   (m) Verifies that data read in is correct (came from Central cache).
   (n) Verifies that the system memory line for addr x was unmodified.
   (o) Verifies that a writeback took place to system memory address x+0x10.
   (p) Verifies that the IO cache tag was updated correctly (valid,clean,address bit
   (q) Verifies that the IO cache data was written correctly.
   (r) Verifies that the Central cache tag was unchanged.
   (s) Verifies that the Central cache data was unchanged.

Upon error, the test loop through steps a - s with a constant test address. Possible error messages for this test are:

```
error1: Bad read data on IOC valid read miss:
testaddr=0x00000000,readaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad system memory where NO writeback expected:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad system memory where WRITEBACK expected:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error4: Bad IOC tag ram after IOC valid read miss:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error5: Bad IOC data ram after IOC valid read miss:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error6: Bad Central Cache tag ram after IOC valid read miss:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error7: Bad Central Cache data ram after IOC valid read miss:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Flush (Valid, Modified) Test

This test verifies that a flush cache block write with valid and dirty data in the IO cache causes the line to be written back to memory and the entry in the IO cache to be invalidated. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA, and EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each line base address within each of 128 pages (0x0,0x4,0x8,0x10,...0x1000,0x2000,0x2004,...0x3000,...0x000f1000), the test does the following:

   (a) Writes the line for memory address x with 00-0f data.
   (b) Writes the line in data cache for addr x with ff-f0 data.
   (c) Writes the address (x), VALID, DIRTY in cache tag RAM.
   (d) Writes IO Mapper entry for test address.
   (e) Turns on IO Cache, DVMA, VME-loopback in the System Enable register.
   (f) Writes to the flush block addr for the test address.
   (g) Turns off IO Cache, DVMA and VME-loopback in the System Enable regis
   (h) Verifies that the line for addr x was written back to memory.
   (i) Verifies that the IOC tag was updated correctly (invalid).
   (j) Verifies that the IOC data RAM remained unchanged.

   Upon error, the test loops through steps a - j with a constant test address. Possible error messages for this test are:

```
error1: Bad system memory where writeback should have occurred:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad IOC tag ram after IOC block flush:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad IOC data ram after IOC block flush:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Flush (Valid, Not Modified) Test

This test verifies that a flush cache block write with valid and clean data in the IO cache causes the line NOT to be written back to memory and the IO cache entry to be invalidated. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA, and EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each line base address within each of 128 pages (0x0,0x4,0x8,0x10,...0x1000,0x2000,0x2004,...0x3000,...0x000f1000), the test does the following:
   (a) Writes the line for memory address x with 00-0f data.
   (b) Writes the line in data cache for addr x with ff-f0 data.
   (c) Writes the address (x), VALID, in cache tag ram.
   (d) Writes the IO Mapper entry for the test address.
   (e) Turns on the IO Cache, DVMA, VME-loopback in the System Enable register.
   (f) Writes to the flush block address for the test address.
   (g) Turns off the IO Cache, DVMA and VME-loopback in the System Enable register.
   (h) Verifies that the line for address x was NOT written back to memory.
   (i) Verifies that the cache tag was updated correctly (invalid).
   (j) Verifies that IOC data RAM remained unchanged.

   Upon error, the test loops through steps a - j with a constant test address. Possible error messages for this test are:

```
error1: Bad system memory where NO writeback should have occurred:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad IOC tag ram after IOC block flush:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad IOC data ram after IOC block flush:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Flush (Not Valid, Not Modified) Test

This test verifies that a flush cache block write with invalid data in the IO cache causes the NOT to be written back to memory and the entry in the IO cache to remain invalidated. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA, and EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each line base address within each of 128 pages (0x0,0x4,0x8,0x10,...0x1000,0x2000,0x2004,...0x3000,...0x000f1000), the test does the following:
   (a) Writes the line for memory address x with 00-0f data.
   (b) Writes the line in data cache for addr x with ff-f0 data.
   (c) Writes the address (x), INVALID, in cache tag ram.
   (d) Writes the IO Mapper entry for the test address.
   (e) Turns on the IO Cache, DVMA and VME-loopback in the System Enable register.
   (f) Writes to the flush block address for the test address.
   (g) Turns off the IO Cache, DVMA and VME-loopback in the System Enable register.
   (h) Verifies that the line for addr x was NOT written back to memory.
   (i) Verifies that the cache tag remained invalid.
   (j) Verifies that IOC data RAM remained unchanged.

   Upon error, the test loops through steps a - j with a constant test address. Possible error messages for this test are:

```
error1: Bad system memory where NO writeback should have occurred:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad IOC tag ram after IOC block flush:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad IOC data ram after IOC block flush:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

I/O Mapper Invalid Page (IO.DT) Test

This test verifies that a data operand read from system memory with the memory block address in the IO Cache INVALID and the IO mapper entry for that page set to INVALID causes an access violation, and that there is no data cached from system memory. For each of (IO.DT = 00, 10, 11), the test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA, and EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each longword aligned address within each of 128 pages (0x0,0x4,0x8,0x10,...0x1000,0x2000,0x2004,...0x3000,...0x000f1000), the test does the following:

   (a) Writes ff-f0 as data into memory for cache line of interest.
   (b) Writes 00-0f as data into cache data RAM for cache line of interest.
   (c) Writes the address, valid bit OFF in cache tag ram.
   (d) Writes the IO Mapper entry for the test address; sets INVALID page.
   (e) Turns on IO Cache, DVMA and VME-loopback in the System Enable register.
   (f) Reads the address, using DVMA offset.
   (g) Turns off IO Cache and DVMA in the System Enable register.
   (h) Reads the address, using DMVA offset.
   (i) Turns off the VME-loopback in System Enable register.
   (j) Verifies that the cache tag is unchanged.
   (k) Verifies that the cache data is unchanged.

   Upon error, the test loops through steps a - k with a constant test address. Possible error messages for this test are:

```
error1: Bad IOC tag ram after cache invalid (page invalid) read:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad IOC data ram after cache invalid (page invalid) read:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Write Miss, Writeback (Write Protect) Test

This test verifies that a data operand write to system memory with a different memory block address in the IO cache valid and dirty and the IO mapper entry for the block's page set to Write Protect (IO.WP = 1) causes NO writeback to system memory. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA, and EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each longword aligned address within each of 128 pages (0x0,0x4,0x8,0x10,...0x1000,0x2000,0x2004,...0x3000,...0x000f1000), the test does the following:

   (a) Turns off IO cache.
   (b) Writes the line for memory address x (writeback addr) with ff-f0 data.
   (c) Writes the line for memory address x+0x10 with 50-5f data.
   (d) Writes the line in data cache for addr x with 00-0f data.
   (e) Writes the address (x), VALID, DIRTY in cache tag RAM.
   (f) Writes the IO Mapper entry for the test address.
   (g) Turns on the IO Cache, DVMA and VME-loopback in the System Enable register.
   (h) Writes the address (with DVMA offset) with inverse address as data.
   (i) Turns off IO Cache, DVMA and VME-loopback in the System Enable register.
   (j) Verifies that the line for addr x was NOT written back to memory.
   (k) Verifies that system memory for location (x+0x10) was unmodified.
   (l) Verifies that the cache tag was NOT updated.
   (m) Verifies that cache data was NOT updated.

   Upon error, the test loops through steps b - m with a constant test address. Possible error messages for this test are:

```
error1: Bad system memory where NO writeback should occur (write protect):
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad system memory where no write should have occurred (write protect):
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error3: Bad IOC tag ram after cache write miss (write protect):
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error4: Bad IOC data ram after cache write miss (write protect):
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Invalid Read Miss (IO Mapper IO.EN = 0) Test

This test verifies that a data operand read from system memory with an invalid memory block address in the IO Cache *and* the IO.EN bit in the IO Mapper entry for this page set OFF causes a read from system memory and not from the IO Cache, but that nothing gets modified in the IOC data and tags. The test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA, and EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each longword aligned address within each of 128 pages (0x0,0x4,0x8,0x10,...0x1000,0x2000,0x2004,...0x3000,...0x000f1000), the test does the following:

   (a) Writes 00-0f as data into memory for cache line of interest.
   (b) Writes ff-f0 as data into cache data RAM for cache line of interest.
   (c) Writes the address, valid bit OFF in cache tag RAM.
   (d) Writes the IO Mapper entry for the test address, IO.EN = 0.
   (e) Turns on IO Cache, DVMA and VME-loopback in the System Enable register.
   (f) Reads the address, using DVMA offset.
   (g) Turns off IO Cache and DVMA in the System Enable register.
   (h) Reads the address again, using DMVA offset.
   (i) Turns off VME-loopback in the System Enable register.
   (j) Verifies that the data read is the same as the address, and therefore is not read from IO Cache, but rather from system memory.
   (k) Verifies that the IOC tag is unchanged (still invalid).

   Upon error, the test loops through steps a - k with a constant test address. Possible error messages for this test are:

```
error1: Bad read data on cache invalid read:
testaddr=0x00000000,dataramaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad IOC tag ram after cache invalid read:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Write Miss (IO Mapper IO.EN = 0) Test

This test verifies that a data operand write to system memory with an invalid memory block address in the IO cache *and* the IO.EN bit in the IO Mapper entry for this page set OFF causes a write to system memory and not to the IOC. This test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA, and EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each longword aligned address within each of 128 pages (0x0,0x4,0x8,0x10,...0x1000,0x2000,0x2004,...0x3000,...0x000f1000), the test does the following:

   (a) Write all zeroes into memory line for test address.
   (b) Write 00-0f data into IOC data ram for cache line of interest.
   (c) Write the address, valid bit OFF in cache tag ram.
   (d) Write IO Mapper entry for test address; IO.EN = 0.
   (e) Turn on IO Cache, DVMA, VME-loopback in System Enable reg.
   (f) Write address (with DVMA offset) with inverse address as data.
   (g) Turn off IO Cache, DVMA, VME-loopback in System Enable reg.
   (h) Verifies that system memory location was modified, since the write should have gone to memory (cache disabled for this page).
   (i) Verifies that the cache tag was not modified (still invalid).

   Upon error, the test loops through steps a - i with a constant test address. Possible error messages for this test are:

```
error1: Bad system memory after cache write miss:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000

error2: Bad IOC tag ram after cache write miss:
testaddr=0x00000000,tagramaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Random Data Block Write Test

This test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA, and EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each (x) test address 0x0,0x4,0x8,0x10,0x20,0x40,0x80,...0x100000, the test:
   (a) Clears out all IOC tags, data.
   (b) Clears out the entire IO Mapper.
   (c) Writes eight IO Mapper entries for the 64MB space to be tested (x-->x+64MB with address,IO.DT,IO.EN.
   (d) Turns on IO Cache, DVMA and VME-loopback in the System Enable register.
   (e) Copies the entire 64KB EPROM contents to memory starting at the test address, doing DVMA writes through the IOC, flushing the last line of each 8K block.
   (f) Turns off the IO Cache, DVMA and VME-loopback in the System Enable register.
   (g) Verifies that 64KB of system memory matches EPROM contents.

Upon error, the test loops through step (g) with a constant test address. Possible error messages for this test are:

```
error1: Bad system memory after block write:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Random Data Block Read (Central Cache off) Test

This test does the following:

1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA, and EN_VME_LOOP in the System Enable Register.
2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each (x) test address 0x0,0x4,0x8,0x10,0x20,0x40,0x80,...0x100000, the test:

```
(a) Clears out all IOC tags, data.
(b) Clears out entire IO Mapper.
(c) Writes eight IO Mapper entries for 64MB space to be tested (x-->x+64MB),
    with address,IO.DT,IO.EN.
(d) Copies the entire 64KB EPROM contents to memory, starting at the test address.
(e) Turns on IO Cache, DVMA and VME-loopback in the System Enable register.
(f) Reads 64KB of data back from memory by way of DVMA; reads one longword at
    a time and compares with contents of EPROM.
(g) Turns off IO Cache, DVMA and VME-loopback in the System Enable register.
```

Upon error, the test loops through step (f) with a constant test address. Possible error messages for this test are:

```
error1: Bad IOC read data:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

IOC Random Data Block Read (Central Cache on) Test

This test does the following: IP 1. Turns off EN_CACHE,EN_IOCACHE,EN_DVMA, and EN_VME_LOOP in the System Enable Register.

2. Clears IOC data, tag RAM.
3. Clears Central Cache tags, data, and the first 64KB of system memory.
4. For each (x) test address 0x0,0x4,0x8,0x10,0x20,0x40,0x80,...0x100000, the test:

```
(a) Clears out all IOC tags, data.
(b) Clears out entire IO Mapper.
(c) Writes eight IO Mapper entries for 64MB space to be tested (x-->x+64MB),
    with address,IO.DT,IO.EN.
(d) Copies the entire 64KB EPROM contents to memory, starting at test address.
(e) Turns on IO Cache, DVMA, VME-loopback, Central Cache in SYSENREG.
(f) Reads 64KB of data back from memory, using DVMA, reading one longword at
    a time and comparing with EPROM contents.
(g) Turns off IO Cache, DVMA, VME-loopback and Central Cache in the
    System Enable register.
```

Upon error, the test loops through step (f) with a constant test address. Possible error messages for this test are:

```
error1: Bad IOC read data:
testaddr=0x00000000,memaddr=0x00000000,exp=0x00000000,obs=0x00000000
```

The LED display is the same as that for the IOC Read Hit Test.

P4 Overlay Frame Buffer Write/Write/Read Test

This test will be executed only if a P4 Video RAM board is detected. First, the test probes for a P4 board, with the following possible messages:

```
<P4 High Resolution Monochrome Video RAM Board Detected>

<P4 Low Resolution Monochrome Video RAM Board Detected>

<P4 Low Res Color Video RAM Board Detected>

<ILLEGAL P4 Video RAM Board ID Detected: 0x00000000>

<P4 Video RAM Board NOT Detected>
```

The same test is executed if either the monochrome or color board is detected. The test verifies the address and data paths to the P4 Overlay Frame Buffer, as well as address bit and data bit uniqueness.

For each test address of P4 Overlay Frame Buffer, the test does the following, for each data pattern at each test address (0x0,0x1,0x2,0x4,0x10,0x20,...0x80000000):
(a) Writes test data to test address.
(b) Writes inverted test data to test address + 0x04.
(c) Reads back data from test address and compares.

Upon error, the test loop through steps a - c with constant test data and a constant test address.

This test enters a scope loop on data compare errors, with the following message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED indications are as follows:

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
21	**0x15**	**bit 7 ooo•o•o• bit 0**	**okay**
21	**0x95**	**•oo•o•o•**	**error**

P4 Overlay Frame Buffer Address Test

Note that this test is only present in the diagnostic PROM code if the `P4VIDEORAM_ADDR_TEST` compile-time flag is on in the Makefile that generates the diagnostic executable code.

The same test is executed if either the monochrome or color board is detected.

This test writes the complete P4 Overlay Frame Buffer address space with the longword address as data, then reads back the entire address space and verifies that no addresses are overwritten. This is a test for addressing uniqueness of the P4 Overlay Frame Buffer RAM.

This test enters a scope loop on data compare errors, with the following message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display is the same as that for the P4 Overlay Frame Buffer Write/Write/Read Test.

P4 Overlay Frame Buffer 3-Pattern Test

Note that this test is only present in the diagnostic PROM code if the `P4VIDEORAM_3PATT_TEST` compile-time flag is on in the Makefile that generates the diagnostic executable code.

The same test is executed if either the monochrome or color board is detected. This test writes the complete P4 Overlay Frame Buffer address space with a repeated three-long-word pattern sequence, then reads back the entire address space and verifies the data.

This test enters a scope loop on data compare errors, with the following message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display is the same as that for the P4 Overlay Frame Buffer Write/Write/Read Test.

P4 Overlay Frame Buffer March Test

Note that this test is only present in the diagnostic PROM code if the `P4VIDEORAM_MARCH_TEST` compile-time flag is on in the Makefile that generates the diagnostic executable code.

The same test is executed if either the monochrome or color board is detected. A background of longword 0's is written to all of the P4 Overlay Frame Buffer RAM, starting from address 0 and ending at the highest address. Then the longword data is read at the first address and a 0xffffffff is written into this address. The same two-step read/write procedure is continued at each sequential longword until the end of memory is reached. Then each longword is tested and changed back to zero in reverse order until the first address is reached. Finally, the same sequence is repeated using complemented data (i.e. a background of 1's is written into memory).

Upon error, the test loops on a complete write then read of Video RAM. This test enters a scope loop on data compare errors, with the following message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display is the same as that for the P4 Overlay Frame Buffer Write/Write/Read Test.

P4 Overlay Frame Buffer Read Byte Alignment Test

The same test is executed if either the monochrome or color board is detected. This test verifies that byte, word, and long word read operations produce the appropriate byte-aligned data.

This test enters a scope loop on data compare errors, with the following message:

```
byte misalignment: addr xxxxxxxx, exp xxxxxxxx, obs xxxxxx
```

The LED display is the same as that for the P4 Overlay Frame Buffer Write/Write/Read Test.

P4 Overlay Frame Buffer Write Byte Alignment Test

The same test is executed if either the monochrome or color board is detected. This test verifies that byte, word, and long word read operations produce the appropriate byte-aligned data for various byte-aligned write operations.

This test enters a scope loop on data compare errors, with the following message:

```
byte misalignment: addr xxxxxxxx, exp xxxxxxxx, obs xxxxxx
```

The LED display is the same as that for the P4 Overlay Frame Buffer Write/Write/Read Test.

P4 Enable Plane Write/Write/Read Test

This test will be executed only if a P4 Low Res Color Video RAM board is detected. The test verifies the address and data paths to the P4 Enable Plane RAM, as well as address bit and data bit uniqueness. For each test address of P4 Enable Plane
(base+0x0,base+0x01,base+0x02,base+0x04,base+0x08,...,base+ram_size), the test does the following:
For each data pattern at each test address:
(0x0,0x1,0x2,0x4,0x10,0x20,...0x80000000)

(a) Writes test data to the test address.
(b) Writes the inverted test data to the test address + 0x04.
(c) Reads the data back from the test address and compares.

Upon error, the test loops through steps a - c with constant test data and a constant test address. This test enters a scope loop on data compare errors, with the following message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display is the same as that for the P4 Overlay Frame Buffer Write/Write/Read Test.

P4 Color Plane Write/Write/Read Test

This test will be executed only if a P4 Low Res Color Video RAM board is detected. The test verifies the address and data paths to the P4 Color Plane RAM, as well as address bit and data bit uniqueness. For each test address of P4 Color Plane
(base+0x0,base+0x01,base+0x02,base+0x04,base+0x08,...,base+ram_size)

the test does the following:

For each data pattern at each test address:
0x0,0x1,0x2,0x4,0x10,0x20,...0x80000000), the test

(a) Writes test data to the test address.
(b) Write inverted test data to the test address + 0x04.
(c) Reads the data back from the test address and compares.

Upon error, the test loops through steps a - c with constant test data and a constant test address. This test enters a scope loop on data compare errors, with the following message:

```
addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The LED display is the same as that for the P4 Overlay Frame Buffer Write/Write/Read Test.

Printer Controller Check — Sun-3/80

The *Printer Controller Check* tests the ability of the processor to access the Parallel Printer Controller registers.

An incrementing pattern is displayed on the data output bits at the parallel printer connector. Each pattern is read back through the internal registers and verified.

If an error is detected, the test will enter a scope-loop, writing and reading the failed pattern. Only the first error will be reported to the serial port.

```
Printer Controller Data Error Expected xxxxxxxx, OBS xxxxxxxx
```

Clock/Calendar Device — 3/80 Only

This test verifies that the Clock/Calendar device is operational. The test is in several phases:

1. It verifies that the battery is not low:

```
WARNING: The Clock/Calendar Battery is low.
```

2. It verifies that the seconds counter is updating (the clock is running). If the seconds counter is not updating, the device is reinitialized and its date and time set to January 1, 1989 00:00:00. Two warning messages may be displayed that indicate an incorrectly initialized device or possibly a failed device.

```
<Clock is not running!, re-starting TOD Clock Module.>

<New TOD Module installed, initializing and starting the C
```

If an error is detected, the test will enter a scope-loop repeating the test sequence that failed. Only the first error will be reported to the serial terminal.

```
Time of Day Clock FAILED, Unit or support logic BAD
```

Configuration Memory Check — 3/80 Only

This test verifies the operation of the Configuration memory.

The Diagnostic Test locations in CMOS are tested and any errors detected are reported.

If an error is detected the test will enter a scope-loop, writing and reading the failed pattern. Only the first error will be reported to the serial port.

```
Data Path error Expected [xxxxxxxx] Found [xxxxxxxx]

Configuration RAM ERROR: Address uniqueness Fault
```

LANCE Ethernet Controller Check — 3/80 only

This test verifies the ability of the processor to access the controller's internal registers.

```
LANCE Controller BAD, unable to access.
```

ESP SCSI Check — 3/80 Only)

The ESP Controller Check includes the ability to access the ESP registers and the operation of the internal state-machine.

If an error is detected the test will enter a scope-loop repeating the test sequence that failed. Only the first error will be reported to the serial terminal.

```
ESP Controller Bad, FLAGS xxxxxxxxx FIFO xxxxxxxxx [xxxxxxx
```

Host System Initialization

After the Sun-3 firmware has executed the power-up sequence, it initializes the host system. Devices initialized at this time are the TIA, TIB, Page Tables, PMMU, keyboard, mouse, serial ports, frame buffer, memory, and other CPU devices. Initialization tasks include memory sizing, interrupt vector setting, trap vector setting, and setting entry points that correspond to support routines.

The Sun-3 PROM Monitor

This section describes the PROM monitor (sometimes called the ''system monitor'') commands available on Sun-3 workstation. For those with Sun-3/400 series or Sun-3/80 workstation, it is intended to replace Chapter 8 of the *PROM User's Manual.*

Taken as a whole, the monitor commands offer a low-level user interface to the Sun hardware. They control a variety of options, including booting from an alternate device, changing the console output, reading or altering registers or memory locations, and so on.

The effects of most monitor commands disappear when the power is turned off, with the exception of the **q** command, which programs the EEPROM. Parameters entered with the **q** command remain until deliberately altered with another

PROM command. Programming the EEPROM *reconfigures* your workstation. The Boot PROM consults the EEPROM to determine whether or not to poll for a boot device, whether to boot a specified program, which device is the console, and so on. Refer to the **q** command in this chapter, for information on programming the EEPROM. Also refer to the `eeprom` command described in the *SunOS Reference Manual*

Bringing up the PROM Monitor

Read *Chapter 3* of the *PROM User's Manual* for instructions on bringing up the PROM monitor.

Conventions

The paragraphs below describe each of the monitor commands in detail. Each paragraph starts with a line describing the command syntax. If a command has more than one distinct format, each one is shown on a separate line. Characters in **`bold courier`** font mean that you should enter them exactly as shown. Plain `courier` font represents what you should see on the screen or a program or path name. Words in *Roman italic* show the type of information you are to enter (variables), or list document names. *Roman italic* or **boldfaced** font is also used for notes or emphasis within the text. Optional arguments and default values are listed in the descriptions.

Monitor Command Overview

The following example shows the menu that comes up when you enter **h** (help) at the monitor prompt. This section describes the general characteristics that all of the commands have in common. Detailed descriptions of each command are listed in the next section. Some commands are available only for Sun-3/400 series workstations, and those will be identified.

NOTE The **i**, **j**, **n**, *and* **y** *commands are only available on systems with on-board cache memory, as indicated in italics.*

Figure 6 *Sun-3 Monitor Help Menu*

```
Boot PROM Monitor Commands
-----------------------------------------------------------------
a [digit]                                |Open CPU Addr Reg (0-7)
b [dev([cntrl],[unit],[part])]           |Boot a file
c [addr]                                 |Continue program at Addr
d [digit]                                |Open CPU Data Reg (0-7)
e [addr]                                 |Open Addr as 16 bit word
f beg_addr end_addr pattn [size]         |Fill Memory
g [addr]                                 |Go to Addr
h or ?                                   |Help Menu
i [c] [i] [addr](Sun-3/470,3/260)        |Open Central or I/O Cache Data
j [c] [i] [addr](Sun-3/470,3/260)        |Open Central or I/O Cache Tags
k [number]                               |Reset (0)CPU, (1)MMU, (2)System
l [addr]                                 |Open Addr as 32 bit long
m [addr]                                 |Open Segment Map
m [A] [B] [addr] (Sun-3/470,3/80)           |Display TIA or TIB Table Entries
n [i/e/d]  (not for Sun-3/470)            |Cache Invalidate/Enable/Disable
o [addr]                                 |Open Addr as 8 bit byte
p [addr]                                 |Open Page Map
p [addr]         (for Sun-3/470)          |Display Page Table (For Page Or I/O Map
q [addr]                                 |Open EEPROM
r                                        |Open CPU/MMU Registers (i.e. PC)
s [digit]                                |Set/Query Function Code (0-7)
t [y/n/c]                                |Trace: Yes/No/Continue
u [arg]                                  |Select Console Device
v beg_addr end_addr [size]               |Display Memory
w [addr] [string]                        |Vector
x                                        |Extended Diag Tests
y [c cxt] [s cxt sg_addr] [p cxt pg_adr] |Flush Cntxt/Seg/Page (not for Sun-3/470 or 3/80)
z [addr]                                          |Set Breakpoint
^a  (Sun-3/80                             |Display physical/virtual address
^f  (Sun-3/80)                           |Flush ATC Cache
^r  (Sun-3/80)                           |Read on-board device registers
-----------------------------------------------------------------
```

Executing a Command

In general, to execute a command, you type the appropriate command letter, followed by any required command arguments. For example, if you wanted to execute the hypothetical command "θ" (which we will say needs two arguments `100` and `200`) you would type a line like this:

```
> θ 100 200
```

The command letter can be upper or lower case, and all commands and arguments are separated by white-space (tabs or spaces). Pressing the return key executes the command.

Default Values

Many of the monitor commands have built in, or *default* values, which the command uses if you do not supply arguments. The default values vary from command to command. Check the command descriptions for the default values of interest.

Word Sizes

Word sizes referred to in this chapter are defined as follows: A **byte** is eight bits long; a **word** is 16 bits long; a **long word** is 32 bits long.

The Monitor Commands

This section provides more detailed information on the commands listed in the help menu example. Both the commands and their arguments are described here.

Displaying and Modifying Memory

A number of the commands listed here may be used to display and/or modify the system's memory and registers. Regardless of the type of memory (RAM, EEPROM, etc.) or register, these commands have the same command syntax. This section describes the memory modification syntax used by the monitor commands. The example here references long words (32 bits) of memory, but the syntax is the same for bytes (8 bits) and words (16 bits).

The *address* argument specifies the initial memory location that is to be displayed and/or modified.

The second argument determines whether the current content of a memory location is to be displayed and/or modified. Entering *only* the address of a memory location after the command *displays* the content of that address.

```
>command FFE80000 [Return]
FFE80000: 12345678 ?
```

At this point, you can respond in one of *three* different ways.

1. Simply pressing the [Return] key displays the contents of the next memory location (in this case, 0xFFE80004) as shown below.

```
>command FFE80000 [Return]

FFE80000: 12345678 ?  [Return]

FFE80004: 00000001 ?
```

   Successively pressing the [Return] key displays the contents of successive memory locations. Assuming that you pressed [Return] four times, the contents of memory locations 0xFFE80004, 0xFFE80008, 0xFFE8000C and 0xFFE80010 would be displayed.

2. To exit the command, enter any *non-hexadecimal* character (q for quit, for example) before pressing [Return] Note that you will now return to the monitor's basic command level, with the > prompt.

```
>command FFE80000 [Return]

FFE80000: 12345678 ? q [Return]

>
```

3. The *third* response to the display of memory location contents is to *modify* those contents. You enter the *new* hexadecimal value immediately following the question mark ?, BEFORE pressing [Return]. The following display demonstrates how the value of memory location 0xFFE80000 can be changed from "12345678" to "00ABCDEF".

NOTE *Following the assignment of the new value to memory location 0xFFE80000, the new value will not be displayed; instead, the contents of the next memory location are shown. You will not be returned to the monitor's basic command level.*

```
>command FFE80000 [Return]

FFE80000: 12345678 ? 00ABCDEF [Return]

FFE80004: 00000001 ?
```

Entering a memory location's virtual address followed *only* by a non-hexadecimal character before pressing the [Return] key causes the monitor to *display* the contents of that location then exit to the monitor command level.

```
>command FFE80004 q [Return]
FFE80004: 00000001
>
```

Entering a non-hexadecimal character *and* a hexadecimal value after an address causes the monitor to display the original value at that virtual address, assign a new value, then display that value. For instance, assume that the original content at address 0xFFE80010 is "00000005". The command

```
>command FFE80010 ? 55555550 [Return]
```

displays the original value "00000005" then assigns the value "55555550" to address FFE80010 before returning to the monitor prompt:

```
>command FFE80010 ? 55555550 [Return]
FFE80010: 00000005 -> 55555550
>
```

You may also enter multiple display and/or modify commands for multiple memory locations on the same command line. If you enter this command line:

```
>command FFE80000 ? 00000000 ? ? 22222220 33333330 q Return
```

You will see this on the screen:

```
FFE80000: 12345678 -> 00000000
FFE80004: 00000001
FFE80008: 00000002 -> 22222220
FFE8000C -> 33333330
FFE80010: 00000004
>
```

The first part of the command line,

>*command* **FFE80000 ? 00000000**

displays the original contents of location FFE80000 before assigning the new value "00000000" to it.

The next question mark directs the monitor to display the contents of FFE80004. The next part of the command line, **? 22222220**, tells the monitor to display the original contents of FFE80008, before assigning the new value "22222220" to it. The **33333330** tells the monitor to assign the value "33333330" to memory location FFE8000C. Finally, the **q** causes the monitor to exit to the command level after the contents of FFE80010 are displayed.

Special Monitor Commands

Address Increment/Decrement Command

By preceding the command with a + or −, you can cause the address display to increment or decrement to the next location.

While traversing a range of addresses, type a + or − to change direction after the program displays the content and waits for input.
For example:

```
>l 0 Return

00000000 00000000 ?  Return

00000004 00000001 ?  Return

00000008 00000002 ? - (you enter a minus sign) Return

00000004 00000001 ?  Return (contents of previous location are displayed)

00000000 00000000 ? + Return (after decrement, you enter a plus sign)

00000004 00000001 ?  Return you increment again

00000008 00000002 ?  Return (you increment again)
```

The `^r` Command — Sun-3/400 series only

A set of commands that are prefaced with the caret (^) character access the second level of the PROM monitor command menu.

Entering the ^ character and then the `r` key displays the following MC68030 (Sun-3/400 series) registers:

The Translation Control Register in the PMMU

The CPU Root Pointer (both Status Long Word and Table Address)

The System Enable Register

The Interrupt Register

The Bus Error Register

Viewing these registers gives a quick view of the current CPU status as opposed to the monitor `r` command, which displays status that was stored during an exception error.

The `^t` Command

Entering the ^ character and then the **`t`** key, followed by a virtual address, displays the physical address to which that address is mapped, along with a detailed description of all the bits in the page table entry, the segment and page RAM addresses, and what space they are in.

For example, entering

>^t 1000 Return

results in this display:

```
Virtual Addr 1000 is mapped to Physical Addr 1000
Context = 0x0, Seg Map = 0x0, Page Map = 0xC0000000.
Page 0 has these attributes:

       Valid bit  = 0
       Permission = 1
       No Cache   = 0
       Type       = 0
       Access bit = 0
       Modify bit = 0
```

Entering **^t** with no arguments provides information with reference to virtual address 0x00.

The ^i Command

Entering the ^ character and then the **i** key, followed with a command, displays compilation information for the system firmware. It includes the date, host name and build directory path. For example:

`Compiled at 6/7/87 on` *hostname* `in` */directory_name*

The ^c Command

^c *source destination n*

Entering the ^ character and then the **c** key, followed with the parameters shown, causes a block of *n* length to be copied from *source* to *destination* address, byte by byte. There is enough delay to copy to EEPROM also.

The ! Command

Entering an exclamation point executes the last monitor command you entered again.

Regular Monitor Commands

Monitor a Command

a *register_number action*

The a command provides access to the CPU's address registers. *register_number* may be a value from 0 to 7 inclusive. The default value is 0. Register_number 7 accesses A7, the system stack pointer. To see the user stack pointer, use the **r** command.

It is important to note that it is *not* possible to display the state of the processor at all times. In particular, processor state is only observable after:

- An unexpected trap
- A user program has "dropped" into the monitor (by calling monitor function `abortent`).
- You have manually "broken" into the monitor mode (by typing the `L1-a` sequence on a Sun keyboard or Break on a dumb terminal's keyboard).

Read the section entitled *Displaying and Modifying Memory* for details on how to use this command. Replace the word *command* in the description with the letter `a`.

Monitor **A** Command—Sun-3/400 Series

For the Sun-3/400 series, the uppercase **A** command displays these physical and virtual addresses:

Device	*Virtual Address:*	*Physical Address:*
Keybrd/Mouse	0x:fef00000	0x:62000000
Serial Port	0x:fef02000	0x:62002000
EEPROM	0x:fef04000	0x:64000000
TOD	0x:fef06000	0x:64002000
Mem Err Reg	0x:fef09000	0x:61001000
Int Reg	0x:fef0b400	0x:61001400
P4 DAC	0x:fef0c000	0x:50200000
ie Ethernet	0x:fef08000	0x:65000000
ECC ENA Reg	0x:fef14000	0x:6a1e0000
Sys ENA Reg	0x:fef16000	0x:61000000
Bus ERR Reg	0x:fef18400	0x:61000400
IDPROM	0x:fef1cc00	0x:61000c00
P4 Video Reg	0x:fef1e000	0x:50300000
VIDEOMEM_BASE	0x:fef20000	0x:50400000
P4 Overlay	0x:fef20000	0x:50400000
P4 ENA Plane	0x:fef40000	0x:50600000
IOMAPPER	0x:fef66000	0x:60000000
CACHETAGS	0x:fefc0000	0x:68000000
CACHEDATA	0x:fefd0000	0x:69000000
IO CACHE TAGS	0x:fef6c000	0x:6c000000
IO CACHE DATA	0x:fef6e000	0x:6c002000

Monitor **b** Command

b ? or **b**! *boot_device path argument_list*

The `boot` command loads and executes the SunOS operating system, an EEPROM-specified program, or a user-specified program. The boot program can be loaded from the default device, the device specified in the EEPROM, or the boot device specified in the command argument. A *boot_device* is a secondary storage device (disk, Ethernet or tape) that contains the program to be loaded and executed.

If the diagnostic switch on the back of the system is in the NORM position, the value in EEPROM address 0x18 is *not* equal to 0x12, and the boot command is entered without arguments, the system will boot the SunOS operating system, using the following default boot device polling sequence.

1. Xylogics Disk (xy), (xd), (xt).
2. SCSI Disk (sd), (st).
3. Ethernet (ie), (le).

If the EEPROM value at address 0x18 *is* equal to 0x12, the system will boot the SunOS operating system from an EEPROM-specified device. The boot device is specified in locations 0x19 through 0x1D, inclusive, of the EEPROM. Refer to the `q` command for information on how to open and modify these EEPROM locations.

When the diagnostic switch is in the DIAG position and command **b** is entered by itself, the system will boot an EEPROM-specified program from an EEPROM-specified device. In this case, the boot path is specified in locations 0x28 through 0x50, inclusive, of the EEPROM; the boot device is specified in locations 0x22 through 0x26, inclusive, of the EEPROM. If the boot attempt fails, the user is returned to the monitor's command level.

In order to boot from a specific device, the **b** command must be followed with a boot device abbreviation, such as those shown below. Enter **b ?** to view the boot device identifier arguments that your PROM monitor will accept.

b *device* (*controller*, *unit*, *partition*) *path argument_list*

You must surround `controller`, `unit`, and `partition` with parentheses, and place a comma after each entry. To invoke the default values, simply enter:

b *device* **(,,)**

device may be one of the following:

fd — Floppy Disk
xd — Xylogics 7053 Disk
xy — Xylogics 450/451 disk
sd — SCSI disk
ie — Intel Ethernet
st — SCSI tape
xt — Xylogics 472 Tape
mt — Tape Master 9-Track Tape

controller stands for the Controller Number, referring to the tape or disk controller board. The default is 0.

unit refers to Unit Number, meaning disk number. The default is 0.

partition is the partition number of the boot device. The default is 0.

path is the path and filename of the program to boot.

argument_list is the list of arguments for the boot program. Up to seven optional arguments (which are passed to the boot program) may follow the path argument.

If you do not want the system to be reset prior to booting, you must insert ! before the device identifier argument.

```
b ie(0,0,1)/stand/video.diag -t
```

The boot command shown above would boot the video diagnostic from the `/stand` directory over the Ethernet. Because the ! argument does *not* precede the device identifier argument, the system is reset before the booting process begins. Note that one optional argument (`-t`) is passed on to program `video.diag`.

Monitor **c** Command

c *virtual_address*

The `continue` command resumes execution of an interrupted program. You can specify the virtual address of the instruction to be executed when the program restarts.

By default, the program resumes execution at the address pointed to by the program counter.

NOTE *This command is helpful if you should use the* `L1-A` *sequence and decide you did not want to abort the operating system.*

Monitor **d** Command

d *register_number action*

The `data register` command provides access to the CPU's data registers. *register_number* can be a value from 0 to 7 inclusive. The default value is 0. If you do not specify a register, this command displays the data registers one-at-a-time, in ascending order.

It is important to note that it is *not* possible to display the state of the processor at all times. In particular, processor state is only observable after:

- An unexpected trap
- A user program has ''dropped'' into the monitor (by calling monitor function `abortent`)
- You have manually ''broken'' into the monitor (by typing `L1-a` on the console's keyboard or **break** on the ''dumb'' terminal's keyboard).

Read the previous section, *Displaying and Modifying Memory* for details on how to use this command. Replace the word *command* in the description with the letter **d**.

Monitor **e** Command

e *virtual_address*

The `display/modify memory` command displays and/or modifies the content of one or more virtual addresses in *word* mode (i.e. each virtual address will be treated as a 16-bit unit). That is, the content of one or more words can be *displayed*, *modified* or, both *displayed* and *modified.*

See the previous section *Displaying and Modifying Memory* for a description of this command's syntax. Use the letter **e** in place of the word *command* in the description.

Monitor **f** Command

f *start_virtual_address end_virtual_address pattern size*

The `block write` command writes the *pattern* you enter into each byte, word or long word in the range of virtual addresses you specify with the *start_virtual_address* and *end_virtual_address* arguments . By default, if the final, memory-cell-size argument is not present, bytes are written. Arguments *start_virtual_address end_virtual_address* and *pattern* are required while *size* is optional. The possible values for *size* are `b` (8-bit byte), `w` (16-bit word) or `l` (32-bit long word).

Monitor F Command—Sun-3/400 Series

For the Sun-3/400 series, the **F** command flushes the ATC Cache on the MC68030 CPU.

Monitor g Command

g *vector argument*
or

g *virtual_address argument*

When you use the `goto` command, you may go to (jump to) a *pre-determined*, *user-specified* or *default* routine. If a pre-determined or default routine is invoked, optional arguments `vector` (a hexadecimal number) and `argument` (a string) are passed to the routine to be executed.

In its other form, the argument `virtual_address` is used to indicate the virtual address of a *user-specified* routine, and the optional `argument` is passed along to that routine. If you do not supply the `vector/virtual_address` argument, the value in the Program Counter is used.

In order to set up a *pre-determined* routine, a user program has to set variable `*romp->v_vector_cmd` equal to the virtual address of the routine. Variable `*romp->v_vector_cmd` must be set prior to executing the `g` command. Pre-determined routines may or may not return to the monitor.

The *default* routine, defined by the monitor, simply prints the user-specified `vector` argument according to the user-specified format (given in `argument`) before returning to the monitor. The only allowable formats are `%x` and `%d`. Format `%x` prints argument `vector` as a hexadecimal number while format `%d` prints argument `vector` as a decimal number.

Monitor h Command

h *or* **?**

The `help` command brings up a Help display that describes the basic monitor commands and their argument(s). An example of the help menu is shown at the beginning of this chapter.

If the help menu on your system has numbered entries, you may enter the number next to the command, and then Return to obtain further information about that command.

Monitor i Command — Sun-3/200 or Sun-3/400 series

i *c icache_data_offset*

The `modify cache data RAM` command displays and/or modifies the contents of one or more of the cache data addresses. For Sun-3/400 series systems, you may add a `c` after entering `i`, to specify Central Cache. Adding a second `i` specifies I/O Cache.

Read the previous section *Displaying and Modifying Memory* for details on how to use this command. Replace the word *command* in the description with the letter **i**.

Monitor `j` Command — Sun-3/200 or Sun-3/400 Series

`j` *cache_tag_offset*

The `modify cache tag RAM` command displays and/or modifies the contents of one or more of the cache tag addresses. For Sun-3/400 Series systems, you may add a `c` to specify Central Cache, or an `i` to specify I/O Cache.

Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *command* in the description with the letter `j`.

Monitor `k` Command

`k` *reset_level*

The `reset` command performs various levels of system resets. It can also be used to display the system's banner. This command accepts one optional argument. Entering `k 0` resets the VME-bus, interrupt register and video monitor (the Low-Level Reset).

Entering `k 1` invokes a Software Reset.

Entering `k 2` invokes a Power-On Reset (Hard Reset).

Finally, entering `k b` displays the banner on the video monitor. The default value of the argument is `0`.

Monitor `l` Command

`l` *virtual_address*

The `modify long words of memory` command displays and/or modifies the contents of one or more virtual addresses in *long* word mode. Each virtual address is treated as a 32-bit unit (long word). Read "Displaying and Modifying Memory" for details on how to use this command. Replace the word *command* in the description with the letter `l`.

Monitor `m` Command

`m` *a b virtual_address*

The `modify segment table` command displays and/or modifies the contents of one or more of the Segment Table entries. For Sun-3/400 series systems, you may specify `a` for translation table TIA, or `b` for translation table TIB. If you do not specify the table, TIA is displayed by default.

Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *command* in the description with the letter `m`.

Monitor **n** Command — Sun-3/200 and 3/400 series only

n *c i cache_command*

The `control cache` command globally controls the cache. Entering **n d** *disables* the cache (Central Cache for Sun-3/400 series systems). Entering **n e** *enables* the cache (Central Cache for Sun-3/400 series systems). Finally, entering **n i** *invalidates* the cache (Central Cache for Sun-3/400 series systems).

For Sun-3/400 series systems, enter

ni *d, e, or i*

to perform the action on the I/O Cache. To select the Central Cache, enter

nc *d, e, or i*

Monitor **o** Command

o *virtual_address*

The `modify bytes of memory` command displays and/or modifies the content of one or more virtual addresses in *byte* mode. Each virtual address is treated as an 8-bit unit (byte). Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *command* in the description with the letter **o**.

Monitor **p** Command

p *virtual_address*

The `modify page table` command displays and/or modifies the contents of one or more of the Page Table entries. For Sun-3/400 series systems, the virtual address determines whether the page displayed is from the page table or from the I/O Mapper RAM. A virtual address between 0xff000000 and 0xffffffff searches for the requested page in the I/O Mapper RAM pages.

Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *command* in the description with the letter **p**.

Monitor **q** Command

q *eeprom_offset* or **q ***

The `modify bytes of EEPROM` command displays and/or modifies the configuration information within the EEPROM in *byte* mode. This command works similarly to the memory commands discussed previously, except that the modified addresses are offset, and the changes you make remain when you power-down the workstation. Read the previous section, *Displaying and Modifying Memory* for details on how to use this command. Replace the word *command* in the description with the letter **q**. Refer to the *Sun-3 EEPROM Layout* chapter of the for information *PROM User's Manual* on what parameters are stored at which EEPROM addresses.

q*—Version 2.4 and later Boot PROM Only

In the Version 2.4 Boot PROM, you may enter an asterisk instead of an EEPROM offset value as an argument, and the **q** command will erase the entire EEPROM.

Monitor `r` Command

`r` *register_symbol*

The `miscellaneous register` command displays and/or modifies the contents of the CPU's miscellaneous registers. You may enter any of the symbols shown on the table below to specify registers. For example, if you enter the following, you will display the state of the MC68030 Cache Control register:

```
>r cc
```

Table 2 *Miscellaneous Registers for the 68020*

Symbol	*Name*
`IS`	Interrupt Stack Pointer
`MS`	Master Stack Pointer
`US`	User Stack Pointer
`SF`	Source Function Code
`DF`	Destination Function Code
`VB`	Vector Base
`CA`	Cache Address Register
`CC`	Cache Control Register
`CX`	Context Register
`SR`	Status Register
`PC`	Program Counter

Table 3 *Miscellaneous Registers for the 68030*

Symbol	*Name*
`SS`	System Stack
`US`	User Stack
`VB`	Vector Base Register
`PC`	Program Counter
`SR`	CPU Status Register
`CC`	Cache Control Register
`CA`	Cache Address Register

It is important to note that it is *not* always possible to display the registers. They may be observed only after

- An unexpected trap
- A user program has "dropped" into the monitor (by calling monitor function `abortent`) or
- You have manually "broken" into the monitor (by typing `L1-a` on the Sun keyboard or [Break] on a "dumb" terminal's keyboard).

Read the previous section *Displaying and Modifying Memory* for details on how to use this command. Replace the word *command* in the description with the letter `r`.

Monitor **R** Command—(Sun-3/400 Series only)

The uppercase **R** command reads the System Enable, Interrupt, Bus Error, I/O Mapper and Memory Error registers in the MC68030. Addresses of those registers are as follows:

Register	*Address*
System Enable	0xa100
Interrupt	0x81
Bus Error	0xe3
I/O Mapper	
Memory Error	0x007feab8

Monitor **s** Command

s *number*

The `modify` command sets or displays the address space to be used by subsequent memory access commands. The processor function codes decode the address spaces. Argument choices represent the function codes:

Table 4 *Function Code Values*

Value	Address Space
0	Reserved (don't use)
1	User Data
2	User Program
3	Control Space (Reserved in Sun-3/400 Series)
4	Reserved (don't use)
5	Supervisor Data
6	Supervisor Program
7	Supervisor CPU Space

If you do not enter a function code number, the current setting (either 1 or 5) is displayed, and entry of the monitor **o** command, for example, would cause the program to look for the specified address in either user or supervisor data space. Conversely, if you reset the function code number, you could query registers in the space represented by the number entered. For example, entering

```
>s 3
```

would allow you to read the bus error or system enable register, which are located in Control Space.

Monitor **T** Command—Sun-3/400 Series

Entering and uppercase **T** for the Sun-3/400 series translates the virtual address entered after the command to its corresponding physical address, similarly to the `^t` command for other Sun-3 systems. If you entered

```
>t 12345678
```

for example, the display might look like this:

```
Virtual Addr 0x12345678 is mapped to Physical Addr 0xfef12345
TIA Entry = 0x7fa400, TIB Entry = 0x7fa60a PTE = 0x59

        Page 0x1a has these attributes:
        No Cache          = 1
        Modify bit        = 1
        Access bit        = 1
        Write Protect bit = 0
        Descriptor Type   = 1
```

Monitor **u** Command

u *port options baud_rate*
or

u *echo*
or

u **u***virtual_address*

The *input/output* command configures the input and output devices and their characteristics or displays the current input and output device set-up.

The **u** command requires arguments to specify from which device(s) you want the system to expect input or which device(s) will display output.

If you do not enter an argument after the u command, the program will display the current settings. If no serial port is specified when changing baud rates, the baud rate of the current input device is changed. The default serial port baud rate is 9600 for both ports during a normal boot. During a diagnostic boot, defaults are 9600 baud for Port A and 1200 for Serial Port B.

Upon normal power-up (diag switch is in NORM position), the default console input device is the Sun keyboard, unless the EEPROM has specified another default input device. If the keyboard is unavailable, the system looks to serial port A for for input.

The default console output device is the Sun monitor (subject to change through EEPROM programming). If the workstation has a color monitor and a color board is unavailable, the program will look for a monochrome monitor as an output device.

You may alter the existing I/O settings while you are in the monitor mode, using the commands listed below; however, the default settings will be reinstated when the system is power cycled.

u Command Arguments:

The *port* argument can be one of the following:

Table 5 *Port Arguments*

Port Argument	Device
a	Serial Port A
b	Serial Port B
k	Keyboard
s	Screen

The `options` arguments are:

Table 6 *Option Arguments*

Option Argument	Meaning
i	input
o	output
u	UART
e	input echoed to output
ne	input *not* echoed to output
r	reset specified serial port

The `baud_rate` argument specifies baud rate of the serial port being discussed. The `virtual_address` argument specifies the virtual address of the UART (Universal Asynchronous Receiver/Transmitter).

Following are examples of port and options arguments:

- Enter **u aio** or **u bio** to select serial port A or B as the input and output device.
- Enter **u ai** or **u bi** to select serial port A or B for input only.
- Enter **u ao** or **u bo** to select serial port A or B for output only.
- Enter **u k** to select the keyboard for input.
- Enter **u ki** to select the keyboard for input.
- Enter **u s** to select the screen for output.
- Enter **u so** to select the screen for output.
- Enter **u ks, sk** to select the keyboard for input and the screen for output.
- Enter **u a***baud rate* or **u b***baud rate* to set the serial port speed.
- Enter **u e** to cause the output to echo the input.
- Enter **u ne** to cause the output **not** to echo the input.
- Enter **u** *address* to set the serial port virtual address.

A *previously mapped* UART can also be used for input and/or output. Command **u u***virtual_address*'', where *virtual_address* is the virtual address of a previously mapped UART, changes the virtual address of the UART. Do not enter a space between the second **u** and the virtual address.

Monitor **v** Command

v *start_virtual_address end_virtual_address size*

The `display memory block` The `display memory block` command displays the contents of each byte, word or long word in the range of virtual addresses specified by *start_virtual_address* and *end_virtual_address*. The *word_size* argument is optional; the default is to display memory in byte format. In this format, sixteen consecutive bytes are displayed on each line. In word format, eight consecutive words appear on each line.

If long word format is enabled, four consecutive long words appear on each line. To the far right of each line, the character corresponding to each byte is also shown. All bytes that contain a non-ASCII code are shown as a period (.). Legal values for *size* are **b** (8-bit byte), **w** (16-bit word), or **l** (32-bit long word).

To terminate the **v** command, press the *space bar*. To "freeze" the display, press any key *except* the space bar. To restart the **v** command, press the space bar again.

Monitor **w** Command

w *virtual_address argument*

The `set execution vector` command allows you to vector to a *pre-determined* or *default* routine. Optional arguments *virtual_address* and *argument* are passed along to the to-be-executed routine.

In order to set up a *pre-determined* routine, a user program sets the variable `*romp->v_vector_cmd` equal to the virtual address of the *pre-determined* routine. Variable `*romp->v_vector_cmd` must be set prior to executing the **w** command. *Pre-determined* routines may or may not return to the monitor.

The *default* routine, defined by the monitor, simply prints the user-specified *virtual_address* argument according to the user-specified format (given in *argument*) before returning to the monitor. The only allowable formats are "%x" and "%d". Format "%x" prints argument *virtual_address* as a hexadecimal number; format "%d" prints it as a decimal number.

Monitor **x** Command

x

The `extended test system` command invokes the Extended Test Sequence or the Extended Test System, depending on what Boot PROM revision is on the CPU board. See *Chapter 9* for details.

Monitor **y** Command — Sun-3/200 or -3/400 Series

y *c number*
or

y *cache_section number virtual_address*

The `flush cache` command performs a number of *flush* operations on the cache. Depending on what is to be flushed, two or three arguments are required. In order to flush a context, enter command **y c** *number*, where *number* is a valid context number.

Entering **y s** *number virtual_address* flushes segment *virtual_address* within context *number*. The argument *number* is a valid context number.

Entering **y p** *number virtual_address* flushes page *virtual_address* within context *number*. The argument *number* is a valid context number.

Monitor **z** Command

z *number breakpoint_virtual_address type len*

Command **z** sets or resets breakpoints for debugging purposes. Optional argument *number* can have values from 0 to 3, corresponding to the processor debug registers DR0 to DR3, respectively. Up to four distinct breakpoints can be specified. If argument *number* is not specified, the monitor will choose a breakpoint number.

The argument *breakpoint_virtual_address* specifies the breakpoint address to set.

The argument *type* can have three values: **x** for Instruction Execution breakpoint, **m** for Data Write only breakpoint, and **r** for Data Reads-and-Writes-only breakpoint. The default value for *type* is **x**.

The argument *len* can also have three values: **b**, **w**, and **l**, corresponding to the breakpoint field length of byte, word, and long-word, respectively. The default value for *len* is b. Since the breakpoints are set in the on-chip registers, an instruction breakpoint can be placed in ROM code or in code shared by several tasks.

If the argument *number* is specified but the argument *breakpoint_virtual_address* is not specified, then the corresponding breakpoint will be reset.

This command without any arguments will display all the existing breakpoints.

1. SPARCsystem 330 Self-Tests and Monitor Commands

For the most part, the diagnostic self-test and monitor command descriptions for the SPARCsystem 330 are the same as those in Chapters 11, 12 and 13 of the *PROM User's Manual.* This addendum provides descriptions of commands or tests that differ from those used for Sun-4/110 or Sun-4/200 series systems.

When a terminal is attached to Serial Port A, and the system in is diagnostic mode, the name of each self-test is displayed as the test is executed, as shown on the following pages.

When in normal mode, self-tests for 8 Mbtyes of memory may last 45 seconds; a 128 MB memory may take up to 8 minutes. In diagnostic mode, when all of memory is tested, an 8 MB system may take four minutes to complete self-tests, while a 128 MB system may take 56 minutes to execute the self-tests.

Self-Test Interaction

There are several commands that allow limited interaction with the SPARCsystem self-tests, and they are described below.

b Press the **b** (a mnemonic for *burn-in*) key, prior to the display of the `...Completed` or `Selftest Finished` message, to execute the power-up test sequence indefinitely. This option is useful during the manufacturing burn-in stage.

For the Sun-3/80, when the last selftest is finished the message tests`Automatically`continuing will be displayed and the self-test will be restarted again, using the System Enable Register read test as the first test. The SCC and terminal I/O tests are bypassed in burn-in mode since operator interaction is required. This looping sequence will continue until an ESCAPE is entered, a reset is done, or the "*b*" key pressed again to turn burn-in mode off. The burn-in mode can be toggled by successive pressing of the "*b*" key. Note that this burn-in mode key can be processed during a test that is running normally (no errors), if the test is presently looping on an error encountered, or at the end of selftest when the "Selftest Finished" is displayed and an operator input is requested.

s Press the **s** key prior to the display of the `...Completed` message to *re-start* the power-up test sequence.

Space Bar
If one of the power-up tests fails, it will continue to re-execute forever unless interrupted. Press the [space bar] to terminate the failed test and execute the next power-up test.

p — Toggle Print Mode
By default, an unsuccessful power-up test prints one error message before entering an infinite scope loop. Once inside of the scope loop, the failing test will *not* print any more messages. If, however, you would like to see test messages while in the scope loop, press the **p** key. Then, to turn the messages back off, press the **p** key a second time. In other words, the `p` command acts like a toggle switch to turn message mode on or off.

`Escape Key`
Pressing [Esc] prior to the `Selftest`Completed message skips the remaining power-up tests and warns:

```
<Warning: Selftests aborted by user>
```

`Control-L`

Holding down the [Control] key while pressing **L** causes the test that is currently running to re-execute forever. Pressing the [Control-L] sequence again turns toggle-loop mode off.

`Control-Q`

Holding down the [Control] key while pressing **Q** terminates the current test and skips to the next test in the power-up self-test sequence. This feature works on a passing or failing test.

NOTE *If EEPROM location 0x17 is set to 0x12, you may press the User Reset Switch at the back of the system to restart the self-tests. The tests will restart regardless of the position of the Diagnostic Switch.*

Figure 7 *SPARCsystem 330 Diagnostic Boot-Up Display*

```
Boot PROM Selftest
EPROM Checksum Test.
Context Register Test.
Segment Map Write-Write-Read-Read Test.
Segment Map Address Test.
Segment Map 3-Pattern Test.
Page Map Write-Write-Read-Read Test.
Page Map Address Test.
Page Map 3-Pattern Test.
Software Traps Test: levels 0x80 -> 0xFF
Interrupt Register Test
Software Interrupts Test.
TOD Interrupt Test.
<Probing for a P4 Video Board>
(For Color systems, refer Video tests change as shown on tables that follow
Video Memory Write-Write-Read Test. 32-bit (black and white)
Video Memory Address Test. 32-bit (black and white)
Video Memory 3-Pattern Test. 32-bit (black and white)
<Probing for Main Memory>
<Found 0x00000008 MB of Main Memory>
Limited Memory Test: End of Megabyte 0x00000008. Tests Complete.
MMU Read Access/Modified Bits Test.
MMU Write Access/Modify Bit Test.
MMU Write to Write-Protected Page Test.
MMU Read From Write-Protected Invalid Page Test
MMU Read Writable Invalid Page Test.
MMU Write To Write-Protected Invalid Page Test.
MMU Write To Writable Invalid Page Test.
Main Memory Non Existent Physical Address Read Timeout Test.
Main Memory Invalid Physical Address Read Timeout Test.
Main Memory Invalid Physical Address Write Timeout Test.
Control Space Timeout Test.
Range Error Test
Size Error Test.
```

The Diagnostic Display is continued on the next page.

Figure 8 *Diagnostic Boot-Up Display - Continued*

```
Parity No Fault Test.
Parity Error Detection and Trap Test.
Cache Tag RAM Write-Write-Read-Read Test.
Cache Tag RAM Address Test.
Cache Tag RAM 3-Pattern Test.
Cache Data RAM Write-Write-Read-Read Test.
Cache Data RAM Address Test.
Cache Data RAM 3-Pattern Test.
<Exiting boot state>
<Successfully exited boot state>
Cache Read Hit (Addr Match,Valid) Test
Cache Read Hit (Addr Match,Valid,Cntx Diff,Spvsr) Test
Cache Read Miss (Addr Match,Valid,Cntx Diff,User) Test
Cache Read Miss (Addr Match,Invalid,Cacheable) Test
Cache Read Miss (No Addr Match,Valid,Cacheable) Test
Cache Read Miss (Addr Match,Invalid,Non-Cacheable) Test
Cache Write Hit (Addr Match,Valid,M=1) Test
Cache Write Hit (Addr Match,Valid,M=0) Test
Cache Write Hit (Addr Match,Valid,Cntx Diff,Spvsr) Test
Cache Write Miss (Addr Match,Valid,Cntx Diff,User) Test
Cache Write Miss (Addr Match,Invalid,Cacheable) Test
Cache Write Miss (No Addr Match,Valid,Cacheable) Test
Cache Write Miss (Addr Match,Invalid,Non-Cacheable) Test
Cache Flush Context (Cntx Match) Test
Cache Flush Context (Cntx No Match) Test
Cache Read Double Hit (Addr Match,Valid) Test
Cache Read Double Hit (Addr Match,Valid,Cntx Diff,Spvsr) Test
Cache Read Double Miss (Addr Match,Valid,Cntx Diff,User) Test
Cache Read Double Miss (Addr Match,Invalid,Cacheable) Test
Cache Read Double Miss (No Addr Match,Valid,Cacheable) Test
Cache Read Double Miss (Addr Match,Invalid,Non-Cacheable) Test
Cache Write Double Hit (Addr Match,Valid,M=1) Test
Cache Write Double Hit (Addr Match,Valid,M=0) Test
Cache Write Double Hit (Addr Match,Valid,Cntx Diff,Spvsr) Test
Cache Write Double Miss (Addr Match,Valid,Cntx Diff,User) Test
Cache Write Double Miss (Addr Match,Invalid,Cacheable) Test
Cache Write Double Miss (No Addr Match,Valid,Cacheable) Test
Cache Write Double Miss (Addr Match,Invalid,Non-Cacheable) Test
Cache Successive Write Double Hit (Addr Match,Valid,M=1) Test
Cache Successive Write Double Hit (Addr Match,Valid,M=0) Test
```

This display is continued on the following page.

Figure 9 *Diagnostic Boot-Up Display - Continued*

```
  Atomic Load/Store Hit (Addr Match,Valid) Test
  Atomic Load/Store Miss (Addr Match,Invalid) Test
  Atomic Load/Store Hit (Addr Match,Valid,Write Protected Page) Test
  Atomic Load/Store Miss (Addr Match,Invalid,Write Protected Page) Test
  <Re-entering boot state>
  <Successfully re-entered boot state>
  VME Loopback and DVMA Test
  VME Loopback and DVMA Read Timeout Test
  VME Loopback and DVMA Write Timeout Test

  END OF SELFTEST # 0x00000000 (SELFTEST PASSED) .
  #PASSED = 0x00000000,  #FAILED = 0x00000001

  <Sizing Main Memory>
  <Found 0x00000008 MB of Main Memory>
  <Initializing Memory...>
  <Turning Parity Checking ON>
  Main Memory Test: Megabyte 0x00000008. Tests Complete.
  <Found 0x00000008 MB of Main Memory>
  <Initializing Memory...>

  Selftest Completed.

Type a key within 10 seconds to enter Extended Test System (e for echo mode).
```

If the system contains a CG6 board, the self-tests function the same as the Video tests shown previously, and the display looks like this:

```
<Probing for a P4 Video Board>
Video Memory Write-Write-Read-Read Test. 32-bit - Color Plane.
Video Memory Address Test. 32-bit - Color Plane.
Video Memory 3-Pattern Test. 32-bit - Color Plane.
```

If the system contains a CG8 board, the self-tests function the same as the Video tests shown previously, and the display looks like this:

```
<Probing for a P4 Video Board>
Video Memory Write-Write-Read-Read Test. 32-bit - Overlay Plane.
Video Memory Address Test. 32-bit - Overlay Plane.
Video Memory 3-Pattern Test. 32-bit - Overlay Plane.
Video Memory Write-Write-Read-Read Test. 32-bit - Enable Plane.
Video Memory Address Test. 32-bit - Enable Plane.
Video Memory 3-Pattern Test. 32-bit - Enable Plane.
Video Memory Write-Write-Read-Read Test. 24-bit - Color Plane.
Video Memory Address Test. 24-bit - Color Plane.
Video Memory 3-Pattern Test. 24-bit - Color Plane.
P4 Color Map Test.
```

If the Diagnostic Switch is in the *diagnostic* position, the power-up tests executed successfully and the user did *not* enter a character on either of the *terminal* keyboards within the initial ten second period, the following display will appear on the *workstation's* screen. Again, note that you have a *second* ten-second opportunity to invoke the additional non-power-up tests by pressing *any* key on the workstation's keyboard, except the [Esc] key. Pressing the [Esc] key in this situation will terminate the delay and invoke the PROM monitor program. Providing two opportunities to enter the Extended Test System will give the user the choice of interacting with either an RS232 terminal or the workstation's keyboard and video monitor.

If you do not respond to either of the *ten-second* delays, an attempt will be made to boot an EEPROM-specified program from the Diagnostic boot path area of the EEPROM. If this attempt should fail, the monitor program is invoked.

Successful Self-Test Display

The test names actually appear on the screen sequentially as each test is performed. The previous example shows the display on the the terminal (connected to Serial Port A at 9600 baud or Port B at 1200 baud) screen after all the self-tests are successfully completed.

NOTE *If no terminal is connected to the serial port, you will not be able to interact with the self-tests, and you won't see the Diagnostic Boot-Up Display. If self-test is successful, the console display prompts will offer choices described on the following page.*

```
Selftest Completed Successfully

         Sun Workstation, Model Sun-4/300 Series,
         Sun-4 Keyboard
        ROM Rev __, __ MB memory installed, Serial #______
         Ethernet address __:__:__:__:__:__

Type a character within 10 seconds to enter Extended Test System.
```

If you do not press a key on the console keyboard, the Boot PROM program checks parameters stored in the EEPROM that tell it to do one of the following:

- Boot a specific (diagnostic) program from a specific device
- Drop into the PROM monitor mode.

If you press a key, the Extended Test System menu is offered. Refer to the SPARCsystem 330 extended test system addendum at the end of this chapter.

To Read the CPU Board LED Table

The following table provides a brief interpretation of the patterns displayed by the LED indicators on the edge of the CPU board. For vertical installations, the LED depicted on the left in this table is located at the top of the display; the LED on the right is at the bottom of the display.

Table 7 *SPARCsystem CPU Board LED Interpretation*

LED Display ● = ON, ○ = OFF								*Self-Test being Performed.*
7	6	5	4	3	2	1	0	
● ⇒	○ ⇒	○ ⇒	○	○ ⇐	○ ⇐	○ ⇐	●	LED Loop Test
○	○	○	○	○	○	○	○	SCC WR 12 Write-Read Test
○	○	○	○	○	○	○	●	Initialize SCC UART
○	○	○	○	○	○	●	○	"BOOT PROM Selftest" Message
○	○	○	○	○	○	●	●	EPROM Checksum Test
○	○	○	○	○	●	○	○	Context Register Read-Write Test
○	○	○	○	○	●	○	●	Segment Map Tests
○	○	○	○	○	●	●	○	Page Map Tests
○	○	○	○	○	●	●	●	Software Traps Test
○	○	○	○	●	○	○	○	Interrupt Tests (Software and Register)
○	○	○	○	●	○	○	●	TOD Interrupt Test
○	○	○	○	●	○	●	○	Video Memory Tests
○	○	○	○	●	○	●	●	Limited Main Memory Tests
○	○	○	○	●	●	○	○	MMU Read Access/Modified Bits Test
○	○	○	○	●	●	○	●	MMU Write Access/Modified Bits Test
○	○	○	○	●	●	●	○	MMU Write to Write-Protected Page Test
○	○	○	○	●	●	●	●	MMU Read Not-Writeable Invalid Page Test.sp 2p
○	○	○	●	○	○	○	○	MMU Read Writeable Invalid Page Test
○	○	○	●	○	○	○	●	MMU Write Not-Writeable Invalid Page Test
○	○	○	●	○	○	●	○	MMU Write Writeable Invalid Page Test
○	○	○	●	○	○	●	●	Main Memory Timeout Test
○	○	○	●	○	●	○	○	Control Space Timeout Test
○	○	○	●	○	●	○	●	Range Error Test
○	○	○	●	○	●	●	○	Size Error Test
○	○	○	●	○	●	●	●	Parity Memory Test
○	○	○	●	●	○	○	○	CPU Cache Tag RAM Tests
○	○	○	●	●	○	○	●	CPU Cache Data RAM Tests
○	○	○	●	●	○	●	○	CPU Cache Functional Tests
○	○	○	●	●	○	●	●	VME Loopback Tests
○	○	○	●	●	●	○	●	Main Memory Tests (Parity on)

The SPARCsystem 330 PROM Monitor

The boot PROM stores a program known as the ''monitor'' that contains self-tests and controls system operation during boot-up until the SunOS kernel takes over. The monitor program is invoked when you use the halt SunOS, and is identified with the ">" prompt.

The PROM monitor program provides commands that perform a variety of tasks, such as booting from alternate devices, changing the console output to a serial port, and so on.

Access the monitor as described in the Manual,*PROM*User's then enter **h** to view a "Help" table of monitor commands that looks something like this:

```
Monitor        REV:1              date              Help Menu

1  b           Boot a program.
2  k           Reset all or part of the machine or display the banner.
3  u           Initialize the input and output devices.
4  c/g/w       Resume or modify program flow.
5  d/r         Display and/or modify the registers.
6  o/e/l       Display and/or modify individual memory locations.
7  ^c/f/v       Copy, display or modify a block of memory.
8  m/p         Display and/or modify segment or page table entries.
9  q           Display and/or modify EEPROM locations.
10 s           Display or modify the Address Space Identifier.
11 i/j         Display and or modify cache data or tag entries.
12 n/y         Disable, enable, invalidate, or flush the cache.
13 ^a/^t       Display virtual and physical address information.
   ^i          Display firmware compilation information.
   ^p          Enable or disable parity circuitry.
    !          Re-execute the previously executed command.
   h/?         Enter the help system for the basic monitor commands.
   x           Enter the Extended Test System.

''option_number''= Additional Help <esc> = Go-To-Previous Menu q= quit

Command == >
```

If you need more help concerning any of the Help Menu options, enter the number that precedes the command, and then [RETURN].

The "Monitor (8S)" section of the *Commands Reference Manual* also provides information on PROM monitor commands.

The monitor commands described in Chapter 12 of the *PROM User's Manual* are valid for the SPARCsystem 330, with the minor changes described below.

When using the `i, j` or `n` or `y` commands, you must specify the type of cache on which you want the operation performed. For example, you may enter the monitor `j` command, followed with **cpu** and the central processor cache will be opened for display or modification. If you replaced `cpu` with **io**, the I/O cache would be displayed or modified:

`j` *cpu cache_offset*

Refer to the "Displaying and Modifying Memory" section of the *PROM User's Manual* for more information on the use of these commands.

Monitor `r` Command

Add the following SPARCsystem 330 registers to the Processor Register table in the *PROM User's Manual* Sun-4 PROM Monitor Command descriptions:

Register Number	*Register Name*
0x80 - 0x87	g0,g1,g2,g3,g4,g5,g6,g7
0x88 - 0x8d	PSR,PC,nPC,WIM,TBR,Y
0x8e - 0xae	FSR,f0,...,f31

Monitor `s` Command

For SPARCsystem 330 CPU boards, you may enter the following Address Space Identifiers (ASI) after the s command. If you do not enter an (ASI) value, the current value will be displayed.

Name	*ASI (hex)*
Control Space	2
Segment Map	3
Page Map	4
Block Copy	5
User Instruction	8
Supervisor Instruction	9
User Data	a
Flush Page	d
Flush Context	e
Flush User	f

Monitor `t` Command

For the SPARCsystem 330, the t command displays or modifies the content of one or more region table entries. Refer to the *PROM User's Manual* for more information on using and modifying memory locations.

Monitor `v` Command

The "view" command, described in the *PROM User's Manual* For the SPARCsystem 330, pressing the space bar terminates the command. To freeze the display before it scrolls off the screen, press any key *other than the space bar*. To restart the **v** command, press any key except the space bar.

If you enter two asterisks (* *) at the end of the **v** command string (which includes the low and high virtual address and byte, word or long word specification), the view command will continue to read each location until you press the space bar. If you add only one asterisk, the content of each location will be displayed once.

Monitor ^a Command

For SPARCsystem 330 CPU boards, entering the ^ character followed by an **a** displays a list of devices and their corresponding physical and virtual addresses.

Monitor ^c Command

For SPARCsystem 330 CPU boards, entering the ^ character followed by an **c**, a source and destination virtual address, and the number of bytes, copies the content of those locations to another location.

Monitor ! Command

Entering an exclamation point after the PROM monitor prompt on a SPARCsystem 330 repeats the last executed basic monitor command.

Identifying A Faulty Memory Module

In the event a memory error is detected during the power-on self tests, the faulty SIMM will be identified by its "U Number" designation.

Additional memory tests are available in the PROM monitor Extended Test System. After entering the "Help" table shown at the beginning of the "The PROM Monitor" subsection, select **x** and enter the extended test system. Most of the extended tests are the same as those documented in Chapter 13 of the *PROM User's Manual.* Chapter 13 describes the extended test system user interface. You may enter command lines, or simply make menu selections. You need only enter the letters shown in upper case on the extended test menus.

NOTE *Entering an exclamation point displays the last five command lines you have entered. To repeat one of those command lines, simply enter the appropriate command line number.*

The tests that differ from those described for the Sun-4/200 and Sun-4/110 series of workstations are described in the next section.

SPARCsystem 330 Extended Tests

```
Monitor  REV:some number  Date  Main Menu


All             All Test Sequence.
Default         Default Test Sequence.
Boot            Boot Paths Menu.
Ethernet        Ethernet Menu.
Keyboard        Keyboard and Mouse Menu.
Memory          Memory Menu.
Serial          Serial Ports Menu.
Video           Video Menu.
Color           Color Map Menu.
Options         Set global flags/options.

? = Help <esc> = Return-To-Previous-Menu l = [n] = Repeat-Command-Line-n-Times  q= Quit
Command == >
```

The first Main Menu choice, `All`, brings up the tests shown below. The `Default` selection runs some of, but not all of these tests. Note that the Ethernet loopback connector, as well as the within-channel external loop-back cables for the keyboard, mouse, Serial Port A, and Serial Port B must be installed prior to running the `All` sequence. Contact Sun customer support to obtain a loopback connector kit. The pin assignments for the various connectors appear in *The PROM User's Manual*, with the exception of the Ethernet loopback connector. The pin assignments for the Ethernet connector are:

Figure 10 *Ethernet Loopback Connector*

From Pin	***To Pin***
3	5
10	12
13	14

These tests are performed when the `All` test sequence is chosen from the Extended Test Main Menu for a black and white system:

```
         All Test Sequence

a. Ethernet Local Loop Back Test
b. Ethernet Encoder Loop Back Test
c. Ethernet External Loop Back Test
d. Keyboard Register 12 Test
e. Mouse Register 12 Test
f. Keyboard Transmit Test
g. Mouse Transmit Test
h. Keyboard Internal Loop Back Test
i. Mouse Internal Loop Back Test
j. Keyboard External Loop Back Test
k. Mouse External Loop Back Test
l. Main Memory Address Test/Byte Mode
m. Main Memory Constant Pattern Test/Word Mode
n. Main Memory Fill Utility/Long Mode
o. Main Memory Check Utility/Long Mode
p. Serial Port A Register 12 Test
q. Serial Port B Register 12 Test
r. Serial Port A Transmit Test
s. Serial Port B Transmit Test
t. Serial Port A Internal Loop Back Test
u. Serial Port B Internal Loop Back Test
v. Serial Port A External Loop Back Test
w. Serial Port B External Loop Back Test
x. Video Address Test/Byte Mode
y. Video Constant Pattern Test/Word Mode
z. Video Fill Utility/Long Mode
aa. Video Check Utility/Long Mode
Cmd=>
```

If the SPARCsystem 330 has a color monitor, these tests are also performed:

```
Overlay Plane Address Test/Byte Mode
Overlay Plane Constant Pattern Test/Word Mode
Overlay Plane Fill Utility/Long Mode
Overlay Plane Check Utility/Long Mode
Enable Plane Address Test/Byte Mode
Enable Plane Constant Pattern Test/Word Mode
Enable Plane Fill Utility/Long Mode
Enable Plane Check Utility/Long Mode
Color Plane Address Test/Byte Mode
Color Plane Constant Pattern Test/Word Mode
Color Plane Fill Utility/Long Mode
Color Plane Check Utility/Long Mode
Color Map Address Test
Color Map Constant Pattern Test
Color Map Fill Utility
Color Map Check Utility
```

Consult the Sun-4 Extended Test System chapter of the *PROM User's Manual* for descriptions of the tests listed above. The Extended Test System user interface is the same for all Sun-4 systems. The color tests function exactly as the Video tests described in Chapter 13 of the *PROM User's Manual*, except that they test the overlay, enable and color planes rather than the video circuitry. Some SPARCsystem extended tests documented in the *PROM User's Manual* may be labeled "for Sun-4/2xx only" or "for Sun-4/110 only". If the test is identical to one of the SPARCsystem 330 tests, please ignore the label.

SPARCsystem 330 Initialization

After error-free completion of the power-up self-tests, but before execution of the default boot sequence, the PROM initializes the system. The initialization sequence is:

1. The system enable register is set to normal state.
2. The context register is set to zero.
3. The processor status register is initialized: the CPU is in supervisor mode, window zero is active, level 14 and 15 interrupts are allowed, and traps are enabled. If the floating point processor probe was successful, the Floating Point Unit is enabled.
4. The virtual address of the trap table is written into the trap base register.
5. The window invalid mask register is initialized to 0x2.
6. All page table entries are initialized.
7. Memory is sized.
8. All available main memory is initialized to zero (an unimplemented instruction.)
9. The initial 32 Megabytes of working memory is mapped in ascending order, starting at location zero.

10. One page of RAM is reserved for a stack area and the stack pointer is initialized.
11. A page of RAM is reserved for the PROM monitor global variables.
12. An additional page of RAM is reserved for the monitor's trap vecor table and font table.
13. Assuming that they exist, these devices are mapped and initialized:

    Parity memory registers

    The EEPROM

    Other PROMs

    Ethernet

    The interrupt register

    The keyboard and mouse

    Serial ports

    Time of day clock

    P4 black and white board

    P4 color board

14. The copy of the monitor's trap vector table, which contains the addresses of the trap and interrupt handling routines, is set up.
15. Central processor cache is initialized.
16. Entry points to all support routines provided by the PROM monitor are set up.

Index