SunDiagnostic Executive™ User's Guide

Contents

Tables

Figures

1

Introduction

1

Introduction

This manual describes the programs on the Sun Diagnostic Executive tape. This chapter provides information about the diagnostic environment in general. The remaining chapters describe the Executive and the Diagnostic Programs themselves.

1.1. Glossary

The following list defines some words used in this manual:

Diagnostic
A program designed to test parts of the Sun workstation and return messages describing what it found. Each diagnostic covers a particular PC board or subsystem: for example, `cpu.exec` tests the CPU board on both Sun-2 and Sun-3 systems.

Executive Tape
A 1/4-inch or 1/2-inch magnetic computer tape that contains the Executive and the Diagnostic Programs. Note that you may install these programs on the disk, or boot them directly from the tape.

Exec
Refers to the SunDiagnostic Executive, the operating system that creates and controls the environment under which the diagnostics are executed. It runs by itself, without booting the SunOS Operating System.

1.2. Conventions

Fonts

In this manual, different fonts are used to make things clearer. The most common fonts are Roman, `typewriter`, **`typewriter bold`**, *italic*, and bold. They are used as follows:

Roman
Roman font is the standard for normal text, just as it appears here.

Roman Bold
Bold Roman font indicates that something deserves more attention than the surrounding text.

Typewriter
`Typewriter font` has two meanings, depending on where it appears. It may represent something that appears in the manual exactly as the computer displays it on the screen, or it may represent a program path/name.

Typewriter bold

Typewriter bold font represents something that you must type verbatim into the computer. This sometimes appears together with typewriter font: the computer output appears in typewriter, and what you must type appears in typewriter bold.

Italic

In this manual *Italic font* usually represents a variable for which you or the computer must provide the exact details. For example:

```
error: obs nnnn, exp nnnn
```

Italic font is also used for emphasis, special notes and to reference documents.

Hexadecimal Values

Hexadecimal values are represented throughout this manual with "0x" preceding the value and sometimes replacing the value's leading zeroes.

1.3. References

See the following documents for further information:

- The Field Service Manual for your system
- The Hardware Installation Manual for your system
- The System Administration Manual for your version of the operating system, which describes various system operations, including the use of the standalone disk facility, `diag`, and the proper way to shut down the operating system.

1.4. The Exec Tape

This manual contains a chapter describing each diagnostic on the tape version released as of the date of this writing. This manual references the 1.1 version of the SunDiagnostic Executive tape, for Sun-2 and Sun-3 workstations. The 1/4-inch tape is Sun PN 700-1717. The 1/2-inch tape is Sun PN 700-1718. The diagnostic programs may either be loaded directly off this tape, or copied onto a disk and loaded from there.

The names of the diagnostics contain three fields; the first identifies the function that the diagnostic tests, the second identifies whether the diagnostic runs on a Sun-2 or Sun-3 system, and the third contains the word `exec`. If the number is missing, the diagnostic tests both Sun-2 and Sun-3 systems.

For example, the name `color3.exec` describes a Color Board diagnostic program for a Sun-3; the name `mem.exec` describes the Memory diagnostic for both Sun-2 and Sun-3.

The names of the chapters in this manual reflect the hardware that the diagnostic tests and whether it works on a Sun-2 or a Sun-3.

1.5. Required Equipment

All Sun diagnostics require a complete Sun system to operate. This includes:

- Card cage
- Power supply
- Monitor, video board and keyboard
- Sun-2 or Sun-3 CPU board
- The SunDiagnostic Executive and a means of booting it — A bootable copy of the Exec either on a tape, a local disk, or a remote disk (over the Ethernet).
- The unit to be tested — The unit that the diagnostic tests, and all supporting subsystems.
- Serial Port Loopback connectors, required for CPU board diagnostics.

Additional equipment requirements are listed where necessary.

Serial Port Loopback Connectors

The Serial Port Loopback connectors are designed to connect serial port A to serial port B of a Sun system for testing purposes. The Transmit/Receive, RTS/CTS, and DTR/DCD signal lines are cross connected between the two ports.

The RS-232 Loopback cable is used to test the serial ports of assembled systems.

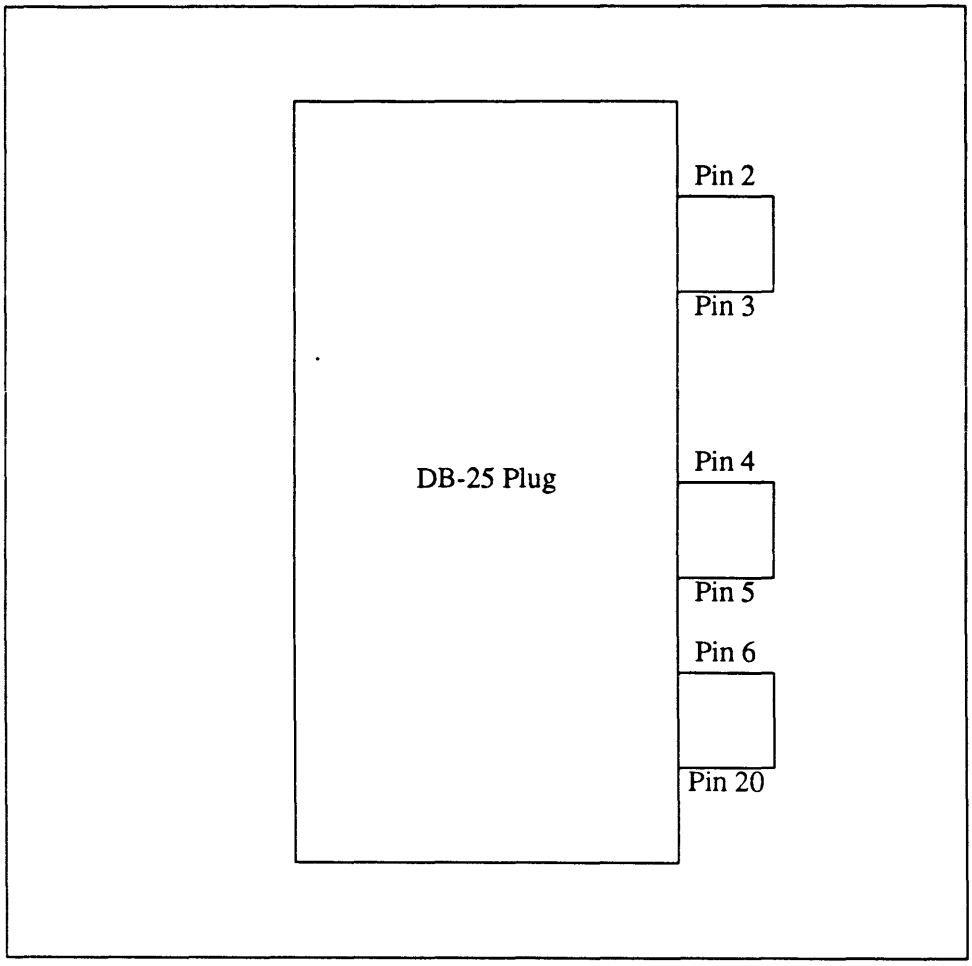
Serial Port Loopback Connector

The RS-232 Loopback Connector is a specially wired male DB-25 connector. It is plugged in to a serial port in the back of a system under test. It is wired as follows:

Connect pin2 to pin3
Connect pin4 to pin5
Connect pin6 to pin20

See the following figure:

Figure 1-1 *RS-232 Loopback Connector*



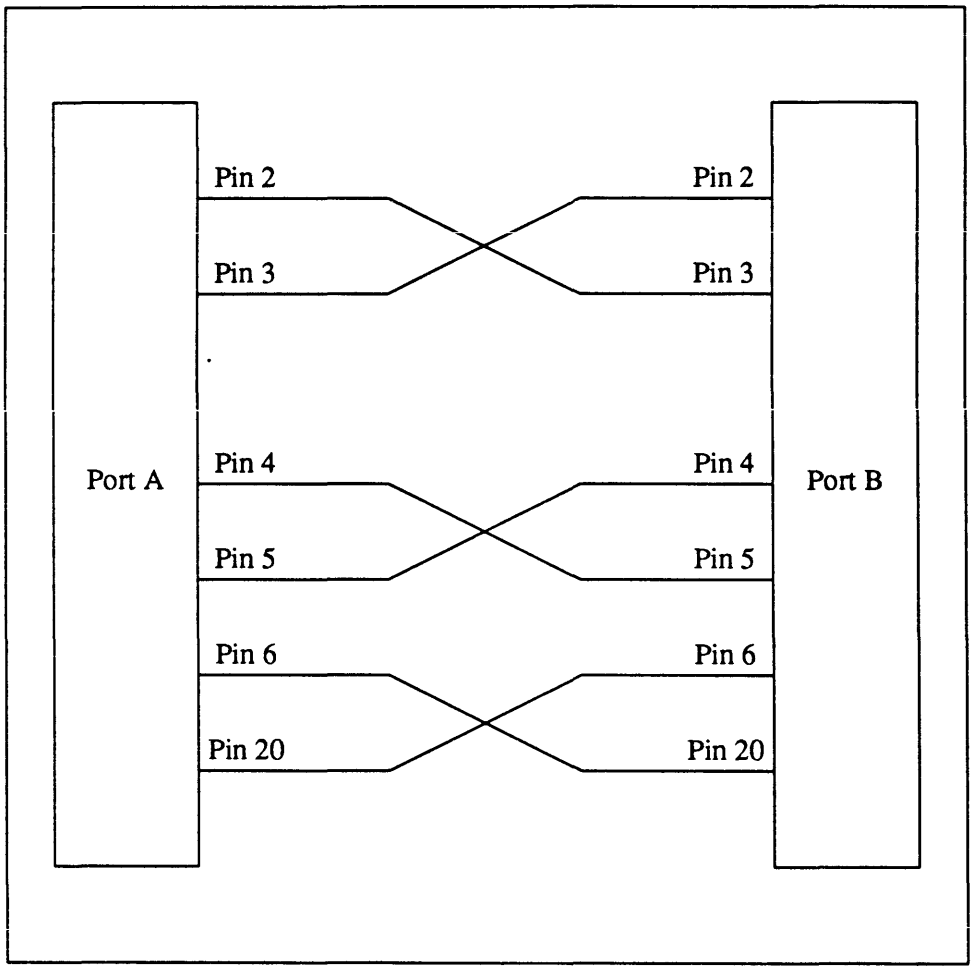
RS-232 Loopback Cable

The RS-232 Loopback Cable is a specially wired cable with two male DB-25 connectors at each end. It is plugged into a pair of serial ports in the back of the system under test. The cable is wired as follows:

Connect pin2 to pin3
Connect pin3 to pin2
Connect pin4 to pin5
Connect pin5 to pin4
Connect pin6 to pin20
Connect pin20 to pin6

See the following figure:

Figure 1-2 *RS-232 Loopback Cable*



NOTE *Loopback connectors must be wired properly and connected firmly for the Serial Port Tests to work correctly. Miswired, poorly soldered, or missing loopback connectors can lead to erroneous diagnostic error messages when diagnostics are run.*

Configuring a Terminal

Some diagnostics are better run from a terminal. To set up the terminal:

Use an ASCII or ANSI terminal, set up as follows:

Full Duplex
9600 baud
XON and XOFF
8 bits/1 stop bit
No parity

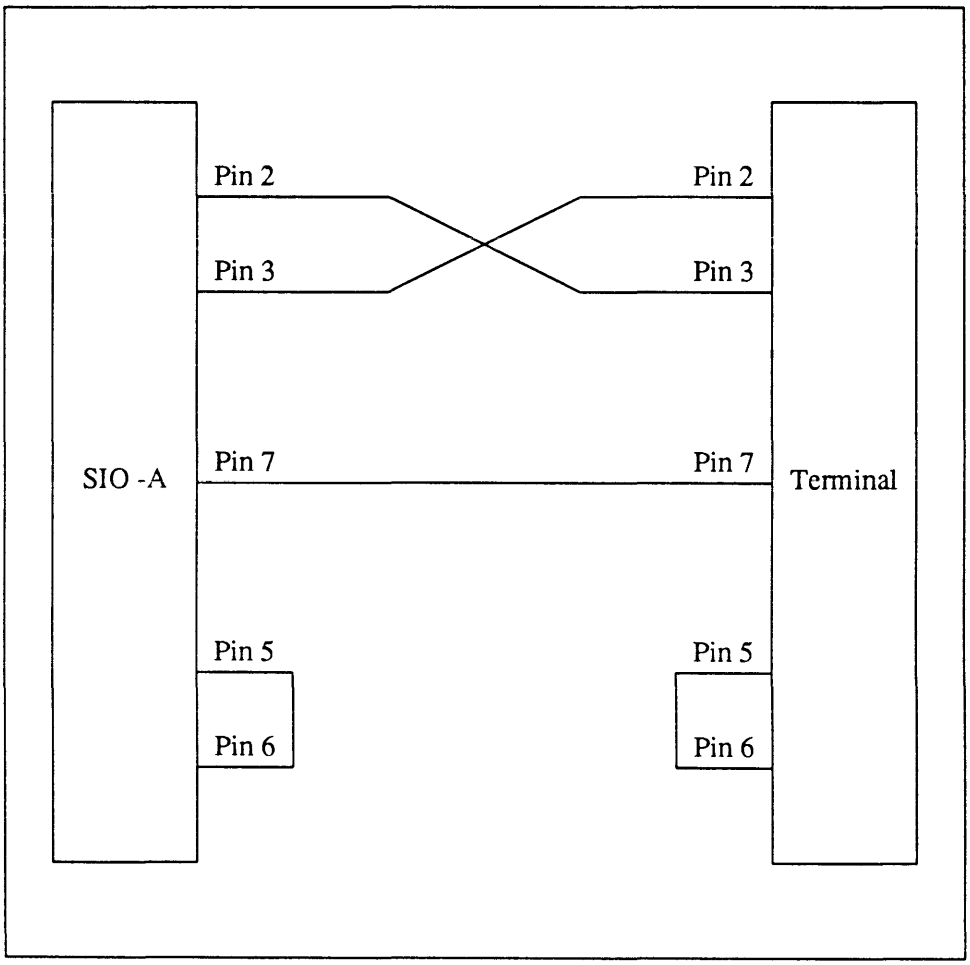
Connect the terminal to the connector labeled "SIO-A" on the system control panel.

Use the following connections:

Cross-connect pins 2 and 3
Loop back pins 5 and 6 at both ends
Connect pin 7 straight through

See the following figure:

Figure 1-3 *RS-232 Connections*



Later, when you have activated the monitor, use the Exec commands to redirect the input and output.

2

Using the SunDiagnostic Executive

2

Using the SunDiagnostic Executive

This document introduces the SunDiagnostic Executive, a hardware diagnostic operating system that provides the user interface for Sun diagnostics.

2.1. History

Originally, Sun diagnostic programs ran "standalone," without the SunOS operating system. They lived in the `/stand` directory, and each program had to be booted from the PROM monitor. This directory contained an inventory of diagnostic programs, typically one per system PC board or major function.

This arrangement worked for a while, but it had two significant drawbacks. First, only one diagnostic program could run at a time, making it hard to load down the system for a thorough test. It took a long time to test more than just a few components. Second, each diagnostic program had its own particular user interface, which made the tests hard to use.

The SunDiagnostic Executive provides a single, unified diagnostic environment with only one user interface. The diagnostic tests still live in individual programs within `/stand`, but they all work through the SunDiagnostic Executive (hereafter called the Exec) which provides a consistent interface and multitasking capabilities.

The Exec has five menus of its own, which contain all of its commands. It can also call up the menus for any diagnostic. It has one Main Menu that appears when the Exec starts. This menu controls the `Environments` Menu, an `Options` Menu, a `Diagnostics` Menu, a `Status` Menu, and a `Log` Menu.

NOTE *In this document, the menus shown are examples only; the menu you see on the screen may differ slightly from these examples.*

2.2. Hardware Requirements

NOTE *Sun does not support a system configuration that includes both a SCSI3 board and a Sysgen disk controller; therefore the SunDiagnostic Executive will not function on a such a system.*

In order to use the SunDiagnostic Executive, the system it runs on must have the following functional hardware:

- Memory — 1Mbyte minimum
- CPU — 68010 (Sun-2) or 68020 (Sun-3)
- The MMU
- Real-time clock
- The system must have at least one of the following:

  System Console and Keyboard
  A terminal plugged into Serial Port A
  A modem plugged into Serial Port B

- A boot path to a storage device:

  A hard disk controlled by SCSI, `sd()`, or Xylogics, `xy()` *xd()*
  A 1/4" tape controlled by SCSI, `st()`
  A 1/2" tape, `mt()` or `xt()`
  An Ethernet controller on a server, `ie()` or `le()`

2.3. Software Requirements

The following software must be available to boot from the system under test:

- The SunDiagnostic Executive
- A set of Diagnostic Programs that run under the Exec.

NOTE *The Diagnostic programs and the Exec must all be at the same revision level (i.e. all from the same tape). The old standalone diagnostics* **won't work** *with the Exec.*

The Exec software may be booted from tape or disk, or booted remotely across the Ethernet.

The Exec Tape

The Exec release tape contains boot programs, the Exec program itself (which works on both Sun-2's and Sun-3's) and the diagnostics that currently can run under the Exec. Two "standalone" diagnostics, `eccmem3.diag` and `cache3.diag`, are included on the tape. They are designed to test Sun-3/200 series workstation memory, and do not function as part of the SunDiagnostic Executive. These tests may be extracted from tape and executed separately. Documentation for these tests is in *Appendix B*.

The table below lists the contents of the tape in order:

Table 2-1 *SunDiagnostic Executive Tape Contents*

Name	*File Number (hex)*	*Decription-Comments*
Sun-2 Boot Block	0	Load before booting Exec on Sun-2
Sun-3 Boot Block	1	Load before booting Exec on Sun-3
Table of Contents	2	Contains list of contents of tape
extract_exec	3	Script to copy diagnostics to disk
Copyright	4	Textfile containing copyright notice
exec	5	The SunDiagnostic Executive, works with Sun-2s and Sun-3s
diags	6	Diagnostic Menu and File names (used by Exec)
color2.exec	7	Sun-2 Color Diagnostic
color3.exec	8	Sun-3 Color Diagnostic
cpu.exec	9	Sun CPU Diagnostic
eeptool.exec	A	EEPROM programming tool
ether.exec	B	Sun Ethernet Diagnostic
exectest.exec	C	Exec Verification Suite
fpa.exec	D	Sun-3 FPA Diagnostic
gp1.exec	E	Graphics Processor1 Diagnostic
kb.exec	F	Sun Keyboard Diagnostic
mcp.exec	10	Sun ALM2/MCP Board Diagnostic
mem.exec	11	Sun Memory Diagnostic
mouse.exec	12	Sun Mouse Diagnostic
mti.exec	13	Sun MTI/ALM Board Diagnostic
scsisub.exec	14	Sun SCSI Subsystem Diagnostic
sky2.exec	15	Sky-2 Diagnostic
smd.exec	16	Sun SMD Diagnostic
tape.exec	17	1/2-inch Tape Diagnostic
video.exec	18	Sun Video Diagnostic
vidmon.exec	19	Sun Video Monitor Diagnostic
vme3.exec	1A	Sun-3 VME Diagnostic
netcon	1B	Network Console Program
logfile	1C	Error Log File
eccmem3.diag	1D	Standalone ECC memory diagnostic
cache3.diag	1E	Standalone cache memory diagnostic
tarfile	1F	`tar` archive of remaining tape contents and data source for `extract_exec`
Copyright	20	Textfile containing copyright notice

The files on the tape are stored in `tar` format, on either a 1/4-inch tape cartridge, PN 700-1717, or a 1/2-inch, 1200 foot tape reel, PN 700-1718. The tapes are titled *1.1 SunDiagnostic Executive Sunbin* and are intended for use with Sun-2 or Sun-3 workstations.

The Exec may be booted directly off this tape, or the contents can be stored in a UNIX† directory for later use.

† UNIX is a registered trademark of AT&T.

2.4. Loading and Booting the Exec

Unless you boot from tape, the Exec loads its programs from UNIX files located on a hard disk. Although the files can be in any directory, we strongly urge you to keep them in the `/stand` directory. This document will assume the Exec and its diagnostic programs are all in `/stand`.

Before Booting

The Exec cannot run unless the system has a minimum amount of functional hardware. The Exec depends on the system tests built into the boot PROMs to ensure the machine is minimumly functional. If you power up the workstation in DIAG mode and the PROM selftest prints out error messages on the attached terminal, the machine is not working well enough to test.

Halting the System

The Exec must be booted from the PROM monitor. To reach monitor mode, you must halt the operating system. This can be done a number of ways. The best way is to use the UNIX `halt` command. To run it, do the following:

```
be sure to shut down all your applications first!

example% su
Password: enter password
example#sync
example#sync
example# /etc/halt

Syncing disks... done
Unix halted

>
```

Another, less preferable way of shutting down the system and bringing up the PROM monitor is to abort the system. Don't do this unless your you have NO OTHER ALTERNATIVE; aborting the operating system may damage your file systems. To abort, hold down the key on the upper left-hand corner of the keyboard (usually L1) and press a. Do this IMMEDIATELY following the

```
Testing _Megabytes of memory....Completed
```

message.

You should see the PROM monitor prompt, >.

Once you are in the monitor, you should reset the system to clear out all of the hardware settings. Do the following:

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

press L1 and A again when the message above finishes

dumping to dev XXX, offset XXXXX      you may not see this one

Abort at XXXXXX
>
```

NOTE *If you are booting the Exec from a SCSI disk or tape, you must* **cycle the power** *before booting. Follow the directions to halt your system, turn the power OFF, then ON, then IMMEDIATELY abort the boot using* `L1-a`, *as described above. Now enter* **k2** *to ensure that the system hardware is reset, in case the operating system began to boot before you aborted.*

Now your system is ready to boot the Exec.

Booting from Tape

If you don't have the Exec installed on disk, you can boot it directly off the Exec Tape. Halt your system as previously described, perform a **k2** reset, then type the following to the PROM monitor:

```
> b st(,,N)        N is 0 for Sun-2 or 1 for Sun-3
boot: st(,,5)
```

Be sure to follow the format shown above. The Exec booting syntax differs from that of the SunOS syntax, so be sure and enter the command exactly as shown. Do not leave parentheses off. The `k2` reset ensures that the operating system does not remain in memory anywhere on the CPU or memory boards.

At this point the Exec should boot up and display the Main Menu.

Installing the Exec

If you don't want to load the Exec from tape each time you run it, you must install it in a UNIX directory.

`/usr/stand`

The Exec and its associated files require approximately 2 Mbytes of disk space. On many systems, the default root partition is not large enough for this. If this is the case, the Exec may be installed in /usr/stand instead of /stand. To do this, follow the installation procedure below, but use the directory `/usr/stand` instead of `/stand`.

If you want to boot the Exec directly from /usr/stand, bootblocks must be installed on the /usr partition. Otherwise, the -a option must be used to boot the Exec. If bootblocks are installed, perform a `k2` reset from the PROM monitor mode, then boot the Exec with:

```
>b device(,,6)stand/exec
```

If bootblocks are *not* installed, boot the Exec with:

```
>b -a
Boot:device(0,0,0)vmunix
Load:device(0,0,0)vmunix
Boot:device(,,6)stand/exec
```

Where *device* is the type of disk the Exec is installed on.

Installing a boot block

If you have the Exec in /usr/stand, and you want to boot it directly, you must install a boot block in the /usr partition. To do this, use the following sequence:

```
% su
Password: enter super-user password
# cd /usr/mdec
# installboot bootdisk /dev/rpartition
# cp /boot /usr
```

disk is the disk controller type and *partition* is the disk partition `/usr` is on; it is usually `0g` for the first disk, g partition.

For example, installing a bootblock on the g partition of disk 0:

```
# installboot bootxy /dev/rxy0g
# cp /boot /usr
```

Servers vs Local Disk and Tape

There are two logical places to install the Exec: on the local disk of the system you want to test (if you don't plan to download it), or on a server from which you want to remotely download the Exec.

Local Disk and Tape

To install the Exec on the test system, first determine whether it is a Sun-2 or a Sun-3. Look at the model number on the workstation to determine this. If you aren't sure, ask your system administrator.

Loading the Exec from tape is the same for Sun-2s and Sun-3s. First you must load the file called `extract_exec` into the `/stand` directory. The example below applies to 1/2-inch tapes. For 1/4-inch tapes, substitute **st** for **mt** in the examples below. Do the following:

```
example% su
Password: enter superuser (root) password

example# cd /stand
example# mt -f /dev/nrmt0 rewind
example# mt -f /dev/nrmt0 fsf 3
example# tar xvf /dev/nrmt0
```

Once `extract_exec` is on the disk, run it by doing the following:

```
example# extract_exec mt0
you will see a number of messages here..
example# exit
example%
```

All of the files on the tape should now be copied onto `/stand`.

server

Installing a server with the Exec is a little more involved. First, go to the `/` (root) directory of the server. When the Exec boots from `/stand` on a server, it is really booting from `/pub/stand`, which can be either `/pub.MC68010/stand` (Sun-2) or `/pub.MC68020/stand` (Sun-3). Use these pathnames to put the files in the right place.

Put the Exec and the Diagnostics in both `/pub.MC68010/stand` and `/pub.MC68020/stand`.

If your server is homogeneous, it will only have one of the two `pub` directories. Put the Exec and diagnostics into that directory.

Loading the Exec from tape is the same for Sun-2s and Sun-3s. First you must load the file called `extract_exec` into the `/pub.MC680X0/stand` (X is 1 or 2) directory. Do the following, using **st** for SCSI tape and **mt** for 1/2-inch tape:

```
example% su
Password: enter superuser (root) password

example# cd /stand
example# mt -f /dev/nrst0 rewind
example# mt -f /dev/nrst0 fsf 3
example# tar xvf /dev/nrst0
```

Once `extract_exec` is on the disk, run it by doing the following:

```
example# extract_exec st0
you will see a number of messages here..
example# exit
example%
```

All of the files on the tape should now be copied onto `/stand`.

Remote Tape

If you are using a remote tape drive to install the SunDiagnostic Executive onto disk, the `rpc.rexd` daemon must be running on the remote device. If it is not, you will see the error message:

```
cannot connect to server
```

Refer to `REXD(8C)` in the *SunOS Reference Manual* (''man'' pages).

Use this sequence to perform the remote tape installation. You may want to read the entry in the *SunOS Reference Manual* for the `on` command, prior to using this command sequence.

```
cd target_directory

on tapeserver mt -f /dev/nrst0 rewind
on tapeserver mt -f /dev/nrst0 fsf 3
on tapeserver tar xvf /dev/nrst0
on tapeserver csh extract_exec st0
```

Use `st` for SCSI tape or `mt` for 1/2-inch tape, and replace *tapeserver* with the name of the system that has the tape drive from which you want to load the Sun-Diagnostic Executive.

Booting from Disk

To boot the Exec from disk — either locally or across the Ethernet — copies of the Exec and the diagnostic programs must be available in a `/stand` or `/pub/stand` directory where they can be booted. Sun-3 machines can be configured so the Exec will boot automatically when the machine is powered-on or reset with the CPU diagnostic switch on the ON position. Setting this configuration requires programming the system's EEPROM, for which you may use the EEPROM editing tool described in *Chapter 6.*

Autoboot

If you have a Sun-3 configured to autoboot the Exec with the normal/diagnostic switch in the diagnostic position, do the following:

1. Halt the system (see the *Halting your System* section).
2. Turn off the power to your system.
3. Set the normal/diagnostic switch on the system from [NORM] to [DIAG]
4. Power up the system. The Exec should boot automatically.

You can also start the Exec by booting the program `/stand/exec` from the PROM monitor manually, as follows:

Booting from Local Disk

1. Halt the system as described in *Halting your System.*
2. Perform a **k2** reset.
3. Boot the Exec by typing the following to the PROM monitor:

```
> b stand/exec
```

Booting from Remote Disk

NOTE *Before you shut down your system, read this entire section. You may need to write down some internet numbers from your* `/etc/hosts` *file to boot the Exec.*

1. Halt your system and start the PROM monitor as previously described.
2. Boot the Exec by typing the following to the PROM monitor:

```
>b le(0,X,0)/stand/exec  (for Sun-3/50s and 3/60s)
or
>b ie(0,X,0)/stand/exec  (for all other Sun systems)
```

where X is the hexadecimal host number of the server that has the Exec on disk. (if your server has the Exec on it, see shortcut, *below)*

NOTE *The Exec lives in* `/stand`*; the path to it is usually* `/stand/exec`*. In the case of a client on a server, it lives in* `/pub/stand`*, but still boots as described above.*

The *host number* tells the PROM monitor what server you want to boot the Exec from. To find the number, look in your `/etc/hosts` file for the server name you are using.

```
199.5.0.135 Execserver   the Exec server to boot from

199.5.0.155 example      the machine being booted
```

The string of numbers separated by periods (.) is the *internet address*. The number after the last dot (the right end of the number) is the number you want. This number must be converted to hexadecimal (base 16) before you can use it in the boot command above. The server you use must be on the same network as the workstation you're booting (i.e. the rest of its internet number, the part to the left of the last dot, must exactly match). In the example above, the host number of the server is 135 decimal. So to boot, we would use 0x87, which is 135 converted to hexadecimal.

shortcut

If the disk server of your test machine has the Exec in its `/stand` directory, you can save yourself some work. Just type the following from the PROM monitor to boot the Exec:

```
>b /stand/exec
```

This command works because your disk server is the default machine to boot from. You only have to give an internet number if you must boot from a different machine.

Invoking a Script File

If you have written a script file (described under the *Main Menu* heading), you may invoke it when you boot the Exec instead of using the `SC` command from the main menu. To boot a script file, do the following:

>b *device* **() /stand/exec SOURCE=***filename*

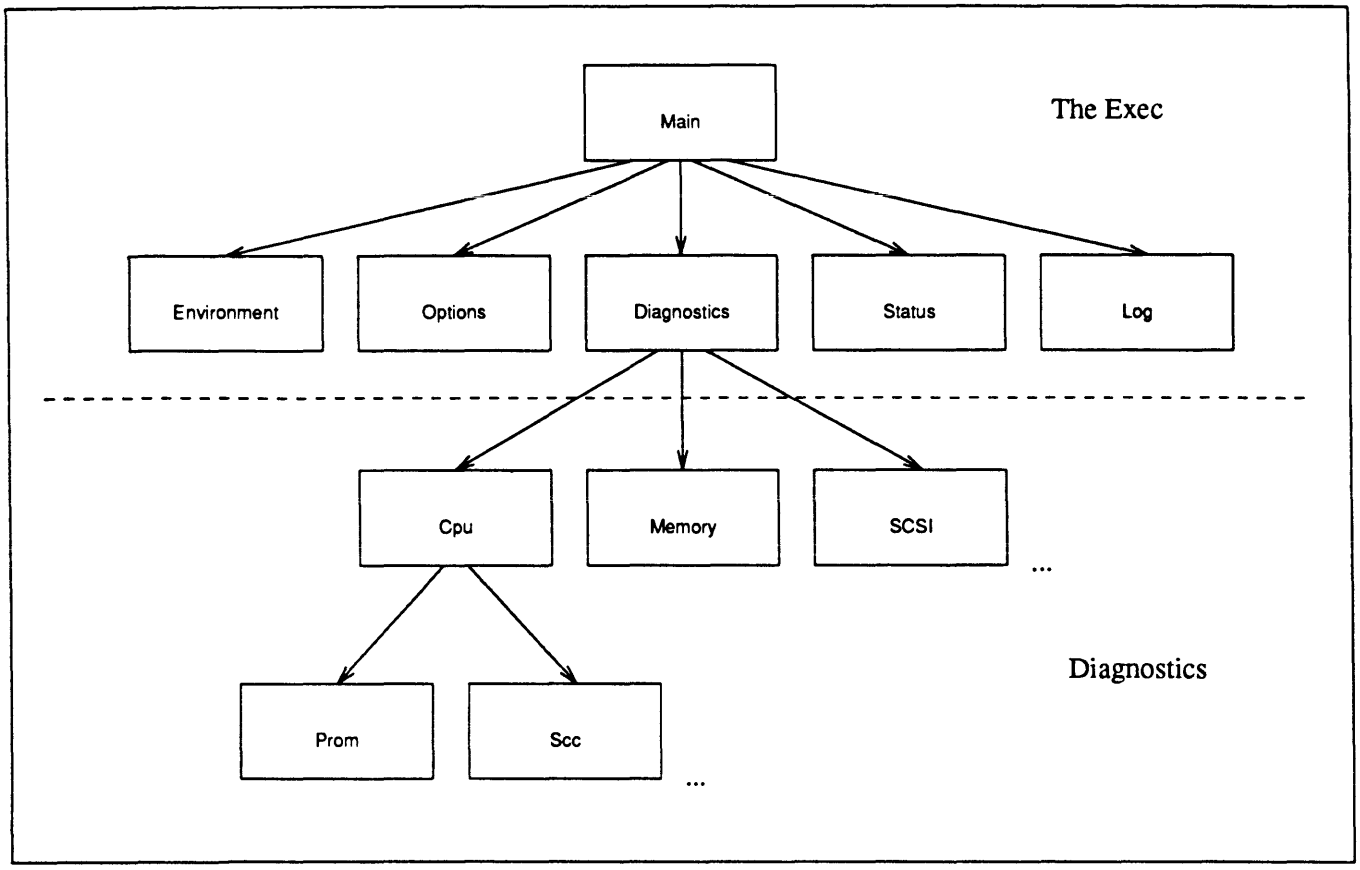
Replace *device* with the boot device designator, such as `ie` for Ethernet, and so on. Replace *filename* with the name you have given to the script file.

2.5. The Exec Environment

There are two ways of looking at the Exec; as a series of menus, or as a set of separate programs running under an operating system. You need to understand both perspectives in order to use the Exec effectively.

The Menu Perspective

The Exec and Diagnostic programs appear as a series of menus. These menus are arranged in a tree structure. The top of the tree is a single menu, called the Main Menu. Five menus branch out below this menu. Still more menus branch out below this layer, and so on, until you get to the last layer at the bottom of the tree. It is similar to the hierarchical structure of UNIX file systems. Here is a diagram of the Exec and Diagnostic Menu tree:



The Exec itself consists of the Main Menu and the layer of menus below it. All menus below the "Diagnostics" Menu are the diagnostic programs themselves. When the word "sub-menu" is used in this document, it is a relative term — it refers to any menu below the one we are currently "in". The the "current" menu is the one displayed on the screen. Commands and the immediate "sub-menus" are listed as selection items in the current menu.

Moving Around

You can move up or down the menu tree. To move down, select the sub-menu you want in the current menu. This makes that sub-menu the new current menu. You can continue down the tree until you get to the menu or command you want. To go up the tree, press the escape key (Esc). This will put you in the menu one level above, making it the "current" menu. Moving up from the Main Menu would mean leaving the Exec; so the escape key doesn't work in the top menu. Use the `BOOT` command to exit the Exec.

The Operating System Perspective

The menu viewpoint is an adequate one for using the Exec — if you run only one diagnostic at a time, don't use the at-sign (`@`) command or do anything in the status menu, you'll be fine. However this approach doesn't give you the full power of the Exec.

Beneath the menu driven interface, the Exec is a multitasking operating system. Its purpose is to run diagnostic programs. With it, you can run many diagnostics at once. Each time you enter a diagnostic menu from the Diagnostic Menu in the

Exec, you start a new diagnostic program. If you exit the top menu of a diagnostic by using the escape key [Esc], you terminate it.

If you leave a diagnostic using the at-sign command (@), you return to the Diagnostic Menu, leaving that diagnostic running, and are free to start another one at the same time. The menu items in the Status Menu list all the diagnostic programs currently running. Using the at-sign command is similar to using the `bg` (background) command in the operating system; the diagnostic continues running in the background, but you are back in the Exec, ready to run another one. You can have up to ten diagnostic programs running at once, nine of them running in the background.

NOTE *In this release, you can't have more than one copy of the same program running simultaneously.*

To bring one of the background diagnostics back into the foreground, go to the Status menu, and select the process number of diagnostic you want. That diagnostic will now take over the screen. You are back in that diagnostic. You can run other tests, move around the menus in that diagnostic, or do anything else you could normally do. If you want to leave that diagnostic again, press the at-sign (@) key again; you will be back in the Status Menu.

NOTE *When starting multiple tests, start those dealing with the boot path last; otherwise you will be unable to load all tests specified.*

There is an exception to the comments above. If you run a set of commands simultaneously, using the semicolon (;) separator, each command will be run in order. Each command waits for the previous one to complete. For example:

```
Command ==> command1 ; command2 ; command3
```

In this example, command2 won't start until command1 completes, and command3 won't start until command1 and 2 complete. Lets say you just started this command string. Command1 is running. If you press the at-sign key now, command1 will go to the background, as expected. However command2 has been waiting for command1 to finish. Putting it in the background makes it look like it's finished to command2, so it starts executing. If you press the at-sign key again, command1 and 2 will be in the background, but command3 will start running. Finally, pressing the at-sign puts the final command in the background, and returns you to the Status Menu.

In general, if you have multiple commands, separated by spaces on one line, pressing the at-sign key will only background the currently running one. You must press the at-sign once for each program, or wait for the ones still in the foreground to complete normally before returning to the Exec.

Whether its in the foreground or the background, running diagnostics all save their log messages in the logfile. Not all messages printed by the diagnostic are log messages. Only log messages are saved. The other messages can only be seen if the program is in the foreground, and will be lost when it gets scrolled off the screen. See the Log Menu section for details.

2.6. Using the Network Console

To run the Exec remotely over Ethernet, the `netcon` program must be installed on a system that will be running the SunOS operating system. This program is on the distribution tape and may be installed wherever you normally install UNIX Executables. Once this program is is installed, the Exec can be run remotely over Ethernet. To do this, first do the following to prevent double echoing and allow transmission of single characters without pressing Return.

```
%stty -echo cbreak Return
```

Do not run more than one netcon session at a time.

Then, start the `netcon` program by typing

```
%netcon Return
```

Netcon then prints the following message:

```
netcon: waiting on nnn/nnnn
```

where nnn/nnnn is some number.

Netcon is now waiting to connect to a system running the Exec. Now go to the system running the Exec and from the Exec's environment menu turn the network console on by typing **`network=on`**.

To disconnect the network console, type `network=off` on the Exec's environment menu, then type control-C on the network console.

After terminating `netcon`, perform the following to reset normal conditions:

```
%stty -cbreak
```

2.7. User Interface

The user sees the Exec as a series of menus. These menus are arranged in a tree-like structure. The diagnostic programs themselves are integrated into the menu tree. The system is designed this way so that entering commands and reading results are the same, whether you're in the Exec part of the tree or in a diagnostic subtree.

NOTE *The user interface looks and acts the same in both the diagnostic menus and the Exec menus (except for the at-sign command `@`).*

Menu and Invisible Commands

There are two types of commands available when you run the Exec; the commands listed in the menu you are in (the *visible* or *menu* commands), and a fixed set of commands which you can use regardless of the menu. These commands are always available, but not shown on the screen; the *invisible* commands.

Menu Commands

The menu commands are the list of choices shown on whatever menu you're in. A menu line will show a menu command, followed by a short description of the command. A menu command is a single word. For commands that set a system parameter, the word has an equals sign at the end. This character must always be included when typing the command, even if the command word itself is abbreviated (see Command Line Syntax, below). At the bottom of the menu will be a prompt that looks like this:

```
Command ==>
```

To run a menu command type its name at the prompt, followed by arguments (if needed). When you type a [Return], the command will be Executed. You can only enter the commands displayed on the menu you're in.

To make things easier, menu commands are designed so you only have to type enough of the command name so the Exec can recognize it. This means you only have to type the letters capitalized in the command name. In some cases this will be the entire word; most of the time it's not, however. You are free to type in more of the word if you want. For example, one line of the Environment Menu is:

```
LOGfile=    Log to file     currently: off
```

To Execute this command you could type any fraction of the word "logfile", as long as you start from the beginning of the word and include the letters that are capitalized in the menu:

```
Command ==>log=off        this is the minimum
Command ==>logf=off
Command ==>logfi=off
etc...
Command ==>logfile=off    this is the maximum
```

NOTE *If a command word is displayed in a menu with an equals sign (=), make sure you include that equals sign at the end of the command when you type it, even if you abbreviate the rest of the word.*

All commands work the same, you don't have to type the entire command word, just the capitalized part. You still must type in any parameters correctly, though.

You can type the commands in any mixture of upper or lower case; the Exec has no preference. Menu commands do one of two things: Execute a command directly, or move you down to a sub-menu containing other commands. Commands that move you to sub-menus have no arguments; you only have to type the command word.

Invisible Commands

Invisible commands are a set of single letters that can be typed from any menu in the diagnostic tree. These commands are **not** listed in the menu. None of the invisible commands take arguments. They enable you to move around the menu tree quickly, obtain help information, or re-execute a previous command. The invisible commands are described below.

Esc

Pressing the escape key [Esc] from any menu moves you up one level to the menu above. This command does nothing in the Main Menu. If you want to exit the Exec from the main menu, use the `BOOT` command.

@

Pressing the at-sign character [@] returns you to the Exec from a diagnostic, leaving it running. It has no effect while you are in the Exec (i.e. in the Exec menus). It returns you to the last Exec menu you were in; either the

Diagnostic or the Status menus. The diagnostic you left is still running in the background; to return to it, go to the Status Menu and select the diagnostic from the process list. For more information, see previous section, *The Operating Systems View.*

!

Pressing the exclamation point [!] displays the last 5 command lines you entered in a numbered list. It acts like a limited version of the UNIX C-Shell's `history` command. You can re-execute any line on the list by typing number listed with it.

1, 2, 3, 4, or 5

Pressing any of the numbers [1] through [5] allows you to re-execute one of the last five menu commands you executed. `5` is the current command, `1` is the oldest command available. Use the `!` command (described above) to list the last five commands entered.

?

Pressing the question mark [?] from any menu displays the help list for the current menu. Sometimes, there is more detailed help available; if the items in the help list are lettered, entering the letter causes the Exec to display an even more detailed help list specifically related to that item.

Command Line Syntax

A command line is formed of one or more commands, each separated by semicolons (;). For example, a command line with the single command "command1" would look like this:

```
Command ==>command1
```

A command line with three commands in it is shown below; the spaces are optional.

```
Command ==>command1 ; command2 ; command3
```

Command Parameters

Some commands may need parameters; they are listed in the menu after the command itself. Parameters are entered two ways: if the command name has an "equals" sign (=) at the end, the parameter is typed immediately after the command word — no spaces are allowed. If the command doesn't have an "equals" sign in it, type one or more parameters after the command, separated by spaces. For example:

```
Command ==>mem all          command name mem has no equals sign
Command ==>source=/stand    command name source= has an equals sign
```

There is one more way to list parameters, used when you are calling diagnostic programs from the Diagnostic Menu. You can selectively call sub-commands as arguments to your command by encasing them in quotes. For example, the `cpu` command brings up a menu of subcommands that run cpu tests. Among the tests listed in this sub-menu are the `scc` and `clock` commands. If you wanted to run just these tests without going to the `cpu` menu, enter the following command:

```
Command ==>cpu "scc ; clock"
```

Note that the commands in quotes are treated as a parameter to the `cpu` command. Also note that the `scc` and `clock` commands are separated by a semicolon; type in commands in quotes as if they were being typed on a separate line. The quoting syntax is a shortcut; you can always do the same thing by typing each command, in sequence, on separate lines. The quoting syntax only works in the Diagnostic Menu or its sub-menus. The menu hierarchy is explained in more detail in the following sections.

All of the methods mentioned above can be mixed and matched in different ways. Here is an example including each:

```
Command ==>cpu "scc ;clock" ; mem all addr=0 size=100000;video
```

Numerical Parameters

Many of the parameters passed to commands are numbers. The Exec automatically selects either decimal (base 10) or hexadecimal (base 16) numbers for command parameters depending on the type of parameter. You can override the Exec's choices by starting or ending a number with `%o` for octal numbers, `%d` for decimal, `%h` for hexadecimal, or `%b` for binary. The numbers you enter must conform to the standards for the base you select; for example, the string `%d1F` or `1F%d` will generate range errors, since "1F" is not a decimal number. It should be `%h1F` or `1F%h`.

Special Characters

The Exec recognizes three special characters: `*`, `-` and `;`.

`*`

The asterisk `*`) character represents the highest possible numerical value, infinity for all practical purposes. You can use it anywhere the Exec expects a parameter. For example, when you enter a `*` for the number of times the test will run (called a passcount), it causes a command to repeat virtually forever (or until you stop it!). The actual value of `*` is 0x7FFFFFFF, hexadecimal.

`-`

The hyphen character (`-`) can be roughly translated as "through", as in "a through d". For example, on a menu with choices a, b, c, and d, entering `a-d` executes them all in sequence. A note of caution here: the hyphen character works with the menus. It will execute all of the commands in the order they are listed in the menu, **not** necessarily in alphabetical order. For example, if your current menu looks like this:

```
      Sample Menu

A     Command A
Q     Command Q
B     Command B
S     Command S
R     Command R
G     Command G
```

Executing the command

```
Command ==>Q-S
```

will execute the commands `Q`, `B`, `S` in that order. You can't have any white space between the hyphen and the command names.

;

The semicolon character (**;**) as mentioned earlier, separates commands for the Exec, allowing you to enter several commands on the same line. For example, the string

```
a; b; c
```

would be interpreted as three separate commands, not a command with two options. The spaces between items are optional.

2.8. Exec Menus

The following sections of this chapter describe each of the five Exec menus in detail. For information on diagnostic programs or their menus, see the remaining chapters of this document.

The Main Menu

The Main Menu is at the top of the menu hierarchy. It is the first menu you see when you start up the Exec. This menu contains seven commands; the first five give you access to other sub-menus, while the last two, `SCript=` and `Boot`, are commands that are executed directly.

NOTE *When a specific group of diagnostics have been called up — when the* `All` *or* `Default` *commands are used, for example — entering a* `Control-C` *sequence aborts the current diagnostic and proceeds to the next diagnostic in the group. Using* `Control-C` *when a specific group of diagnostics is not specified returns you to the Diagnostic Main Menu.*

```
SunDiagnostic Executive Rev:x.x dd/mm/yy Main Menu

Environment     Set Executive environment
Options         Set global diagnostic options
Diagnostics     Available diagnostics
Status          Display task status
Log             Display error log
Script=         Source a script file
Boot            Exit and boot another program

Command ==>
```

`environment`

Selecting this command moves you into the Environment Menu of the Exec. Go to this menu to configure the Exec to its operating environment. See The Environment Menu section for details.

`options`

Selecting this command moves you into the Options Menu of the Exec. Go to this menu to set the global diagnostic test options. See The Options Menu

section for details.

diagnostics
Selecting this command moves you into the Diagnostic Menu of the Exec. Go to this menu to run the diagnostic tests. See The Diagnostic Menu section for details.

status
Selecting this command moves you into the Status Menu of the Exec. Go to this menu to see to output of the currently running diagnostic tests and run particular tests in the foreground. See The Status Menu section for details.

log
Selecting this command moves you down to the Log Menu of the Exec. Go to this menu to read the entire output of a particular diagnostic test, as recorded in the logfiles. See The Log Menu section for details.

script=*scriptfile*
Selecting this command will cause the Exec to read commands from the file named after `script=` instead of from the console. The *scriptfile* file will contain a sequence of command lines, arranged in a script, to be run by the Exec. This command allows you to run a predefined sequence of tests automatically. When the Exec runs to the end of the file, control returns to the user.

The script file is created prior to running the Exec and may exist on any loadpath exclusive of a tape device. Basically, the commands are entered just like they would if you were running all the diagnostics from the Diagnostic Main Menu. However, the current implementation of the script processor imposes some notational requirements on the script writer, as enumerated below.

1. The escape character should be typed out as **<esc>**.
2. The **d;** (entering the Diagnostic Main Menu) must exist alone with nothing after it. All subsequent commands for the diagnostic menus must exist on the next line:

```
d ;
ether cmd="quick;<esc> d"
```

The script processor has problems dealing with CR/LF combinations.

The following is a script to run four diagnostics consecutively.

```
d ;
ether cmd="quick;<esc> d" ; cpu cmd="quick;<esc>" ; video cmd="d;<esc>";mem cmd="quick;<esc>" ;
```

The example above first selects the Diagnostic Main Menu. The next line selects the Ethernet "quick" test sequence, then escapes back to the main menu. After the second semicolon, the CPU quick test is selected from the main menu, and then the script escapes to the previous menu (the main menu) and selects the Video diagnostic Default test, followed with an escape to the Memory quick test and finally an escape back to the Diagnostics Menu. The *Command Line Syntax* section at the beginning of this chapter provides more information on the use of quotation marks and semicolons in command lines.

The following is an example of a script to run specific diagnostics from the diagnostic sub-menus:

```
d;
cpu cmd="e;quick;<esc>;f;quick;<esc>;<esc>"
```

In this case the script selects the CPU Diagnostic main menu and then, with the `e` entry selects the System Enable Test sub-menu. The quick test sequence is then selected from that menu, followed with an escape back to the CPU Diagnostic main menu, from which a second sub-menu is selected with the `f` command, from which another quick test sequence is invoked. Finally, the two escapes bring you up through the CPU diagnostic main menu to the SunDiagnostic Executive Main Menu.

3. While reading scriptfiles, the type-ahead buffers are not used. If you are writing scripts, you must start concurrent tasks differently. Start a background task using the `cmd=` and `bg` options. See the *Diagnostic Menu* section in this chapter, for details.

`boot` *bootfile*

Selecting this command causes the Exec to exit and the *bootfile* to be booted on the system. Running `boot` without any arguments restarts the Exec; use it if the Exec is malfunctioning and can't be fixed any other way. At this time the only argument that the `boot` command accepts is this:

```
boot exec
```

The Environment Menu

The commands in the environment menu allow you to configure the Exec to its operating environment. In this menu you can: identify the machine and directory where the Exec code is stored; tell the location and characteristics of the control terminals that the Exec takes its commands from; and control diagnostic output to the logfile.

```
SunDiagnostic Executive  Rev:x.x   dd/mm/yy   Setup Menu


Load=         Change load path            currently:  boot path
LOGfile=      Log to file                 currently:   off
TTYA=         TTY A console               currently:   off
TTYABaud=     TTY A baud rate             currently:   9600
TTYATerm=     TTY A terminal type         currently:   adm
TTYB=         TTY B console               currently:   off
TTYBBaud=     TTY B baud rate             currently:   1200
TTYBTerm=     TTY B terminal type         currently:   ansi
NETwork=      Network console             currently:   off
NETTerm=      Network console type        currently:   ansi
Default       Assign default values to all environment flags

Command ==>
```

load=*loadpath*

After the Exec is booted, it selectively loads diagnostic programs as they are required by the user. It uses the *loadpath* variable to determine where (device and directory) to load from. Upon booting, the *loadpath* variable is automatically initialized to the directory the Exec was booted from. You will only need to change *loadpath* if you need to load the diagnostic programs from a place other than the place from which the Exec was booted.

The loadpath consists of two parts; the storage device and the pathname. The storage device tells the Exec where to look for the pathname. It is a device name. It can be a disk, tape or Ethernet. The devices supported for this release are given in the table.

Disks	*Tapes*	*Remote*
sd() xy() xd()	st()	ie() le()

Combine the device name with the directory path on the device to make *loadpath*. End the loadpath with a forward slash (/).

if:
device = sd0 -> sd()
directory = /stand

then:
`loadpath=sd()/stand/`

logfile=*enableflag*

This parameter enables or disables the logging process. To enable logging, the *enableflag* should be `on`. If the Exec is not logging messages, it loads the current logfile into the RAM log, then starts logging all new messages to both the RAM and the disk logfile. To disable logging, enter `off`. If the Exec is already logging messages, that procedure stops, and any new messages are discarded.

In this release, the log is always saved in a file called `logfile` in the directory indicated by the `loadpath` variable on the test system's **local** disk. You cannot save a logfile on a remote disk. You must create the file, with at least 32K bytes of data in it, **before** you run the Exec.

ttya=*enableflag*

This command tells the Exec whether you are using a terminal connected to the ttya serial port on the test system. If you are, the *enableflag* should be the string `on`. If you're not, use the string `off`. The Exec is controlled from one or more sources, called consoles. The console default is the system keyboard and screen. Setting `ttya`, `ttyb` or `network` to `on` allows these devices to act as consoles. Any enabled device will automatically act as a console when it starts receiving characters through its port.

ttyabaud=*baudrate*

This command sets the baud rate for the `ttya` port on the test system. You can use any of baud rates in the table below:

Baudrates	
300	2400
600	4800
1200	9600

Enter one of these rates for the *baudrate* parameter. The default rate is 9600.

ttyaterm=*termtype*

This command sets the terminal type the Exec expects to be connected to the ttya port. The legal values are `ansi`, `adm`, or `tty`. The table below shows when to use each:

Use	*If your terminal is:*
ansi	VT100, Sun workstations, any other ansi
adm	ADM, TVI925, Wyse
tty	Any other terminal or uncertain of type

`tty` contains no escape sequences, so it will work on nearly any terminal, display or printer.

`ttyb=`*enableflag*

This command tells the Exec whether you are using a modem connected to the ttyb serial port on the test system. If you are, the *enableflag* should be `on`. If you're not, enter `off`. The Exec is controlled from one or more sources, called consoles. The console default is the system keyboard and screen. Setting `ttya`, `ttyb` or `network` to `on` allows these devices to act as consoles. Any enabled device will automatically act as a console when it starts receiving characters through its port. The `ttyb` port responds to modem signals as well as data, letting you control the Exec from a phone line.

`ttybbaud=`*baudrate*

This command sets the baud rate for the `ttya` port on the test system. You can use any of baud rates shown for `ttyabaud`. Enter one of those rates for the *baudrate* parameter. The default rate is 1200.

`ttybterm=`*termtype*

This command sets the terminal type the Exec expects to be connected to the ttyb port. The legal values are `ansi`, `adm`, or `tty` as shown for `ttyaterm`. `tty` contains no escape sequences, so it will work on nearly any terminal, display or printer.

`network=`*enableflag*

This command tells the Exec whether the console is on a remote machine over the Ethernet. If it is, *enableflag* should be `on`. If it isn't, enter `off`. The Exec is controlled from one or more sources, called consoles. The console default is the system keyboard and screen. Setting `ttya`, `ttyb` or `network` to `on` allows these devices to act as consoles. Any enabled device will automatically act as a console when it starts receiving characters through its port.

The remote console should be running on the network **before** this flag is enabled. To do this, find a Sun workstation running on the same network that the system under test is on. Enter `stty -echo cbreak`, as described in the *Using the Network Console* subsection, then run the program `netcon` under the SunOS operating system.

`Netcon` is included on the Exec release tape, and should be in the /stand directory of any system that contains the Exec system. Under netcon, the screen will act like an ansi terminal. Now, if the `netterm=` variable is set in the Exec, the remote workstation can control it.

Don't forget to run the `stty -cbreak` command when `netcon` is terminated, to reset normal conditions.

CAUTION **In this release, if you enable `network=` without a network console running, the Exec will "freeze" looking for it. The only way to break out is to start up a network console, or to cycle the power on the test system.**

`netterm=`*termtype*

This command sets the terminal type the Exec expects to be connected to the The other end of the network. The legal values are `ansi`, `adm`, or `tty`. Since the "terminal" on the other end will most likely be a Sun, ansi is the most common setting. The table in the section that describes `ttyaterm` shows all of the terminal selections and when to use them. `tty` contains no escape sequences, so it will work on nearly any terminal, display or printer.

`default`

This command sets all of the values in this menu to their defaults. The default values are listed in the table below:

Command	*Default*	*Command*	*Default*
`Load=`	*boot path*	`ttyb`	off
`logfile=`	off	`ttybbaud`	1200
`ttya`	off	`ttybterm`	ansi
`ttyabaud`	9600	`network=`	off
`ttyaterm`	adm	`netterm=`	ansi

The Options Menu

The commands in the Options Menu control how the diagnostic tests will react to errors. These commands allow you to configure some characteristics of all of the diagnostics at once.

Two behaviors are controlled from this menu: how a test responds when it finds a hardware error; and how many times it runs. These behaviors are controlled by the option variables in the menu. Change the variables and you change the behavior of all the diagnostics. These variables always show the current option state.

```
SunDiagnostic Executive Rev:x.x  dd/mm/yy  Options Menu

STop=        Stop on nth error          currently: *
SCope=       Scope loop on error        currently: off
SOft=        Soft error retry count     currently: 0
Pass=        Pass count                 currently: 1
Default      Assign default values to all options

Command ==>
```

stop=*numb_errors*

This command controls the number of times a test will detect an error before stopping. The parameter *numb_errors* is a decimal number or the metacharacter *. Entering * means "keep testing no matter how many errors you see". Whether or not a test stops, it will always log any errors it finds into the Exec's error log.

scope=*enableflag*

This command determines whether a test will run a scopeloop if it detects an error. Entering `on` for *enableflag* causes the test to scopeloop; entering `off` makes it continue the test. This setting takes precedence over the `stop=` parameter; if `scope=` is on, the test will scopeloop no matter what the `stop=` setting may be.

A scopeloop is a write or read cycle repeated endlessly. It is used with an oscilloscope or logic analyzer to isolate hardware bugs. Only diagnostics that have a scopeloop test in their menus will scopeloop if this parameter is set.

soft=*numb_trys*

This command controls the number of times a test will detect a soft error before stopping. A soft error is a temporarily incorrect value found in a storage area (it can be in RAM, disk, registers, etc.) Soft errors are not as serious as hard errors, and if they don't happen too frequently, are often tolerated. What constitutes an unacceptable soft error rate is a matter of judgment. Refer to your test procedures for guidance.

The parameter *numb_trys* is a decimal number or the metacharacter *. Entering * means "keep retrying no matter how many soft errors you see". Whether or not a test stops, it will always log any errors it finds into the Exec's error log.

pass=*numb_tests*

This command sets the default for the number of times a test will run before exiting. This is the number of times the test will run if it finds no errors. The number of times a test will run can change if it encounters errors, as determined by the `stop=`, `scope=`, and `soft=` parameters and the number and type of errors encountered. The *numb_tests* parameter can be a decimal number or the metacharacter *****, which means "keep running the tests over and over without stopping". Since this value is only a default, it is overridden when a command is entered with a passcount argument.

default

This command sets all of the values in this menu to their defaults. The default values are listed in the table below:

Command	*Default*	*Command*	*Default*
`stop=`	*	`soft=`	0
`scope=`	off	`pass=`	1

Diagnostic Menu

This menu gives you access to the diagnostic programs. Typing a command in this menu moves you to the main menu of the corresponding diagnostic program. Since different machines have different hardware and thus need different tests, the diagnostic menu doesn't have a fixed set of commands. This menu varies depending on the diagnostic programs the Exec has loaded — the menu shown here is only an example. There is one command on this menu that doesn't change; that is the `Default` selection. Running this command will run all of the tests in the menu with their default parameters.

NOTE *Make sure you have configured the* `/stand/diags` *file (while running the operating system) before you run the* `Default` *command. The* `diags` *file is a list of the tests and menu entries. You need to remove both the menu and program name entries that pertain to hardware not available in the system under test.*

Since the Exec is multitasking, you can run more than one test program at a time. Read the *Operating System Perspective* and *Command Syntax* section for details on running jobs in the background.

NOTE *When starting multiple tests, start those that deal with the boot path LAST or you will be unable to load all the tests you have specified.*

```
SunDiagnostic Executive   Rev:1.x    dd mm yy    Diagnostics Menu

C2                Sun-2 Color Board Diagnostic
C3                Sun-3 Color Board Diagnostic
CPu               Sun CPU Board Diagnostic
Ether             Ethernet Diagnostic
EEprom            EEPROM Editing tool
Fpa               Sun-3 FPA Diagnostic
Gp                Graphics Processor 1 and Graphics Buffer Diagnostic
Kb                Keyboard Diagnostic
Mem               Memory Diagnostic
MCp               MCP and ALM-2 Diagnostic
MOuse             Mouse Diagnostic
MTi               MTI/ALM Diagnostic
SUBsystem         SCSI Subsystem Diagnostic
SMd               SMD Subsystem Diagnostic
Tape              1/2" Tape Diagnostic
Video             Sun-x Video Diagnostic
VIDMon            Video Monitor Diagnostic
VME               Sun-3 VME Interface Diagnostic
Default           Start all of the above diagnostics

Command ==>
```

NOTE *When a specific group of diagnostics have been called up — when the* `All` *or* `Default` *commands are used, for example — entering a* `Control-C` *sequence aborts the current diagnostic and proceeds to the next diagnostic in the group. Using* `Control-C` *when a specific group of diagnostics is not specified returns you to the Diagnostic Main Menu.*

Starting a Diagnostic

Entering the name of a diagnostic will put you in that program's main menu. The diagnostics are designed to all have a similar user interface. Each top menu will probably have:

- A "Run all Tests" command — runs every test in the diagnostic.
- A "Run quick Tests" command — runs a subset of all the tests, which completes in 2 minutes or less.
- A "Run default Tests" command — runs the most important tests in the diagnostic
- A set of test sub-menus, covering different areas of the hardware.
- A set of utility and debugging commands, including scopeloops.
- A set of option commands for configuring the particular diagnostic to its operating environment.

In addition to the commands visible in the menus, the invisible commands, mentioned earlier in this chapter, are also available.

A diagnostic may be explicitly given a command to run when it is started. This feature is used when writing script files. For example, to run all the commands in the CPU diagnostic, you normally type **cpu**, wait for the prompt, then type **all**. In a script file, you would type instead:

```
cpu cmd=all.
```

The `bg` option may be added to run the diagnostic in the background. To run the example above in the background, type

```
cpu cmd=all bg
```

The CPU diagnostic will start running in the background, but you will remain in the Exec. This feature has little use outside of scriptfiles, since you can use the at-sign (`@`) to background diagnostics.

default

Running the default command from the diagnostics menu will run all of the diagnostics shown in the menu with their default tests.

Status Menu

This menu lists all of the diagnostic tests currently running under the Exec. This menu is highly variable; it depends entirely on what's running at the present moment. Each test is displayed as a menu item, along with the *process number* the Exec has assigned to it. You can use the process number to reenter individual diagnostics. Here is an example Status Menu:

```
SunDiagnostic Executive Rev:1.x      25 Sept 1987      Diagnostics Menu

procn  vme

procn  rtest

procn  ktest

Command ==>
```

procn

Entering the process number of a diagnostic puts you back in that diagnostic. You will be in the same menu that you left; if a test is running, you will see its status and error messages on the screen. To leave the diagnostic, enter the "at-sign" (@). You will return to the Status Menu. The diagnostic you left will still be running in the background. You can only foreground one test at a time.

@

Entering an at-sign character (@) will stop a test from displaying on the screen and start it running in the background. The current status menu is redisplayed. This command has no effect when there are no tests running in the foreground.

Log Menu

This menu deals with viewing and controlling the current log file. The Exec collects all of the error messages from all of the tests and writes them into a log file. The log file is resident in memory. The commands here allow you to view the current log file, save it to disk, erase it, and turn it on or off.

The log messages themselves have a fixed format. They are made of five parts; *testname*, *timestamp*, *error_number*, *error_location*, and *error_info*. The first two parts are supplied by the Exec. *Testname* is the name of the test that failed. *timestamp* is when it failed; in this release, it is the number of seconds after the test started. The last three parts are supplied by the diagnostic itself. In future releases, the *error_number* will be used to look up a description of a particular error in this manual. *error_location* describes what part of the test failed. The *error_info* section provides more information about the particular error. For example:

```
mem3.diag:943578 9 - Address Test - Loc 0x9456 Exp 0x0 Obs 0x1
```

In this example, the mem3 test failed with an error 9, in the address test. It expected a 0 and read a 1 instead. A description of error 9 would be in the appendix of the mem3 diagnostic chapter.

Here is the Log Menu:

```
SunDiagnostic Executive Rev:x.x  dd/mm/yy  Log Menu

Display     Display log
Clear       Clear log
Save        Save log to log file
Logfile     turn logfile on or off

Command ==>
```

display

Running the display command prints the current log file onto the screen. The display runs with character flow control; you can freeze the display by typing **^S** and continue it with **^Q**. The **^** symbol means Control. To type a **^Q**, hold down the control key, type the Q, then let go of both keys. The procedure is the same for **^S**.

clear

Running the `clear` command erases the current logfile in RAM. If the `logfile=` parameter is set to "on", the logfile on disk will also be cleared.

Clearing the file does not stop new messages from being accumulated. If diagnostics are running, the log will immediately start to refill as log messages are generated.

save

Running the save command saves the log currently in RAM onto disk. The disk it is saved on depends on the value of `loadpath=` in the Environment Menu.

NOTE *The Log is always saved in a file called* `logfile` *in the directory indicated by the loadpath variable on the test system's* **local** *disk. You cannot save a logfile on a remote disk. You must create the file, with at least 32K bytes of data in it,* **before** *you run the Exec.*

`logfile`

Running the logfile command starts or stops the logging process. If the Exec is already logging messages, that procedure stops, and any new messages are discarded. If the Exec is not logging messages when this command is entered, it loads the current logfile into the RAM log, then starts logging all new messages to both the ram and the disk logfile.

Note that this command has the same effect as the `logfile=` variable in the Environment Menu. It is included here for convenience.

3

Sun-2 Color Board Diagnostic

3

Sun-2 Color Board Diagnostic

3.1. General Description

This diagnostic program, which runs under the Exec, tests the Sun-2 Color Graphics Board. This diagnostic runs on both Sun-2 and Sun-3 systems that are using a Sun-2 Color Graphics Board.

3.2. Required Equipment

There are two color boards available from Sun. The older version, introduced with the Sun-2 line, runs on both Sun-2's and Sun-3's. It has 4 BNC connectors on the back of the board. Use this diagnostic to test it. Use the Sun-3 Color Board Diagnostic to test the new color board. It has 5 BNC connectors on the back.

Because the diagnostic displays patterns on the color monitor, we advise you to run the diagnostic from a terminal connected to SIO-A.

NOTE *Starting the system "cold" may effect DAC conversion. If you just powered-on a cold system and you encounter DAC problems, allow the system to "warm up" for about 20 minutes, then rerun any tests that failed.*

3.3. User Interface

As the diagnostic exercises the color circuits of the Sun color board, it generates colors and images on the color monitor. These displays provide important information about the condition of the color board. You must check the colors and patterns for correctness and even distribution of color.

Tests that run continuously can be stopped by typing the letter q on the keyboard. For some tests, there may be a delay before it actually exits.

3.4. The Main Menu

The diagnostic is divided into two parts, auto and manual tests. The Auto test provides good coverage of every part of the hardware. The manual test contains most of the subtests run by the auto test, as well as scope loops and debugging tools for isolating hardware problems.

```
Sun-2 Color Board Diagnostic Program Rev: X XX/XX/XX Main Menu

  Auto test         Test all hardware
  Manual test       Select any part of hardware

Command =>
```

Auto

If you type **a**, the *auto test* is invoked. When the test starts, it prompts for the number of times to run the test. Enter the count (in decimal), then press Return to start the test. The Auto test includes DAC, interrupt, TTL / ECL color map, frame buffer, zoom and pan, ROPC, pixel plane mask, and word plane mask tests. Only the DAC test requires user interaction. You must press Return to advance to the next DAC test. All other tests are executed automatically. At the end of a test, a cumulative error message is displayed on the screen. Press Return to get back to main menu.

Manual

The *manual test* allows you to test each part of the hardware. If the Auto test discovers an error, you should re-run the specific test from the Manual Test menu. The Manual Test mode also provides scope loops and some useful debugging tools, chosen from the Control Register Menu. From the Main Menu, type *m* to run Manual Tests.

3.5. Manual Test Menu

You may select any item on the menu or press Esc to back up to the main menu.

```
Sun-2 Color Board Diagnostic Program Rev: X XX/XX/XX Manual Test Menu

  Cntl              Test control registers
  Int               Test interrupts
  COlor             Test color maps
  Buffer            Test frame buffer
  Ropc              Test ROPC units
  Zoom              Test zoom and pan
  DAc               Test DACs and monitor
  Monitor           Brief monitor test
  Auto              Perform auto test once
                    .
Command =>
```

Cntl
Type **c** to select the *control register test.* The diagnostic will display the Control Register menu.

Int
Type **i** to select the *interrupt test.* The diagnostic will display the Interrupt Test menu.

COlor
Type **co** to select the *color map test.* The diagnostic will display the Color Map Test menu.

Buffer
Type **b** select the *frame buffer test.* The diagnostic will display the Frame Buffer Test menu.

Ropc
Type **r** to select the *ROPC test.* The diagnostic will display the ROPC Test menu.

Zoom
Type **z** to select the *zoom and pan test.* The diagnostic displays the Zoom and Pan Test menu.

DAc
Type **da** menu to select the *DAC test.* The diagnostic will display the DAC Test menu.

Monitor
Type **m** to select the *brief monitor test.* This test displays a series of visual tests to check the monitor. There is no sub-menu for this test.

Auto
Type **a** to select the *auto test.* It tests all parts of the hardware automatically, once. There is no sub-menu for this test.

3.6. Control Register Menu

Each of the choices below allows you to exercise a different register. You can only choose one register at a time.

After selecting a register to test, a long menu is displayed with different read, write, increment (or decrement) and compare options. These options allow you to exercise the register in question in various ways while examining the circuits with an oscilloscope.

```
Sun-2 Color Board Diagnostic Program Rev: X  XX/XX/XX Register Test Menu

  Status          Status register
  Perplane        Per_Plane register
  Word            Word Pan register
  PIxel           Pixel Pan register
  Line            Line Offset and Zoom register
  Variable        Variable Zoom register
  Interrupt       Interrupt register

Command =>
```

Status
The *Status Register* is sixteen bits wide. It contains all the status information for the color board. It is cleared when a bus reset is issued.

Perplane
The *Per_Plane Register* is eight bits wide. Its used to restrict access to selected bit planes.

Word
The *Word Pan Register* sets the origin of the region being displayed. This register is only used during a word pan.

PIxel
The *Pixel Pan Register* sets the origin of the region being displayed. This register is only used during a pixel pan. If the panning is horizontal, a four-bit field called the pixel_offset can be used to limit panning to between one and four pixels at a time.

Line
The *Line Offset and Zoom Register* is an eight bit register. The least significant four bits hold a value that specifies the display size of a single frame buffer pixel. As the zoom level increases, each pixel in the frame buffer will be displayed as a larger and larger region on the screen. If vertical panning is being done, the most significant four-bit field, called the line_offset, can be used to limit panning to between one and four lines at a time.

Variable
The *Variable Zoom Register* specifies the line number that is the lower limit of the zoom region.

Interrupt

The *Interrupt Register* contains an eight bit interrupt vector. The color board sends it to the VME bus during a vectored interrupt.

Register Test Sub-Menu

After selecting the register you want to test, the following sub-menu gives you various ways to test it.

```
Sun-2 Color Board Diagnostic Program Rev: X  XX/XX/XX Register Test Sub Menu

  Read              Read once
  RC                Read continuously
  Write             Write once
  WC                Write continuously
  WRRd              Write/read once
  WRRC              Write/read continuously
  Inc               Wrt/rd/cmp and dec data by 3
  IC                Wrt/rd and inc data continuously
  Forever           Wrt/rd & stop/rd forever on error
  INN               Wrt/rd/cmp & inc data by n
  Alt               Wrt/rd alternating data

Command =>
```

Read

The *Read Once* command reads the indicated register once, and prints its hexadecimal contents on the screen. The message printed is

`Register` *name*`. Read:` *value*`.`

RC

The *Read Continuously* command reads the indicated register continuously until it is interrupted. The message printed is

`Register` *name*`. First Read:` *value*

which is the value of the register the first time the diagnostic reads it. The test displays nothing else if this value is consistent. At the end of 0x10000 reads the test does two things. First, if the value read differs, the test prints an `E` on the screen. Next, the diagnostic checks the keyboard to see if a `q` has been typed. If it has, the test quits. If it hasn't, the test starts the cycle over.

Write

The *Write Once* command prompts for a value to write, by printing

`Enter Datum(hex)`

It then writes the indicated register once, using the value of the argument given. The message printed is

`Register` *name*`. Wrote:` *value*

WC

The *Write Continuously* command prompts for a value to write, by printing `Enter Datum(hex)`. It then writes the indicated register repeatedly. At the end of 0x10000 writes, the test prints the following message:

`Register` *name*`. First Write:` *value*`.`

then checks to see if q has been entered. If it has, the test exits. If it wasn't entered, the test starts the cycle over.

WRRd

The *Write then Read once* command prompts for a value to write, with `Enter Datum(hex)`. It then writes the indicated register once, using the value of the argument given. It then immediately reads the value of the register back, and prints it on the screen. The message printed is

`Register` *name* `Wrote:` *value*`. Read:` *value*`.`

WRRC

The *Write, then Read continuously* command prompts for a value to write, by printing:

```
Enter Datum(hex)
```

It then writes to the indicated register, using the value of the argument given. It immediately reads the value of the register back, then starts over again. Every 0x10000 iterations, the test does three things. First, the test prints the following message:

`Register` *name* `First Write:` *value*`. First Read:` *value*`.`

which is the value of the register the first time the diagnostic reads it. Next, if the value read differs (indicating an error), the test prints an `E` on the screen. Finally, the diagnostic checks to see if a `q` was typed. If it was, the diagnostic exits. If not, the cycle repeats.

Inc

The *Write, Read, Compare, and Decrement* command writes to the indicated register, starting with the value of the `0XFFFF`. It then immediately reads the value of the register back, then compares it against the original value. The command then decrements the value by 3 and starts over again. If the value read back differs from the value that was written, the register is read twice more, and a message showing the discrepancies is printed:

`Device` *device*`. Register` *name*`. Wrote` *value*`. Read` *rd1, rd2, rd3*

When the value gets decremented below zero, the test ends.

IC

The *Write, Read, and Increment* command writes to the indicated register, starting with the value `0X0`. It immediately reads the value of the register back, then increments the value by one and starts over again. The cycle repeats continuously until
a `q` is typed. The command does not print the register's contents.

Forever

The *Write, Read, and Read on Error* command writes to the indicated register, starting with the value `0XFFFF`. It immediately reads the value of the register back, then compares it against the original value. The value is decremented by one, then the cycle repeats. The command ends when the value is decremented to zero. If the data doesn't match, the command prints the message:

```
Device device. Register name. Wrote value. Read value.
```

followed by

```
Hit any Character to Continue (r to read forever)
```

If any key but `r` is pressed, the test ignores the error, and continues on to the next data value.

If the `r` key is pressed, the command starts reading the register continuously. Every 0x10000 reads, the test displays the values with the message:

```
Register name. Read value
```

and reads the keyboard. If the `q` has been pressed, the test quits. Otherwise the test repeats the cycle, reading another 0x10000 times.

INN

The *Write, Read, Compare and Increment by n* command prompts for an increment value to write, by printing

```
Enter increment(hex):
```

It then writes to the indicated register, starting with the value `0X0`. It immediately reads the value of the register back, then increments the value by the amount given and starts over again. The cycle repeats until the value exceeds `0X10000` If the value read does not match the value written, the test reports it with this message:

```
Device device. Register name. Wrote value. Read value.
```

Alt

The *Write and Read alternating data* command does a write and read cycle using two different data values. The values are written alternatively on each cycle. The test prompts for the values by printing:

```
Enter first Datum(hex):
```

followed by

```
Enter second Datum(hex):
```

and finally,

```
Print Error Messages (y/n)?
```

If error messages are enabled, the test will loop for 0x10000 iterations, then print the following message when the value read and the value written don't match:

```
Device device. Register name. Wrote value. Read value.
```

It then reads the keyboard to see if the letter q has been typed. If it has, the test quits. If it hasn't, the test repeats the cycle.

3.7. Interrupt Test Menu

Type **i** from the Manual Test Menu to select the interrupt test menu.

```
Sun-2 Color Board Diagnostic Program Rev: X XX/XX/XX Interrupt Test Menu

  Auto              Perform Auto Test once

Command =>
```

Auto

The *Auto Interrupt Test* selection asks the Exec to install an interrupt handler. After the test it asks the Exec to remove that handler. If installing or removing the interrupt handler fails, the diagnostic aborts and returns to the Exec top menu.

3.8. Color Map Test Menu

This test checks the Sun-2 color map by loading it with different values and patterns, then verifying that the values in the map are correct. Select the color map values by choosing them yourself, or have the values selected automatically. These tests can be set to run one time or continuously.

To display the image in the Sun-2 color frame buffer memory, each 8-bit pixel is used as an index into a 256-element color look-up table. Each element of the table is 24 bits; 8 bits for the red component, 8 bits for the green, and 8 for the blue. The color look-up tables consist of a high speed ECL look-up table that controls the color monitor, and a TTL shadow color lookup table that is loaded and read by the software. The TTL shadow color lookup table is loaded into the ECL lookup table to make the new colors visible.

While running various tests, you should see the appropriate image displayed on the color monitor screen. Note that the

```
Load TTL -> ECL cmap
```

command must be executed before values loaded into the color map become visible.

```
Sun-2 Color Board Diagnostic Program Rev: X  XX/XX/XX Color Map Test Menu

  Acqttl          Acquire access to TTL cmap
  Relttl          Relinquish access to TTL cmap
  Loadttlecl      Load TTL -> ECL cmap once
  Setrgb          Set 0-255 red, 256-511 grn, 512-767 blue
  SIngle          Test single location
  AUto            Auto test
  ATC             Continuous auto test

Command =>
```

Following are brief descriptions of each Color Map Test Menu command.

Acqttl
The *TTL Acquire* command acquires access to the TTL color map.

Relttl
The *TTL Relinquish* command relinquishes access to the TTL color map.

Loadttlecl
The *TTL to ECL once* command transfers the data from the TTL color map to the ECL color map once.

Setrgb
The *set color map* command loads the TTL color map with three linear ramps of Red, Green, and Blue.

SIngle
The *test single location* command checks a selected TTL color map entry. You are prompted for an offset from the beginning of the color map, then that location is read. The legal values for an offset range from 0x00 to 0xFF.

AUto
The *auto test* command performs the auto color map test once.

ATC
The *continuous auto test* command performs the auto color map test continuously until the letter q is typed on the keyboard.

3.9. Frame Buffer Test Menu

The Frame Buffer Memory Tests allow reading and writing to or from the frame buffer. They are useful for checking frame buffer DRAM in word or pixel mode, and for testing the frame buffer data and address lines. These tests are controlled by a series of command menus.

Some tests enable you to write selected patterns or constant values into memory and then read them back and verify them. There are also a set of automatic frame buffer test routines that can be run once or continuously.

```
Sun-2 Color Board Diagnostic Program Rev: X  XX/XX/XX FB Test Menu

  Checker          Write checkerboard
  Vertical         Write a vertical line
  Horizontal       Write a horizontal line
  VRfyv            Verify a vertical line
  VRYh             Verify a horizontal line
  COnstant         Fill region with constant
  Printv           Print all vertical lines
  PTh              Print all horizontal lines
  Auto             Auto test
  ATc              Continuous auto test
  Word             Fill frame buffer in word mode
  Oneram           Fill one ram
  HRztalw          Write horizontal line in word mode
  Evenv            Write even vertical lines in fbuf
  ODdv             Write odd vertical lines in fbuf
  SCanw            Scan word mode memory for a value
  Filladdr         Fill frame buffer with addresses

Command =>
```

Checker
The *checkerboard* command draws a 16 x 16 grid on the frame buffer. Each box has 72 x 56 pixels. The boxes are numbered, starting with zero, increasing from left to right, then top to bottom.

Vertical
The *vertical line* command prompts for a color and a column number, then draws the corresponding vertical line.

Horizontal
The *horizontal line* command prompts for a color and a row number, then draws the corresponding horizontal line.

VRfyv
The *verify vertical line* command verifies that the line displayed by the `vertical line` command was drawn correctly. The test prompts for a column number and a color, then checks to see if the line exists in the frame buffer. You must run the `vertical line` command with the appropriate parameters before running this test.

If the test finds an error, it prints the following message:

`Error. X= xvalue. Y = value. Rd color`

VRYh

The *verify horizontal line* command verifies that the line displayed by the `horizontal line` command was drawn correctly. The test prompts for a row number and a color, then checks to see if the line exists in the frame buffer. You must run the `horizontal line` command with the appropriate parameters before running this test. If the test finds an error, it prints the following message:

`Error. X= xvalue. Y = yvalue. Rd color`

COnstant

The *fill region* command draws a rectangular region of constant color on the screen. The command prompts for the x and y coordinates of the rectangle's upper left corner, along with its height, width and color.

Printv

The *all vertical lines* command draws every possible vertical line on the screen. The color of each line is computed by performing a logical `and` of the line's column number and the value `0xFF`.

PTh

The *all horizontal lines* command draws every possible horizontal line on the screen. The color of each line is computed by performing a logical `and` of the line's row number and the value `0xFF`.

Auto

The *auto test* prompts for which addressing mode to use (pixel or word), then performs a series of automatic tests. If double buffering RAM is installed on the system, the test will prompt you to select the proper set to test: A or B. When the test completes, it prints the total number of errors it found.

ATc

The *continuous auto test* runs a series of automatic tests continuously, using both pixel and word addressing modes. If double buffering RAM is installed on the system, both sets are tested. After each cycle, the test prints the total number of errors it found. Type the letter `q` on the keyboard to exit this test.

Word

The *word mode* command prompts for a 32-bit value. It then writes this value into every location in the frame buffer. The frame buffer is accessed in word mode for this test.

Oneram

The *one ram* command starts by prompting for a bit plane and ram column number. Then you choose one of three patterns: all `0X00`'s all `0X01`'s or alternating `0X00`'s and `0X01`'s. The test then fills the area with the pattern selected.

HRztalw
The *horizontal word mode* command starts by prompting for the proper bit plane, row, and data to be written. The test then fills the proper row with the data value.

Evenv
The *even vertical lines* command prompts for a color value. It then fills every even numbered column with that color.

ODdv
The *odd vertical lines* command prompts for a color value. It then fills every odd numbered column with that color.

SCanw
The *scan word* command prompts for a data value. It then scans the buffer, in word mode, looking for that value. Every time it encounters the value, it prints:

```
Data matches value at address location
```

Filladdr
The *fill with addresses* command writes to the frame buffer in word mode, filling each location with its address value. The test starts at location 0, bit plane 0.

3.10. ROPC Test Menu

Type **r** at the manual test menu to select the ROPC test menu.

```
Sun-2 Color Board Diagnostic Program Rev: X XX/XX/XX DAC Test Menu

  Rhramp          Print Horizontal Red Ramp
  Ghramp          Print Horizontal Grn Ramp
  Bhramp          Print Horizontal Blu Ramp
  Whramp          Print Horizontal White Ramp
  RVramp          Print Vertical Red Ramp
  GVramp          Print Vertical Grn Ramp
  BVramp          Print Vertical Blu Ramp
  WVramp          Print Vertical White Ramp
  RGBw            Print Simultaneous RGBW horizontal Ramps
  BORder          Print Screen Borders x=(0:1152) y=(0:899)
  Alter           Write alternating bars of color to test glitches
  AUto            Continuous auto tests
  Stab            Test Screen Stability

Command =>
```

Rhramp
The *Horizontal Red Ramp test* draws a shaded black-red-black image horizontally across the screen. This test checks the linearity of the red DAC.

Ghramp
The *Horizontal Green Ramp test* draws a shaded black-green-black image horizontally across the screen. This test checks the linearity of the green DAC.

Bhramp
The *Horizontal Blue Ramp test* draws a shaded black-blue-black image horizontally across the screen. This test checks the linearity of the blue DAC.

Whramp
The *Horizontal White Ramp test* draws a shaded black-white-black image horizontally across the screen. This test checks the linearity of all three DACs working together.

RVramp
The *Vertical Red Ramp test* draws a shaded black-red-black image vertically down the screen. This test checks the linearity of the red DAC.

GVramp
The *Vertical Green Ramp test* draws a shaded black-green-black image vertically down the screen. This test checks the linearity of the green DAC.

BVramp
The *Vertical Blue Ramp test* draws a shaded black-blue-black image vertically down the screen. This test checks the linearity of the blue DAC.

WVramp
The *Vertical White Ramp test* draws a shaded black-white-black image

vertically down the screen. This test checks the linearity of all three DACs working together.

RGBw
The *RGBW horizontal Ramp test* draws red, green, blue, and white horizontal ramp images in sequence on the screen.

BORder
The *Screen Border test* draws a dark screen with a white line around the borders. This checks the beam deflection circuitry.

Alter
The *Color Bar test* writes alternating bars of color across the screen.

AUto
The *Continuous auto tests* command runs through the tests listed above until a 'q' is typed on the keyboard.

Stab
The *Screen Stability test* Displays a gray pattern on the entire screen. Check the display for consistent shading and color.

3.11. Error messages

If the diagnostic encounters a hardware failure, it prints the error message on both the screen and to the Exec's error log file. At the beginning of each error message a test number is displayed to indicate the test which failed. The following table describes each test number.

Table 3-1 *Color2 Error Message Table*

Error Messages	
Number	*Test*
0	Status Reg
1	Plane Mask Reg
2	Word Pan Reg
3	LOff & Zoom Reg
4	Pixel Pan Reg
5	Variable Zoom Reg
6	Interrupts
7	Shadow Color Map
8	FB Word-Memory
9	FB Pixel-Memory
10	ROPC Plane 0
11	ROPC Plane 1
12	ROPC Plane 2
13	ROPC Plane 3
14	ROPC Plane 4
15	ROPC Plane 5
16	ROPC Plane 6
17	ROPC Plane 7
18	Pix-mode Plane Masking
19	Word-mode Plane Masking
20	ROPC Pixel Memory
21	FB Word Memory Plane 0
22	FB Word Memory Plane 1
23	FB Word Memory Plane 2
24	FB Word Memory Plane 3
25	FB Word Memory Plane 4
26	FB Word Memory Plane 5
27	FB Word Memory Plane 6
28	FB Word Memory Plane 7
29	Interrupt Vector Reg

The address of the bad hardware, shown in an error message, is relative. It is an offset from the starting address of the Color Board. For example, the first address of word mode frame buffer is 0x0, and the first address of pixel mode frame buffer is 0x100000.

3.12. Glossary

DAC
Digital to Analog Converter.

DMA
Direct Memory Access.

Exec
Diagnostic Executive, a multi-tasking environment under which these diagnostics run.

FB Frame Buffer

ROPC
RasterOP Chip.

4

Sun-3 Color Board Diagnostic

4

Sun-3 Color Board Diagnostic

4.1. General Description

This diagnostic program, which runs under the Exec, tests the Sun-3 Color Graphics Board. This diagnostic runs on both Sun-2 and Sun-3 systems that are using the Sun-3 Color Graphics Board.

4.2. Required Equipment

There are two color boards available from Sun. The older version, introduced with the Sun-2 line, runs on both Sun-2's and Sun-3's. It has 4 BNC connectors on the back of the board. Use the Sun-2 Color Board Diagnostic to test it. Use this diagnostic to test the new color board. It has 5 BNC connectors on the back.

Because the diagnostic displays patterns on the color monitor, we advise you to run the diagnostic from a terminal connected to SIO-A.

NOTE *Starting the system "cold" may effect DAC conversion. If you just powered-on a cold system and you encounter DAC problems, allow the system to "warm up" for about 20 minutes, then rerun any tests that failed.*

4.3. User Interface

As the diagnostic exercises the color circuits of the Sun color board, it generates colors and images on the color monitor. These displays provide important information about the condition of the color board. You must check the colors and patterns for correctness and even distribution of color.

Tests that run continuously can be stopped by typing the letter `q` on the keyboard. For some tests, there may be a delay before it actually exits.

4.4. The Main Menu

The diagnostic is divided into two parts, auto and manual tests. The Auto test provides good coverage to every part of the hardware. The manual test contains most of the subtests run by the auto test, as well as scope loops and debugging tools for isolating hardware problems.

```
Sun-3 Color Board Diag (SINGLE/DOUBLE BUF) Rev: X  XX/XX/XX Main Menu

  Auto test          Test all hardware
  Manual test        Select any part of hardware

Command =>
```

The title at the top of the menu will show either `SINGLE` or `DOUBLE BUF`, depending on whether the board is single or double buffered.

`Auto`

If you type **a**, the *auto test* is invoked. When the test starts, it prompts for the number of times to run the test. Enter the count (in decimal), then press Return to start the test. The Auto test includes DAC, interrupt, TTL/ECL color map, frame buffer (including double buffering if the hardware exists), DMA window (if the hardware exists), ROPC, pixel plane mask, and word plane mask tests. Only the DAC test requires some user interaction. You must press [Return] to advance to the next DAC test. All other tests are executed automatically. At the end of test, a cumulative error message is displayed on the screen. Press [Return] to get back to main menu.

`Manual`

The *manual test* allows you to test each part of the hardware. If the Auto test discovers an error, you should re-run the specific test in Manual Test menu. The Manual test also has scope loops and some useful debugging tools in it. From the Main Menu, type **m** to run the Manual Test.

4.5. Manual Test Menu

You may select any item on the menu or press Esc to back up to the main menu.

```
Sun-3 Color Board Diag (SINGLE/DOUBLE BUF) Rev:X XX/XX/XX Manual Test Menu

  Cntl          Test control registers
  Int           Test interrupts
  COlor         Test color maps
  Buffer        Test frame buffer
  Ropc          Test ROPC units
  Dma           Test DMA
  DAc           Test DACs and monitor
  Monitor       Brief monitor test
  Auto          Perform auto test once

Command =>
```

Cntl
Type **c** to select the *control register test.* The diagnostic will display the Control Register menu.

Int
Type **i** to select the *interrupt test.* The diagnostic will display the Interrupt Test menu.

COlor
Type **co** to select the *color map test.* The diagnostic will display the Color Map Test menu.

Buffer
Type **b** select the *frame buffer test.* The diagnostic will display the Frame Buffer Test menu.

Ropc
Type **r** to select the *ROPC test.* The diagnostic will display the ROPC Test menu.

Dma
Type **d** to select the *DMA window test.* Both pattern write/read and visual tests are executed in this test. Note that there's no sub-menu for this test.

DAc
Type **da** to select the *DAC test.* The diagnostic will display the DAC Test menu.

Monitor
Type **m** to select the *brief monitor test.* This test displays a series of visual tests to check the monitor. There is no sub-menu for this test.

Auto
Type **a** to select the *auto test.* It will test all parts of the hardware automatically once. There is no sub-menu for this test.

4.6. Control Register Menu

Each of the choices below allows you to exercise a different register. You can only choose one register at a time.

After selecting a register to test, a long menu is displayed with different read, write, increment (or decrement) and compare options. These options allow you to exercise the register in question in various ways while examining the circuits with an oscilloscope.

```
Sun-3 Color Board Diag (SINGLE/DOUBLE BUF)Rev: X XX/XX/87 Register Test Menu

  Status          Status register
  Perplane        Per_Plane register
  Frame           Read-only Frame Count register
  Base            Write-only Dma Base register
  Width           Write-only Dma Width register
  Double          Double-buffering register
  Interrupt       Interrupt register

Command =>
```

Status
The *Status Register* is sixteen bits wide. It contains all the status information for the color board. It is cleared when a bus reset is issued.

Perplane
The *Per_Plane Register* is eight bits wide. Its used to restrict access to selected bit planes.

Frame
The *Frame Register* is a counter that is incremented by the display hardware during every vertical retrace.

Base
The *Base Register* points to the beginning of the DMA window.

Width
The *Width Register* contains a value which is 1/16th the width of the DMA window.

Double
The *Double Register* is sixteen bits wide. Its cleared when a bus reset is issued.

Interrupt
The *Interrupt Register* contains an eight bit interrupt vector. The color board sends it to the VME bus during a vectored interrupt.

Register Test Sub-Menu

After selecting the register you want to test, the following sub-menu gives you various ways to test it.

```
Sun-3 Color Board Diag (SINGLE/DOUBLE BUF)Rev: X  XX/XX/87 Register Test Sub Menu

  Read              Read once
  RC                Read continuously
  Write             Write once
  WC                Write continuously
  WRRd              Write/read once
  WRRC              Write/read continuously
  Inc               Wrt/rd/cmp and dec data by 3
  IC                Wrt/rd and inc data continuously
  Forever           Wrt/rd & stop/rd forever on error
  INN               Wrt/rd/cmp & inc data by n
  Alt               Wrt/rd alternating data

Command =>
```

Read

The *Read Once* command reads the indicated register once, and prints its hexadecimal contents on the screen. The message printed is

`Register` *name*`. Read:` *value*`.`

RC

The *Read Continuously* command reads indicated register repeatedly. It prints the following message at the beginning of the cycle:

`Register` *name*`. First Read:` *value*

which is the value of the register the first time the diagnostic reads it. The test displays nothing else if this value is consistent. At the end of 0x10000 reads the test does two things. First, if the value read differs, the test prints an `E` on the screen. Next, the diagnostic checks the keyboard to see if a `q` has been typed. If it has, the test quits. If it hasn't, the test starts the cycle over.

Write

The *Write Once* command prompts for a value to write, by printing

`Enter Datum(hex)`

It then writes the indicated register once, using the value of the argument given. The message printed is

`Register` *name*`. Wrote:` *value*

WC

The *Write Continuously* command prompts for a value to write, by printing `Enter Datum(hex)`. It then writes the indicated register repeatedly. At the end of 0x10000 writes, the test prints the following message:

```
Register name. First Write: value.
```

then checks to see if q has been entered. If it has, the test exits. If it wasn't entered, the test starts the cycle over.

WRRd

The *Write then Read once* command prompts for a value to write, by printing `Enter Datum(hex)`. It then writes the indicated register once, using the value of the argument given. It then immediately reads the value of the register back, and prints it on the screen. The message printed is

```
Register name Wrote: value. Read: value.
```

WRRC

The *Write, then Read continuously* command prompts for a value to write, by printing:

```
Enter Datum(hex)
```

It then writes to the indicated register, using the value of the argument given. It immediately reads the value of the register back, then starts over again. Every 0x10000 iterations, the test does three things. First, the test prints the following message:

```
Register name First Write: value. First Read: value.
```

which is the value of the register the first time the diagnostic reads it. Next, if the value read differs (indicating an error), the test prints an E on the screen. Finally, the diagnostic checks to see if a q was typed. If it was, the diagnostic exits. If not, the cycle repeats.

Inc

The *Write, Read, Compare, and Decrement* command writes to the indicated register, starting with the value of the `0XFFFF`. It then immediately reads the value of the register back, then compares it against the original value. The command then decrements the value by 3 and starts over again. If the value read back differs from the value that was written, the register is read twice more, and a message showing the discrepancies is printed:

```
Device device. Register name. Wrote value. Read rd1, rd2, rd3
```

When the value gets decremented below zero, the test ends.

IC

The *Write, Read, and Increment* command writes to the indicated register, starting with the value `0X0`. It immediately reads the value of the register back, then increments the value by one and starts over again. The cycle repeats continuously until a q is typed. The command does not print the register's contents.

Forever

The *Write, Read, and Read on Error* command writes to the indicated register, starting with the value `0XFFFF`. It immediately reads the value of the register back, then compares it against the original value. The value is decremented by one, then the cycle repeats. The command ends when the

value is decremented to zero. If the data doesn't match, the command prints the message:

```
Device device. Register name. Wrote value. Read value.
```

followed by

```
Hit any Character to Continue (r to read forever)
```

If any key but `r` is pressed, the test ignores the error, and continues on to the next data value.

If the `r` key is pressed, the command starts reading the register continuously. Every 0x10000 reads, the test displays the values with the message:

```
Register name. Read value
```

and reads the keyboard. If the `q` has been pressed, the test quits. Otherwise the test repeats the cycle, reading another 0x10000 times.

INN

The *Write, Read, Compare and Increment by n* command prompts for an increment value to write, by printing

```
Enter increment(hex):
```

It then writes to the indicated register, starting with the value `0X0`. It immediately reads the value of the register back, then increments the value by the amount given and starts over again. The cycle repeats continuously until the command is interrupted. If the value read does not match the value written, the test reports it with this message:

```
Device device. Register name. Wrote value. Read value.
```

Alt

The *Write and Read alternating data* command does a write and read cycle using two different data values. The values are written alternatively on each cycle. The test prompts for the values by printing:

```
Enter first Datum(hex):
```

followed by

```
Enter second Datum(hex):
```

and finally,

```
Print Error Messages (y/n)?
```

If error messages are enabled, the test will loop for 0x10000 iterations, then print the following message when the value read and the value written don't match:

```
Device device. Register name. Wrote value. Read value.
```

It then reads the keyboard to see if the letter `q` has been typed. If it has, the test quits. If it hasn't, the test repeats the cycle.

4.7. Interrupt Test Menu

Type *i* from the Manual Test menu to select the interrupt test menu.

```
Sun-3 Color Board Diag (SINGLE/DOUBLE BUF) Rev:X XX/XX/87 Interrupt Test Menu

  Auto              Perform Auto Test once

Command =>
```

Auto

The *Auto Interrupt Test* tests the interrupt bit in the Status Register and the wait bit in the Double Buffer Register. Interrupts are enabled, then verified. Interrupts are disabled again before the test exits.

4.8. Color Map Test Menu

This test checks the Sun-3 color map by loading it with different values and patterns, then verifying that the values in the map are correct. Select the color map values by choosing them yourself, or have the values selected automatically. These tests can be set to run one time, or continuously.

To display the image in the Sun-3 color frame buffer memory, each 8-bit pixel is used as an index into a 256-element color lookup table. Each element of the table is 24 bits; 8 bits for the red component, 8 bits for the green, and 8 for the blue. The color look-up tables consist of a high speed ECL look-up table that controls the color monitor, and a TTL shadow look-up table that is loaded and read by the software. The TTL shadow color look-up table is loaded into ECL lookup table to make the new colors visible.

While running various tests, you should see the appropriate image displayed on the color monitor screen. Note that the

```
Load TTL -> ECL cmap
```

command must be executed before values loaded into the color map become visible.

```
Sun-3 Color Board Diag (SINGLE/DOUBLE BUF) Rev:X XX/XX/87 Color Map Test Menu

  Acqttl          Acquire access to TTL cmap
  Relttl          Relinquish access to TTL cmap
  Ttl             TTL-to-ECL cmap transfers
  Ecl             ECL-to-TTL cmap transfers
  Loadttlecl      Load TTL -> ECL cmap once
  LDecIttl        Load ECL -> TTL cmap once
  Setrgb          Set 0-255 red, 256-511 grn, 512-767 blue
  SIngle          Test single location
  AUto            Auto test
  ATC             Continuous auto test

Command =>
```

Acqttl
The *TTL Acquire* command acquires access to the TTL color map.

Relttl
The *TTL Relinquish* command relinquishes access to the TTL color map.

Ttl
The *TTL to ECL* command enables data transfer from the TTL to the ECL color map.

Ecl
The *ECL to TTL* command enables data transfer from the ECL to the TTL color map.

Loadttlecl
The *TTL to ECL once* command transfers the data from the TTL color map to the ECL color map once.

LDecIttl
The *ECL to TTL once* command transfers the data from the ECL color map to the TTL color map once.

Setrgb
The *set color map* command loads the TTL color map with three linear ramps of Red, Green, and Blue.

SIngle
The *test single location* command checks a selected TTL color map entry. You are prompted for an offset from the beginning of the color map, then that location is read. The legal values for an offset range from `0x00` to `0xFF`.

AUto
The *auto test* command performs the auto color map test once.

ATC
The *continuous auto test* command performs the auto color map test until the letter `q` is typed on the keyboard.

4.9. Frame Buffer Test Menu

The Frame Buffer Memory Tests allow reading and writing to or from the frame buffer. They are useful for checking frame buffer DRAM in word or pixel mode, and for testing the frame buffer data and address lines. These tests are controlled by a series of command menus.

The tests enable you to write selected patterns or constant values into memory and then read them back and verify them. There are also a set of automatic frame buffer test routines that can be run once or continuously.

```
Sun-3 Color Board Diag (SINGLE/DOUBLE BUF)Rev:X XX/XX/87 FB Test Menu

  Checker          Write checkerboard
  Vertical         Write a vertical line
  Horizontal       Write a horizontal line
  VRfyv            Verify a vertical line
  VRYh             Verify a horizontal line
  COnstant         Fill region with constant
  Printv           Print all vertical lines
  PTh              Print all horizontal lines
  Auto             Auto test
  ATc              Continuous auto test
  Word             Fill frame buffer in word mode
  Oneram           Fill one ram
  HRztalw          Write horizontal line in word mode
  Evenv            Write even vertical lines in fbuf
  ODdv             Write odd vertical lines in fbuf
  SCanw            Scan word mode memory for a value
  Filladdr         Fill frame buffer with addresses

Command =>
```

Checker

The *checkerboard* command draws a 16 X 16 grid on the frame buffer. Each box is 72 x 56 pixels. The boxes are numbered, starting with zero, increasing from left to right, then top to bottom.

Vertical

The *vertical line* command prompts for a color and a column number, then draws the corresponding vertical line.

Horizontal

The *horizontal line* command prompts for a color and a row number, then draws the corresponding horizontal line.

VRfyv

The *verify vertical line* command verifies that the line displayed by the `vertical line` command was drawn correctly. The test prompts for a column number and a color, then checks to see if the line exists in the frame buffer. You must run the `vertical line` command with the appropriate parameters before running this test.

If the test finds an error, it prints the following message:

`Error. X= xvalue. Y = yvalue. Rd color`

VRYh

The *verify horizontal line* command verifies that the line displayed by the `horizontal line` command was drawn correctly. The test prompts for a row number and a color, then checks to see if the line exists in the frame buffer. You must run the `horizontal line` command with the appropriate parameters before running this test. If the test finds an error, it prints the following message:

`Error. X= xvalue. Y = yvalue. Rd color`

COnstant

The *fill region* command draws a rectangular region of constant color on the screen. The command prompts for the x and y coordinates of the rectangle's upper left corner, along with its height, width and color.

Printv

The *all vertical lines* command draws every possible vertical line on the screen. The color of each line is computed by performing a logical `and` of the line's column number and the value `0xFF`.

PTh

The *all horizontal lines* command draws every possible horizontal line on the screen. The color of each line is computed by performing a logical `and` of the line's row number and the value `0xFF`.

Auto

The *auto test* prompts for the addressing mode to use (pixel or word), then performs a series of automatic tests. If double buffering RAM is installed on the system, the test will prompt for the proper set to test: A or B. When the test completes, it prints the total number of errors it found.

ATc

The *continuous auto test* runs a series of automatic tests continuously, using both pixel and word addressing modes. If double buffering RAM is installed on the system, both sets are tested. After each cycle, the test prints the total number of errors it found. Type the letter `q` on the keyboard to exit this test.

Word

The *word mode* command prompts for a 32 bit value. It then writes this value into every location in the frame buffer. The frame buffer is accessed in word mode for this test.

Oneram

The *one ram* command starts by prompting for a bit plane and ram column number. Then you choose one of three patterns: all `0X00`'s all `0X01`'s or alternating `0X00`'s and `0X01`'s. The test then fills the area with the pattern selected.

HRztalw
The *horizontal word mode* command starts by prompting for the proper bit plane, row, and data to be written. The test then fills the proper row with the data value.

Evenv
The *even vertical lines* prompts for a color value. It then fills every even numbered column with that color.

ODdv
The *odd vertical lines* prompts for a color value. It then fills every odd numbered column with that color.

SCanw
The *scan word* prompts for a data value. It then scans the buffer, in word mode, looking for that value. Every time it encounters the value, it prints:

```
Data matches value at address location
```

Filladdr
The *fill with addresses* command writes to the frame buffer in word mode, filling each location with its address value. The test starts at location 0, bit plane 0.

4.10. ROPC Test Menu

Type **r** from the Manual Test Menu to select ROPC test. Then the following menu will be displayed.

```
Sun-3 Color Board Diag (SINGLE/DOUBLE BUF) Program Rev:X XX/XX/87 ROPC Test Menu

  Auto              Auto tests

Command =>
```

Auto
The *auto* test checks the Raster Op chips extensively. There is a Raster Op chip for each bit plane in the frame buffer.

4.11. DAC Test Menu

Type *da* from the Manual to select DAC test menu. DAC stands for digital to analog converter. These devices are used to convert digital color values to the proper voltages in the color monitor. Select the appropriate item to do the intended visual test.

```
Sun-3 Color Board Diag (SINGLE/DOUBLE BUF) Rev:X XX/XX/87 DAC Test Menu

  Rhramp        Print Horizontal Red Ramp
  Ghramp        Print Horizontal Grn Ramp
  Bhramp        Print Horizontal Blu Ramp
  Whramp        Print Horizontal White Ramp
  RVramp        Print Vertical Red Ramp
  GVramp        Print Vertical Grn Ramp
  BVramp        Print Vertical Blu Ramp
  WVramp        Print Vertical White Ramp
  RGBw          Print Simultaneous RGBW horizontal Ramps
  BORder        Print Screen Borders x=(0:1152) y=(0:899)
  Alter         Write alternating bars of color to test glitches
  AUto          Continuous auto tests
  Stab          Test Screen Stability

Command =>
```

Rhramp
The *Horizontal Red Ramp test* draws a shaded black-red-black image horizontally across the screen. This test checks the linearity of the red DAC.

Ghramp
The *Horizontal Green Ramp test* draws a shaded black-green-black image horizontally across the screen. This test checks the linearity of the green DAC.

Bhramp
The *Horizontal Blue Ramp test* draws a shaded black-blue-black image horizontally across the screen. This test checks the linearity of the blue DAC.

Whramp
The *Horizontal White Ramp test* draws a shaded black-white-black image horizontally across the screen. This test checks the linearity of all three DACs working together.

RVramp
The *Vertical Red Ramp test* draws a shaded black-red-black image vertically down the screen. This test checks the linearity of the red DAC.

GVramp
The *Vertical Green Ramp test* draws a shaded black-green-black image vertically down the screen. This test checks the linearity of the green DAC.

BVramp
The *Vertical Blue Ramp test* draws a shaded black-blue-black image vertically down the screen. This test checks the linearity of the blue DAC.

WVramp
The *Vertical White Ramp test* draws a shaded black-white-black image vertically down the screen. This test checks the linearity of all three DACs working together.

RGBw
The *RGBW horizontal Ramp test* draws red, green, blue, and white horizontal ramp images on the screen in sequence.

BORder
The *Screen Border test* draws a dark screen with a white line around the borders. This checks the beam deflection circuitry.

Alter
The *Color Bar test* writes alternating bars of color across the screen.

AUto
Continuous auto tests runs through the tests listed above until a `q` is typed on the keyboard.

Stab
The *Screen Stability test* displays a gray pattern on the entire screen. Check the display for consistent shading and color.

4.12. Error messages

If the diagnostic encounters a hardware failure, it prints the error message on both the screen and to the Exec's error log file. At the beginning of each error message a test number is displayed to indicate the test which failed. The following table describes each test number.

Table 4-1 *Color3 Error Message Table*

Number	*Test*	*Number*	*Test*
0	Status Reg	21	FB Word Memory Plane 0
1	Plane Mask Reg	22	FB Word Memory Plane 1
2	Reserved	23	FB Word Memory Plane 2
3	Reserved	24	FB Word Memory Plane 3
4	Reserved	25	FB Word Memory Plane 4
5	Reserved	26	FB Word Memory Plane 5
6	Interrupts	27	FB Word Memory Plane 6
7	Shadow Color Map	28	FB Word Memory Plane 7
8	FB Word-Memory	29	Interrupt Vector Reg
9	FB Pixel-Memory	30	ECL Color Map
10	ROPC Plane 0	31	FB Word-Memory (Set B)
11	ROPC Plane 1	32	FB Pixel-Memory (Set B)
12	ROPC Plane 2	33	DB Wait Bit
13	ROPC Plane 3	34	Frame Count Reg
14	ROPC Plane 4	35	Dma Base Loading
15	ROPC Plane 5	36	Dma Width Counting
16	ROPC Plane 6	37	Double Buffering Reg
17	ROPC Plane 7	38	Dma Base Reg
18	Pix-mode Plane Masking	39	Dma Width Reg
19	Word-mode Plane Masking	40	Byte Write to ROPC
20	ROPC Pixel Memory		

The address of the bad hardware shown in an error message is relative. It is an offset from the starting address of the Color Board. For example, the first address of the word mode frame buffer is 0x0, and the first address of pixel mode frame buffer is 0x100000.

4.13. Glossary

DAC
Digital to Analog Converter

DMA
Direct Memory Access.

Exec
SunDiagnostic Executive, the multi-tasking environment under which these diagnostics run.

ROPC
RasterOP Chip.

FB Frame Buffer

5

Sun CPU Diagnostic

5

Sun CPU Diagnostic

5.1. General Description

The Sun CPU Diagnostic contains tests to exercise and debug critical components on any Sun CPU Board. The CPU board is the heart of all Sun systems.

Test patterns provide flexible sequencing and control of the tests. All test primitives can be run on command. This feature is useful for isolating problems during debugging. At a higher level, a default test sequence is provided for your convenience.

This diagnostic covers the following components on the CPU board :

- Time-Of-Day Clock (Intersil 7170) — for Sun-3 and Sun-4

  (National MM58167) — Sun-2 VME and Multibus
- System Enable Register — Sun-3 only
- Floating-Point Coprocessor (Motorola MC68881 FPC) — Sun-3 only
- Interrupt Register and Interrupts
- PROMs : IDPROM, EEPROM, BOOT PROM — Sun-3 only
- Serial Ports A and B (Zilog 8530 SCC)

5.2. Hardware Requirements

The Sun-3 CPU Diagnostic runs on any system configuration that meets the requirements below :

- You can run the Serial Ports tests without a loopback (i.e. using internal loopback) or with one of the two different kinds of (external) loopbacks: loopback connectors (A-to-A and B-to-B) or loopback cables (A-to-B and B-to-A). (Refer to *Chapter 1* for pin assignments).
- The Floating-Point Coprocessor tests require the FPC to be on-board. The exception is the Probe test, which can be used to detect the presence of an FPC and does not require it to be on-board.

5.3. Command-line Parameters

All tests in Sun CPU Diagnostic accept the following command-line parameters:

`Pass=`

This argument controls the number of times a test is repeated. The number entered after the equals sign should be a non-negative decimal integer. If the argument is 0 or *, the test is repeated indefinitely. If not specified, the current value of the environment variable *Pass=* is used. In the following

sections, a default *Pass=* is a number used if *Pass=* is not specified on the command-line AND the environment variable *Pass=* is not set.

5.4. Looping on Read and Write

While looping on read or write, the following keys can be used to change the address being looped on or the pattern to write with:

- Space bar, [=], [>], or [.] (period) keys are used to increment the address.
- [-], (hyphen), [<], or [,] (comma) keys are used to decrement the address.
- For a read loop, the [+] key is also used to increment the address, but for a write loop, it is used to increment the pattern.
- For a read loop, the [_] (underbar) key is also used to decrement the address, but for a write loop, it is used to decrement the pattern.
- A hex value "left-shifts" itself into the address; for example, if the current address is 0x1234, pressing [E] will make it 0x234E.
- To "left-shift" a pattern during a write loop, press [P] before the hex digit ; for example, if current pattern is 0x12, pressing [P] then [A] makes it 0x2A.
- Any non-hexadecimal entry terminates the loop.

5.5. Main Menu

The Main Menu is the first menu displayed on the screen after the Exec loads the diagnostic. From Main Menu, you select which logical block of hardware components needs to be tested, then enter the appropriate command in order to switch to the selected menu. This is how the Main Menu looks on the screen:

```
Sun CPU Diagnostic Rev x.x  MM/DD/YY Main Menu

All          Execute ALL CPU Board Tests
Default      Default CPU test sequence
Quick        Quick CPU test sequence

Clock        Time-Of-Day Clock Tests Menu
Dcp              Data Ciphering Processor Test Menu
Enable       System Enable Tests Menu *
Epc          Floating-Point Coprocessor Tests Menu *
Interrupt    Interrupt Tests Menu
Prom         PROM Tests Menu *
Serial       Serial Port Tests Menu

Command ==>

 * these tests do not appear on the main menu for Sun-2 systems.
```

All

The *All* command executes the following test sequence:

```
c; a <esc> e; a <esc> f; a <esc> i; a <esc> s; a <esc>
```

This sequence goes to each sub-menu in the diagnostic, and runs every test found there.

Default

The *Default* command executes the following test sequence:

```
c; d <esc> e; d <esc> f; d <esc> i; d <esc> s; d <esc>
```

This sequence goes to each sub-menu in the diagnostic and runs the default set of tests there.

Quick

The *Quick* command executes the following test sequence:

```
c; q <esc> e; q <esc> f; q <esc> i; q <esc> s; q <esc>
```

This sequence goes to each sub-menu in the diagnostic, and runs the quick set of tests there.

Clock

The *Clock Tests Menu* command displays a menu containing the tests for the Time-Of-Day Clock.

Dcp

The *Data Ciphering Test Menu* command displays a menu containing the tests for the encryption processor.

Enable
The *Enable Tests Menu* command displays a menu containing the tests for the enabling various CPU systems.

Fpc
The *FPC Tests Menu* command displays a menu containing the tests for the Floating-Point Coprocessor.

Interrupt
The *Interrupt Tests Menu* command displays a menu containing the tests for the Interrupt Circuitry.

Prom
The *PROM Tests Menu* command displays a menu containing the tests for the system PROMs.

Serial
The *Serial Tests Menu* command displays a menu containing the tests for the Serial Ports.

5.6. Clock Tests Menu

Use the following command-line parameters to set clock mode or frequency of interrupt signal:

FAst
Programs the clock to FAST mode, which makes it run about 40 times faster. Default is normal mode.

Freq=
Sets frequency of interrupt signal. Valid frequencies for normal mode are 1, 10 and 100 for 1-Hz, 10-Hz and 100-Hz signals respectively. For FAST mode, they are 1, 60 and 36 for 1-Hz, 1/minute and 1/hour signals. If not specified or an invalid value is used, a default frequency is used. This frequency depends on the test being executed. Pressing the space bar during the test "rotates" the interrupt frequency. In normal mode, pressing the space bar repeatedly "rotates" the interrupt frequency through 1-Hz, 10-Hz and 100-Hz then back to 1-Hz. Any other key aborts the test.

NOTE *Keyboard input is not acknowledged until a clock interrupt occurs ; in fast mode with 1/hour interrupt frequency, the keyboard will "hang" until the hour changes.*

Here is the Clock test menu for Sun-3 systems:

```
Sun CPU Diagnostic  Rev x.x MM/DD/YY  Clock Menu

DIsplay     Display and Calibrate Clock
Now         Set Time-Of-Day
Read        Read Clock registers loop
Test        Write then Read registers test
Write       Write Clock registers loop

All         Test then Display time (120 sec)
Default     Test then Display time ( 10 sec)
Quick       Test then Display time (   5 sec)

Command ==>
```

Here is the Clock test menu for Sun-2 systems:

```
Sun CPU Diagnostic  Rev x.x  mm/dd/yy  Clock Menu

Register        Register Test
Rollover        Rollover Test
DIsplay         Display Test

All             Execute ALL TOD2 tests (100)
Default         Execute ALL TOD2 tests (20)
Quick           Execute ALL TOD2 tests (5)
```

Following are descriptions of the various tests.

Display *Pass= FAst Freq= 12*

The *Display and Calibrate Clock test* displays the current time and date (updated every second) for a number of seconds determined by `Pass=` or until a non-space key is pressed. See the beginning of this section to set clock mode and frequency of interrupt signal. Format of display:

```
hh:mm:ss   mm/dd/yy   day_of_week   Frequency = interrupt_freq
```

12 If you enter the parameter `12`, the program displays time in 12-hour (with AM/PM) format rather than the default 24-hour or military format.

Defaults are:

```
Pass= 0
Freq= 1 (generate 1-Hz interrupt signal)
```

Now *Hour= MIn= Sec= Month= Date= Year= Week=*

The *Set Time-Of-Day* command first displays the current time on one line. If there is at least one command-line parameter, the new time is set according to the command-line parameters. Otherwise, the current time is displayed again on the next line for on-screen and interactive setting of each time field. While setting, a decimal digit "left-shifts" itself into the current field (e.g. if the current field contains "23", pressing 7 will make it "37"). A space bar moves the cursor to the next field. The Esc key recovers the current time, then finishes the setting. Any other key finishes the setting. With or without a command-line parameter, this command displays the new time on another line then pauses until a key is pressed.

Hour=

If you enter a decimal value that is less than 100 after *Hour=*, the hour is set according to that number. If the number you enter is between 100 and 9999, it is assumed to be of the form `hhmm`; and the hour and minute are set accordingly. If the number you enter is larger than 9999, it is assumed to be `hhmmss`, and the hour, minute, and second are set.

MIn=

Enter a decimal value (00-59) for after *MIn=* to set the minute.

Sec=

Enter a decimal value (00-59) after *Sec=* to set the second.

Month=MM

Enter a decimal value (01-12) after *Month=* to set the month.

Date=

If you enter a decimal value less than 100 after *Date=*, only the date is set. If the number you enter is between 100 and 9999, it is assumed to be of the form `mmdd`; and month and date are set accordingly. If you enter a value larger than 9999, it is assumed to be `mmddyy`; and the month, date, and year are set accordingly.

Year=

Enter a decimal value (00-99) after *Year=* to set the year.

Week=

Enter a decimal value from 0 (Sunday) to 6 (Saturday) after *Week=* to set the day of the week.

Notes :

- The hour to be set must be in 24-hour or military format.
- For both command-line and interactive setting, day-of-week must be a single digit between 0 (Sunday) and 6 (Saturday).
- The actual value stored in the clock register for Year is offset by 68 (decimal); for example, if `Year` is set to 68, then the actual value stored is 0; if `Year=87`, the actual value stored is 19.

Read *Pass= Offset=*

The *Read Clock registers loop* command loops on reading and displaying

contents of clock registers. To select a clock register to loop, see the *Looping on Read and Write* section at the beginning of this chapter.

Default :

```
Pass= 0
Offset= 0 (clock register to loop on, from 0x00 to 0x11)
```

Test *Pass=*

The *Write then Read Clock registers test* tests all counter and RAM registers, from 0x00 to 0x0F, by writing, then reading them. This test destroys current contents of all clock registers.

Default :

```
Pass= 1
```

Write *Pass= Offset= PATtern=*

The *Write Clock registers loop* command loops on writing, reading and displaying the contents of clock registers. This test destroys the current contents of clock registers being written to. To select a clock register to loop on or change the pattern to be written, see the *Looping on Read and Write* section at the beginning of this chapter.

Default :

```
Pass= 0
Offset= 0 (clock register to loop on, from 0x00 to 0x11)
PATtern= 0 (pattern to write to clock register)
```

All

The *All* command executes the following command sequence:

```
test pass=10  ;  display pass=120
```

Default

The *Default* command executes the following command sequence:

```
test pass=5   ;  display pass=10
```

Quick

The *Default* command executes the following command sequence:

```
test pass=2   ;  display pass=5
```

5.7. System Enable Tests Menu

NOTE *This test menu is not available for Sun-2 systems.*

Each Enable test (except the Diagnostic Switch test) toggles its associated bit within the System Enable Register OFF, then ON without doing anything to any other component that might be affected by the changing of that bit. For example, the Copy test only toggles the Copy bit, without actually copying anything to video memory. Likewise, the FPC test only toggles the FPC bit without performing any FPC operations. As a result, the Enable tests (including the Diagnostic Switch) do not produce error messages. A test can be aborted any time by pressing any key.

All Enable tests (except Diagnostic Switch) accept the following parameter:

Delay=
: controls how fast or slow to toggle the tested bit. The value after the equals sign should be a non-negative integer. The higher it is, the longer it will take between togglings.

Pass=
: Sets the number of times the test is executed.

Defaults for the Enable Menu tests are:

```
Pass= 100
Delay= 30
```

The `pass=` default is used only if the number of passes was not specified on the command line and the environment variable is not set.

```
Sun CPU Diagnostic  Rev R.RR MM/DD/YY  Enable Menu

CAche        Enable External Cache test
Copy         Enable Copy mode to video memory
DIagnostic   Diagnostic Switch test
FPA          Enable Floating-Point Accelerator
Fpc          Enable Floating-Point Coprocessor
Sdvma        Enable System DVMA test
Video        Enable Video Display test

All          Execute ALL Enable tests (100)
Default      Execute ALL Enable tests ( 20)
Quick        Execute ALL Enable tests (   5)

Command ==>
```

CAche *Pass= Delay=*
: The *Enable External Cache test* turns the Enable External Cache bit OFF, then ON in the System Enable Register.

Copy *Pass= Delay=*
: The *Enable Copy mode to video memory test* turns the Enable Copy bit OFF, then ON in the System Enable Register.

DIagnostic *Pass=*

The *Diagnostic Switch test* reads and displays the current setting of the Diagnostic Switch.

ON : switch is set to DIAGnostic position (or middle position on model using 3-positions switch).

OFF : switch is set to NORMal position.

FPA Pass= Delay=

The *Enable Floating-Point Accelerator test* turns the Enable Floating-Point Accelerator bit OFF, then ON in the System Enable Register.

Fpc *Pass= Delay=*

The *Enable Floating-Point Coprocessor (MC68881) test* turns the Enable Floating-Point Coprocessor bit OFF, then ON in the System Enable Register.

Sdvma *Pass= Delay=*

The *Enable System DVMA test* turns the Enable System DVMA bit OFF, then ON in the System Enable Register.

Video *Pass= Delay=*

The *Enable Video Display test* turns the Enable Video bit OFF, then ON in the System Enable Register. The Video display should flash during this test.

All

The *All* command executes the following command sequence:

```
set Pass=100  ;  cache-video
```

Default

The *Default* command executes the following command sequence:

```
set Pass=20   ;  cache-video
```

Quick

The *Quick* command executes the following command sequence:

```
set Pass=5    ;  cache-video
```

5.8. FPC Tests Menu

NOTE *This test menu is not available for Sun-2 systems.*

`Mcrom` test and all tests in the `Monadic` and `Dyadic` sub-menus use the Floating- Point Coprocessor (MC68881) to perform a calculations, then compare that result with the expected, software-computed value. When an FPC test is executed, the diagnostic first checks to see if an FPC is installed on the system. If it isn't, the test is not executed. No error message is printed if a test isn't executed. All bits within the FPCR (Control Register) are cleared before testing.

All tests accept this command:

`DISable`
Disable the FPC (by turning off the FPC bit within System Enable Register) before testing. Default is enable.

`Mcrom` test and all tests in the `Monadic` and `Dyadic` sub-menus accept these commands:

`PAUse`
Pause after displaying result of each pass. Press the space bar to continue; any other key aborts that test. Default is no pause.

`X=#.#`
`Mcrom` test ignores this parameter. A monadic test performs calculations using this `X` value. A dyadic test uses this `X` value as its first argument. If `X=` is not specified, a random value is generated for it.

`Y=#.#`
`Mcrom` and monadic tests ignore this parameter. A dyadic test uses this `Y` value (or a random value if it is not specified) as its second argument.

```
Sun CPU Diagnostic  Rev xx  MM/DD/YY  Floating-Point Menu

Mcrom    FMOVECR Move Constant ROM
MOvem    FMOVEM  Move Multiple Data Register
Probe    Probe Floating-Point Coprocessor

1        Switch to Monadic Tests Menu
2        Switch to Dyadic  Tests Menu

All      Execute ALL FPC tests
Default  same as All but shorter
Quick    same as All but very quick

Command ==>
```

`Mcrom` *Pass= PAUse=*
The *Move Constant ROM test* is used to obtain constants from the FPC's on-chip opback Connector for SCSI Bus Connector"

You must install a loopback connector on the SCSI connector at the backpanel edge of the CPU Board in order to test the SCSI Bus, using the Extended Menu

Tests provided by the x Monitor command. This shorting connector is a DB50 connector. A list of interconnects is provided below. The set of chip and connector (*Conn*) pins on ROM, display them and compare with expected values. If a constant has no expected value, it is displayed both in double-precision real (decimal) format and hexadecimal format. *Pass=* is used to control how many constants are displayed. For example, `Pass=10` displays constant #0 through constant #9

Default :

`Pass= 64` (which is the size of the FPC ROM table of constants)

MOvem *Pass=*

The *Move Multiple Data Register test* is used to move all FPC Data (FP0 to FP7) and Control registers (FPCR, FPSR and FPIAR) to and from memory. No comparison or verification is performed. The default number of passes is 100.

Probe *Pass=*

The *Probe Floating-Point Coprocessor* instruction is used to detect for the presence of the FPC. While this test is running, the space bar can be used to toggle enable/disable FPC. The default number of passes is zero.

1

The *Monadic* command brings up the Monadic sub-menu.

2

The *Dyadic* command brings up the Dyadic sub-menu

All

The *All* command executes the following command sequence:

```
mc p=64 ; mo p=100 ; 1 ; all <esc> 2 ; all <esc>
```

Default

The *Default* command executes the following command sequence:

```
mc p=64 ; mo p=20  ; 1 ; def <esc> 2 ; def <esc>
```

Quick

The *Quick* command executes the following command sequence:

```
mc p=64 ; mo p=5   ; 1 ; qui <esc> 2 ; qui <esc>
```

FPC Monadic Tests Sub-menu

NOTE *This test menu is not available for Sun-2 systems.*

This menu executes FPC functions that require only one argument. As mentioned in the beginning of the previous section, all of these tests operate on the variable *X=#.#*. If *X=* is not specified on the command-line, a random value is used; the range of this value depends on the test being executed. Do not enter the symbol *#.#*; enter a value such as 50.0 after X=. The default number of passes for these tests is 100.

```
Sun CPU Diagnostic  Rev x.x  MM/DD/YY  FPC Monadic Menu

2               FTWOTOX         2 to the X power
ABsolute        FABS            Absolute
Cosine          FCOS            Cosine
Exponential     FETOX           e to the X power
Hcos            FCOSH           Hyperbolic Cosine
HSin            FSINH           Hyperbolic Sine
HTan            FTANH           Hyperbolic Tangent
Integer         FINT            Get integer portion
LGt             FLOG10          Log base 10
Log             FLOGN           Log base e
Negate          FNEG            Negate a number
Root            FSQRT           Square Root
Sine            FSIN            Sine
Tangent         FTAN            Tangent
TEst            FTST            Test a number

All             Execute ALL     Monadic tests (100 times)
Default         same as ALL     (20 times)
Quick           same as ALL     (5 times)

Command ==>
```

2 *Pass= X=#.# PAUse*

The *2 to the X power test* uses the FTWOTOX instruction to calculate 2 to the X power, then compares it with the results from software calculation.

The default value of *X=* is a random value (from -50.0 to +50.0).

ABsolute *Pass= X=#.# PAUse*

The *Absolute value test* uses the FABS instruction to calculate the absolute value of X, then compares it with the results from software calculation.

The default value of *X=* is a random (from -100000.0 to +100000.0).

Cosine *Pass= X=#.# PAUse*

The *Cosine test* uses the FCOS instruction to calculate cosine of X (X must be in radians), then compares it with the results from software calculation.

The default value of *X=* is a random number (from -50.0 to +50.0).

Exponential *Pass= X=#.# PAUse*

The *e to the X power test* uses the FETOX instruction to calculate e (2.71828...) to the X power, then compares it with the results from software calculation.

The default value of *X=* is a random number (from -50.0 to +50.0).

Hcos *Pass= X=#.# PAUse*

The *Hyperbolic Cosine test* uses the FCOSH instruction to calculate hyperbolic cosine of X, then compares it with the results from software calculation.

The default value of *X=* is a random number (from -50.0 to +50.0).

Hsin *Pass= X=#.# PAUse*

The *Hyperbolic Sine test* uses the FSINH instruction to calculate the hyperbolic sine of X, then compares it with the results from software calculation.

The default value of *X=* is a random number (from -50.0 to +50.0).

HTan *Pass= X=#.# PAUse*

The *Hyperbolic Tangent test* uses the FTANH instruction to calculate the hyperbolic tangent of X, then compares it with the results from software calculation.

The default value of *X=* is a random number (from -50.0 to +50.0).

Integer *Pass= X=#.# PAUse*

The *Get integer portion test* uses the FINT instruction to obtain the integer portion of X (using "rounded to nearest" mode since the FPCR is cleared before testing), then compares it with the results from software calculation.

The default value of *X=* is a random number (from -100000.0 to +100000.0).

LGt *Pass= X=#.# PAUse*

The *Log base 10 test* uses the FLOG10 instruction to calculate the common log of the absolute value of X, then compares it with the results from software calculation. Note that when `X=0.0`, the result is Infinity. The default value of *X=* is a random number (from > 0.0 to +50.0).

Log *Pass= X=#.# PAUse*

The *Log base e test* uses the FLOGN instruction to calculate the natural log of the absolute value of X then compares it with the results from software calculation. Note that when `X=0.0`, the result is Infinity.

The default value of *X=* is a random number (from > 0.0 to +50.0).

Negate *Pass= X=#.# PAUse*

The *Negate a number test* uses the FNEG instruction to negate X, then compares it with the result from software calculation.

The default *X=* value is a random number (from -100000.0 to +100000.0).

Root *Pass= X=#.# PAUse*

The *Square root test* uses the FSQRT instruction to calculate the square root of (absolute value of) X, then compares it with the results from software calculation.

The default value of *X=* is a random number (from 0 to +100000.0).

Sine *Pass= X=#.# PAUse*

The *Sine test* uses the FSIN instruction to calculate the sine of X (X must be in radians), then compares it with the results from software calculation.

The default value of *X=* is a random number (from -50.0 to +50.0).

Tangent *Pass= X=#.# PAUse*

The *Tangent test* uses the FTAN instruction to calculate the tangent of X (X must be in radians) then compares it with the results from software calculation.

The default value of *X=* is a random number (from -50.0 to +50.0).

TEst *Pass= X=#.# PAUse*

The *Test a number* test uses the FTST instruction to compare X with 0, then compares the Condition Code Byte (within FP Status Register) with the results from software emulation :

X < 0	Condition Code = 8
X = 0	Condition Code = 4
X > 0	Condition Code = 0

The default *X=* value is a random number (from -100000.0 to +100000.0).

All

The *All* command executes the following command sequence:

```
set Pass=100 ; 2-test
```

Default

The *Default* command executes the following command sequence:

```
set Pass=20  ; 2-test
```

Quick

The *Quick* command executes the following command sequence:

```
set Pass=5   ; 2-test
```

FPC Dyadic Tests Sub-menu

NOTE *This test menu is not available for Sun-2 systems.*
This menu executes FPC functions that require two arguments. As mentioned previously, all of these tests operate on the variables X=#.# and Y=#.#; if X= an X= and Y= are not specified on the command-line, random values are assigned to them. Do not enter the #.# symbols; enter a value such as 100000.0 after *X*= or *Y*=. The default number of passes is 100.

```
Sun CPU Diagnostic       Rev R.RR      MM/DD/YY     FPC Dyadic Menu

ADd          FADD        Add 2 numbers
Compare      FCMP        Compare 2 numbers
DIvide       FDIV        Divide 2 numbers
Multiply     FMUL        Multiply 2 numbers
SDiv         FSIGDIV     Single Precision Divide
SMul         FSIGMUL     Single Precision Multiply
Subtract     FSUB        Subtract 2 numbers

All          Execute ALL Dyadic tests (100 times)
Default      same as ALL (20 times)
Quick        same as ALL (5 times)

Command ==>
```

ADd *Pass= X=#.# Y=#.# PAUse*
The *Add 2 numbers* command uses the FADD instruction to perform X + Y, then compares it with the results from software calculation.

Default *X*= *Y*= values are:

```
X= random (from -100000.0 to +100000.0)

Y= random (from -100000.0 to +100000.0)
```

Compare *Pass= X=#.# Y=#.# PAUse*
The *Compare 2 numbers* command uses the FCMP instruction to compare X and Y, then compares the Condition Code Byte (within FP Status Register) with the results from software emulation :

X > Y Condition Code = 0
X < Y Condition Code = 8
X = Y >= 0 Condition Code = 4
X = Y < 0 Condition Code = 12

Default *X*= *Y*= values are:

```
X= random (from -100000.0 to +100000.0)

Y= random (from -100000.0 to +100000.0)
```

DIvide *Pass= X=#.# Y=#.# PAUse*

The *Divide 2 numbers* command uses the toFDIVinstruction results from software calculation. Note that when `Y=0.0`, the result is Infinity.

Default *X=* *Y=* values are:

```
X= random (from -100000.0 to +100000.0)

Y= random (from -100000.0 to +100000.0)
```

Multiply *Pass= X=#.# Y=#.# PAUse*

The *Multiply 2 numbers* command uses the FMUL instruction to perform X multiplied by Y, then compares it with the results from software calculation.

Default *X=* *Y=* values are:

```
X= random (from -100000.0 to +100000.0)

Y= random (from -100000.0 to +100000.0)
```

SDiv *Pass= X=#.# Y=#.# PAUse*

The *Divide 2 numbers using single-precision* command uses the FSIGDIV instruction to perform X divided by Y, using single-precision calculation, then compares it with the result from software calculation.
Note that when `Y=0.0`, the result is Infinity.

Default *X=* *Y=* values are:

```
X= random (from -100000.0 to +100000.0)

Y= random (from -100000.0 to +100000.0 and not equal to 0.0)
```

SMul *Pass= X=#.# Y=#.# PAUse*

The *Multiply 2 numbers using single-precision* command uses the FSIGMUL instruction to perform X multiplied by Y using single-precision calculation, then compares it with the result from software calculation. Default *X=* *Y=* values are:

```
X= random (from -100000.0 to +100000.0)

Y= random (from -100000.0 to +100000.0)
```

Subtract *Pass= X=#.# Y=#.# PAUse*

The *Subtract 2 numbers* command uses the FSUB instruction to perform X minus Y, then compares it with the result from software calculation. Default *X=* *Y=* values are:

```
X= random (from -100000.0 to +100000.0)

Y= random (from -100000.0 to +100000.0)
```

All

The *All* command executes the following command sequence:

```
set Pass=100 ; add-multiply ; subtract
```

Default

The *Default* command executes the following command sequence:

```
set Pass=20  ; add-multiply ; subtract
```

Quick

The *Quick* command executes the following command sequence:

```
set Pass=5   ; add-multiply ; subtract
```

5.9. Interrupt Tests Menu

```
Sun CPU Diagnostic  Rev n.nn MM/DD/YY  Interrupt Menu

Read          Read Interrupt register loop
Test          Write/Read Interrupt register test
Write         Write Interrupt register loop

1             Level 1 Interrupt, Software
2             Level 2 Interrupt, Software
3             Level 3 Interrupt, Software
4             Level 4 Interrupt, Video
5             Level 5 Interrupt, Clock
6             Level 6 Interrupt, Serial Port
7             Level 7 Interrupt, Clock

All           Execute all test (100 times)
Default       Execute all test (20  times)
Quick         Execute all test (5   times)

Command ==>
```

NOTE *The* `Read`, `Test` *and* `Write` *menu selections are not available for Sun-2 systems.*

Read *Pass=*

The *Read Interrupt Register loop* command loops while reading and displaying the contents of the Interrupt Register, until any key is pressed.

Default :

```
Pass= 0
```

Test *Pass=*

The *Write then Read Interrupt Register test* command tests the Interrupt Register by writing, then reading it. The previous contents of the Interrupt Register are saved before test, then recovered after testing.

Default :

```
Pass= 1
```

Write *Pass= PATtern=*

The *Write Interrupt Register loop* command loops while writing, reading and displaying the contents of the Interrupt Register, until a key is pressed. The previous contents of the Interrupt Register are saved before test, then recovered after testing.

Default :

```
Pass= 0
```

```
PATtern=0 (pattern to write to Interrupt Register)
```

1, 2, 3 and **4**

The *Level 1, 2, 3 and 4 Interrupt tests* turns ON the appropriate bit within the Interrupt Register, then waits for the expected interrupt to occur.

Syntax : *1 or 2 or 3 or 4 Pass= Delay=*

Default :

```
Pass= 100
```

```
Delay= 100 (how long to wait for the interrupt to occur).
```

5 *Pass= FAst Freq=*

7 *Pass= FAst Freq=*

The *Level 5 and 7 Clock Interrupt tests* programs the clock to generate an interrupt signal periodically, turn ON the appropriate bit within the Interrupt Register, then wait for the expected interrupt to occur. See the beginning of the "Clock Tests Menu" section to set clock mode and frequency of interrupt signal.

Default :

```
Pass= 100
Freq= 100
```

(generate 100-Hz interrupt signal).

6 *Pass= Delay=*

The *Level 6, Serial Port Interrupt test* programs the Serial Port SCC, channel A to generate an interrupt signal periodically, turn ON the appropriate bit within the Interrupt Register, then wait for the expected interrupt to occur. For Sun-3/50, the level 6 Auto Vector (vector #30 decimal; 0X1E hex) is used to catch the interrupt signal. For others, the User Defined Vector #10 (or vector #74 decimal; 0x4A hex) is used.

Default :

```
Pass= 100
Delay= 100
```

(how long to wait for the interrupt to occur).

All

The *All* command executes the following command sequence:

```
set Pass=100  ;  test  ;  1-7
```

Default

The *Default* command executes the following command sequence:

```
set Pass=20    ;  test   ;  1-7
```

Quick

The *Quick* command executes the following command sequence:

```
set Pass=5     ;  test   ;  1-7
```

5.10. PROM Tests Menu

NOTE *This test menu is not available for Sun-2 systems.*

All PROM tests accept the following parameters :

Offset=

A hex number specifying starting offset of the PROM (either EEPROM, EPROM or IDPROM) to be read, displayed, tested, or written. Default is 0.

Length=

A hex number specifying the length of region (in bytes). Default is length of the PROM being acted on.

For tests that write to EEPROM, the previous contents of the EEPROM are saved before testing and recovered afterward. They also accept the following:

Delay=

Controls the delay in milli-seconds after each write to an EEPROM location. Default is 11 milli-seconds.

PATtern=

A hex pattern to be written to EEPROM locations.

While PROM contents are being displayed, pressing any key freezes screen output. At that point, Esc terminates the display, while any other key resumes it.

If you use a command line to invoke one of the PROM tests, such as

```
> b stand/exec
Command=> d;
Command=> cpu;p;ef
```

and you DO NOT enter a `pass=` parameter, the standard SunDiagnostic Executive default of one pass is used, and the test will run once and stop. If you enter a command such as **`cpu;p;"ef pass="`** with no number after it, the PROM test's default number of passes is used.

```
Sun CPU Diagnostic  Rev x.x MM/DD/YY PROM Menu

EEprom      Display content of EEPROM
EFill       EEPROM Fill and Test
EMarch      EEPROM March Test
ERead       EEPROM Read Loop
EWrite      EEPROM Write Loop

Eprom       Display content of EPROM
ELoop       EPROM Read Loop

Idprom      Display content of IDPROM
IRead       IDPROM Read Loop

All         EMarch (9 passes), EFill (18 passes)
Default     Execute EEPROM March Test (9 passes)
Quick       Execute EEPROM March Test (1 pass)

Command ==>
```

EEprom *Pass= Offset= Length=*

The *Display content of EEPROM* command displays the contents of the system EEPROM.

Default

`Pass= 1`

`Offset= 0x000`

`Length= 0x800` (2 Kbytes)

EFill *Pass= Offset= Length= PATtern=*

The *EEPROM Fill and Test* command fills the system EEPROM with a constant pattern, then verifies it.

Default

The test performs 18 passes — nine using a 0's pattern and nine using a 0ffff pattern.

Offset= 0x000

The default Length= 0x800 (2 Kbytes). The minimum length you may enter is0x10, which is the hexadecimal equivalent of 16.

If you enter PATtern= with no argument, the patterns 0x00, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80 are used one pattern per pass. In other words, the first pass uses 0's, the second pass uses 1's, and so on.

EMarch *Pass= Offset= Length= PATtern=*

The *EEPROM March Test* tests the system EEPROM using the marching 1's method.

Default:

`Pass= 9`

```
Offset= 0x000

Length= 0x800 (2 Kbytes)
```

If you enter PATtern= with no argument, the patterns 0x00, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80 are used one pattern per pass. In other words, the first pass uses 0's, the second pass uses 1's, and so on.

ERead *Pass= Offset=*

The *EEPROM Read loop test* loops while reading and displaying contents of locations in EEPROM. To select a location in EEPROM to loop on, see the *Looping on Read and Write* section at the beginning of this chapter.

Default :

```
Pass= 0

Offset= 0x000
```

EWrite *Pass= Offset= PATtern=*

The *EEPROM Write loop test* loops while writing, reading and displaying contents of locations in EEPROM. To select a location in EEPROM to loop on or to change the pattern to be written, see the *Looping on Read and Write* section at the beginning of this chapter.

Default :

```
Pass= 0

Offset= 0x000

PATtern= 0x00
```

Eprom *Pass= Offset= Length=*

The *Display content of EPROM* command displays the contents of the system Boot PROM.

Default :

```
Pass= 1

Offset= 0x0000

Length= 0x10000 (64 Kbytes)
```

Eloop *Pass= Offset=*

The *EPROM Read loop test* command loops while reading and displaying contents of locations in the Boot PROM. To select a location in the Boot PROM to loop on, see the *Looping on Read and Write* section at the beginning of this chapter.

Default :

```
Pass= 0

Offset= 0x0000
```

Idprom *Pass= Offset= Length=*

The *Display content of IDPROM* command displays the contents of the system's IDPROM.

Default :

```
Pass= 1

Offset= 0x00

Length= 0x20   (32 bytes)
```

IRead *Pass= Offset=*

The *IDPROM Read loop test* loops while reading and displaying contents of locations in IDPROM. To select a location in IDPROM to loop on, see the "Looping on Read and Write" section at the beginning of this chapter.

Default :

```
Pass= 0

Offset= 0x00
```

All

The *All* command executes the following command sequence:

```
emarch pass=9  ;  efill pass=9  ;  efill pass=9 pattern=FF
```

Default

The *Default* command executes the following command sequence:

```
emarch pass=9
```

Quick

The *Quick* command executes the following command sequence:

```
emarch pass=1
```

5.11. Serial Port Tests Menu

All tests in the three sub-menus of this Menu accept the command-line parameter *Delay=*, which specifies the delay (in arbitrary units) after each output or transmission.

All commands in this menu (but NOT those in the three sub-menus) and the Serial command itself (in Main Menu) accept `Config=`, which selects the proper loopback to use. Its valid values are:

- Cross, Cacb, Cbca, AB or BA : a loopback cable between port A and B.
- External, Eaeb or Ebea : a loopback connector each on both port A and B.
- None, Internal, Iaib or Ibia : no loopback on either port; test both ports in internal or local loopback mode.
- A, EASb or SBEa : only port A has a loopback connector; skip port B.
- B, EBSa or SAEb : only port B has a loopback connector; skip port A.
- SKIP, SASB or SBSA : skip both ports.
- EAIb or IBEa : only port A has a loopback connector; test port B internally.
- EBIa or IAEb : only port B has a loopback connector; test port A internally.
- IASb or SBIa : test port A in internal loopback mode ; skip port B.
- IBSa or SAIB : test port B in internal loopback mode ; skip port A.

```
Sun CPU Diagnostic Rev x.x MM/DD/YY  Serial Main Menu

ASync       Asynchronous Tx ==> Rx Tests
Modem       Modem Signals Tests
Register    Write then Read Registers Tests

All         Execute ALL Serial tests
Default     Execute a default set of Serial tests
Quick       Execute a quick   set of Serial tests

Command ==>
```

ASync
The *Asynchronous Tests* command brings up the Asynchronous Tests Sub-menu.

AModem
The *Modem Signal Tests* command brings up the Modem Tests Sub-menu.

ARegister
The *Register Tests* command brings up the Register Tests Sub-menu.

All

The *All* command executes the following command sequence:

```
async ; all <esc> modem ; all <esc> register ; all <esc>
```

This sequence is stored in the variable `A_Serial`.

Default

The *Default* command executes the following command sequence:

```
async ; def <esc> modem ; def <esc> register ; def <esc>
```

This sequence is stored in the variable `D_Serial`.

Quick

The *Quick* command executes the following command sequence:

```
async ; qui <esc> modem ; qui <esc> register ; qui <esc>
```

Asynchronous Tests Sub-menu

All asynchronous tests accept the command-line parameter *Baud=* which selects the baud rates to run a test at. Its valid values are :

Baud=All

Runs a test at all possible baud rates.

Baud=Default

Runs a test at 2 baud rates : 300 and 9600.

Baud=Quick

Runs a test at 9600.

Baud=some number

Runs a test at the specified baud rate.

If not specified or improperly specified, `Baud=Default` is assumed.

```
Sun CPU Diagnostic  Rev x.x  MM/DD/YY SCC Async Menu

AAe          Tx Channel A ==> Rx Channel A (external)
AAI          Tx Channel A ==> Rx Channel A (internal)
AB           Tx Channel A ==> Rx Channel B
AS           Tx Channel A Signal test

BA           Tx Channel B ==> Rx Channel A
BBe          Tx Channel B ==> Rx Channel B (external)
BBI          Tx Channel B ==> Rx Channel B (internal)
BS           Tx Channel B Signal test

All          All Async tests at all baud rates
Default      same as 'All' but use default baud rates
Quick        same as 'All' but use spec baud rate

Command ==>
```

The default parameters for the SCC Async tests are:

```
Pass= 256

Delay= 100

Baud= 300 or 9600
```

AAe *Pass= Delay= Baud=*
The *Tx Channel A ==> Rx Channel A (external)* command transmits all possible data patterns at pre-selected baud rates to the TxD output on port A (pin 15 on the SCC and pin 2 on the RS-232C connector), then compares it with data received from its RxD input (pin 13 on the SCC and pin 3 on the RS-232C connector). This test requires a loopback connector on port A.

AAI *Pass= Delay= Baud=*
The *Tx Channel A ==> Rx Channel A (internal)* command programs the SCC to local loopback mode, then transmits all possible data patterns at pre-selected baud rates to the TxD output on port A (pin 15 on the SCC and pin 2 on the RS-232C connector), then compares it with data received from its internal RxD input. This test does not require any loopback.

AB *Pass= Delay= Baud=*
The *Tx Channel A ==> Rx Channel B* command transmits all possible data patterns at pre-selected baud rates to the TxD output on port A (pin 15 on the SCC and pin 2 on the RS-232C connector, A end) then compares it with data received from the RxD input on port B (pin 27 on the SCC and pin 3 on the RS-232C connector, B end). This test requires a loopback cable between port A and port B.

AS *Pass= Delay= Baud=*
The *Tx Channel A Signal test* loops while transmitting a pattern to TxD output of port A (pin 15 on the SCC and pin 2 on the RS-232C connector). This test does not require any loopback.

BA *Pass= Delay= Baud=*
The *Tx Channel B ==> Rx Channel A* command transmits all possible data patterns at pre-selected baud rates to the TxD output on port B (pin 25 on the SCC and pin 2 on the RS-232C connector, B end), then compares it with data received from the RxD input on port A (pin 13 on the SCC and pin 3 on the RS-232C connector, A end). This test requires a loopback cable between port A and port B.

BBe *Pass= Delay= Baud=*
The *Tx Channel B ==> Rx Channel B (external)* command transmits all possible data patterns at pre-selected baud rates to the TxD output on port B (pin 25 on the SCC and pin 2 on the RS-232C connector), then compares it with data received from its RxD input (pin 27 on the SCC and pin 3 on the RS-232C connector). This test requires a self loopback on port B.

BBI *Pass= Delay= Baud=*
The *Tx Channel B ==> Rx Channel B (internal)* command programs the SCC to local loopback mode, then transmits all possible data patterns at pre-selected baud rates to the TxD output on port B (pin 25 on the SCC and

pin 2 on the RS-232C connector), then compares it with data received from its internal RxD input. This test does not require any loopback.

BS *Pass= Delay= Baud=*
Then *Tx Channel B Signal test* loops while transmitting a pattern to TxD output of port B (pin 25 on the SCC and pin 2 on the RS-232C connector). This test does not require any loopback.

All
The *All* command executes the command sequence stored in the variable `A_SAsync`. The baud rates are automatically set.

Default
The *Default* command executes the command sequence stored in the variable `D_SAsync`. The baud rates are automatically set.

Quick
The *Quick* command executes the command sequence stored in the variable `Q_SAsync`. The baud rates are automatically set.

Modem Tests Sub-menu

All modem tests accept the command-line parameter `Signal=`, which specifies the level of output signal. Its valid values are :

Signal=Low
Produces a high (+5v) on the tested pin of SCC and a low (-6v) on the corresponding pin of RS-232C connector.

Signal=High
Produces a low (0v) on the tested pin of SCC and a high (+6v) on the corresponding pin of RS-232C connector ;

Signal=Pulse
Alternates between Low and High.

If not specified or improperly specified, `Signal=Pulse` is assumed. While testing, pressing the space bar repeatedly changes the signal from Pulse to Low to High, then back to Pulse. Pressing any other non-space key terminates the test.

Here is the SCC Modem Test menu:

```
Sun CPU Diagnostic  Rev x.x  MM/DD/YY  SCC Modem Menu

DAa          DTR Channel A ==> DSR Channel A
DAB          DTR Channel A ==> DSR Channel B
DAS          DTR Channel A Signal test
DBA          DTR Channel B ==> DSR Channel A
DBb          DTR Channel B ==> DSR Channel B
DBS          DTR Channel B Signal test

RAa          RTS Channel A ==> CTS Channel A
RAB          RTS Channel A ==> CTS Channel B
RAS          RTS Channel A Signal test
RBA          RTS Channel B ==> CTS Channel A
RBb          RTS Channel B ==> CTS Channel B
RBS          RTS Channel B Signal test

All          Execute applicable modem lines tests
Default      same as All
Quick        same as All

Command ==>
```

Parameter defaults for the SCC Modem tests are:

```
Pass= 0
Delay= 100
Signal= Pulse
```

Following are descriptions of each Modem Test sub-menu choice.

DAa *Pass= Delay= Signal=*

The *DTR Channel A ==> DSR Channel A* command puts out a signal level to the DTR output of port A (pin 16 on the SCC and pin 20 on the RS-232C connector), then compares it with signal from its DSR input (pin 11 (sync A) on the SCC and pin 6 on the RS-232C connector). This test requires a loopback connector on port A.

DAB *Pass= Delay= Signal=*

The *DTR Channel A ==> DSR Channel B* command puts out a signal level to the DTR output of port A (pin 16 on the SCC and pin 20 on the RS-232C connector, A end), then compares it with signal from the DSR input of port B (pin 29 (sync B) on the SCC and pin 6 on the RS-232C connector, B end). This test requires a loopback cable between port A and port B.

DAS *Pass= Delay= Signal=*

The *DTR Channel A Signal test* loops while transmitting a signal level to the DTR output of port A (pin 16 on the SCC and pin 20 on the RS-232C connector). This test does not require loopback.

DBA *Pass= Delay= Signal=*
The *DTR Channel B ==> DSR Channel A* command puts out a signal level to the DTR output of port B (pin 24 on the SCC and pin 20 on the RS-232C connector, B end), then compares it with signal from the DSR input of port A (pin 11 (sync A) on the SCC and pin 6 on the RS-232C connector, A end). This test requires a loopback cable between port A and port B.

DBb *Pass= Delay= Signal=*
The *DTR Channel B ==> DSR Channel B* command puts out a signal level to the DTR output of port B (pin 24 on the SCC and pin 20 on the RS-232C connector), then compares it with signal from its DSR input (pin 29 (sync B) on the SCC and pin 6 on the RS-232C connector). This test requires a loopback connector on port B.

DBS Pass= Delay= Signal=
The *DTR Channel B Signal test* loops while transmitting a signal level to the DTR output of port B (pin 24 on the SCC and pin 20 on the RS-232C connector). This test does not require loopback.

RAa *Pass= Delay= Signal=*
The *RTS Channel A ==> CTS Channel A* command puts out a signal level to the RTS output of port A (pin 17 on the SCC and pin 4 on the RS-232C connector), then compares it with signal from its CTS input (pin 18 on the SCC and pin 5 on the RS-232C connector). This test requires a loopback connector on port A.

RAB *Pass= Delay= Signal=*
The *RTS Channel A ==> CTS Channel B* command puts out a signal level to the RTS output of port A (pin 17 on the SCC and pin 4 on the RS-232C connector, A end), then compares it with the signal from the CTS input of port B (pin 22 on the SCC and pin 5 on the RS-232C connector, B end). This test requires a loopback cable between port A and port B.

RAS *Pass= Delay= Signal=*
The *RTS Channel A Signal test* loops while transmitting a signal level to the RTS output of port A (pin 17 on the SCC and pin 4 on the RS-232C connector). This test does not require loopback.

RBA *Pass= Delay= Signal=*
The *RTS Channel B ==> CTS Channel A* command puts out a signal level to the RTS output of port B (pin 23 on the SCC and pin 4 on the RS-232C connector, B end), then compares it with signal from the CTS input of port A (pin 18 on the SCC and pin 5 on the RS-232C connector, A end). This test requires a loopback cable between port A and port B.

RBb *Pass= Delay= Signal=*
The *RTS Channel B ==> CTS Channel B* command transmits a signal level to the RTS output of port B (pin 23 on the SCC and pin 4 on the RS-232C connector), then compares it with signal from its CTS input (pin 22 on the SCC and pin 5 on the RS-232C connector). This test requires a loopback connector on port B.

RBS *Pass= Delay= Signal=*

The *RTS Channel B Signal test* loops while transmitting a signal level to the RTS output of port B (pin 23 on the SCC and pin 4 on the RS-232C connector). This test does not require loopback.

All

The *All* command executes the following command sequence:

```
quick
```

This sequence is identical to the Quick Sequence.

Default

The *Default* command executes the following command sequence:

```
quick
```

This sequence is identical to the Quick Sequence.

Quick

The *Quick* command executes one of the following command sequences:

`"set Pass=1 ; daa ; raa"`	*loopback plug on A only*
`"set Pass=1 ; dbb ; rbb"`	*loopback plug on B only*
`"set Pass=1 ; daa ; dbb ; raa ; rbb"`	*loopback plug on A and B*
`"set Pass=1 ; dab ; dba ; rab ; rba"`	*loopback cable bw A and B*
`""`	*no loopback*

Register Tests Sub-menu

```
Sun CPU Diagnostic  Rev x.x  MM/DD/YY  SCC Register Menu

R2          Write then Read Register  2, Channel A
R12A        Write then Read Register 12, Channel A
R13A        Write then Read Register 13, Channel A
R15A        Write then Read Register 15, Channel A

R12B        Write then Read Register 12, Channel B
R13B        Write then Read Register 13, Channel B
R15B        Write then Read Register 15, Channel B

All         Execute all tests (100 times)
Default     Execute all tests ( 20 times)
Quick       Execute all tests (  5 times)

Command ==>
```

R2 *Pass= Delay=*

R12A *Pass= Delay=*

R13A *Pass= Delay=*

R15A *Pass= Delay=*

The *Channel A : Write then Read Registers 2, 12, 13, 15* commands write to a register (in port A) with all possible patterns, then read back (after delay) to verify. These tests work with or without loopback (either cable or connector).

Default :

```
Pass= 0

Delay= 100
```

R12A *Pass= Delay=*

R13A *Pass= Delay=*

R15A *Pass= Delay=*

The *Channel B : Write then Read Registers 12, 13, 15* commands write to a register (in port B) with all possible patterns, then read back (after delay) to verify. These tests work with or without loopback (either cable or connector).

Default :

```
Pass= 0

Delay= 100
```

All

The *All* command executes the following command sequence:

```
set Pass=100  ;  r2-r15b
```

Default

The *Default* command executes the following command sequence:

```
set Pass=20   ;  r2-r15b
```

Quick

The *Quick* command executes the following command sequence:

```
set Pass=5    ;  r2-r15b
```

5.12. Glossary

Exec

The SunDiagnostic Executive, the controlling program that starts all the diagnostics.

FPC

Floating-Point Coprocessor chip, Motorola MC68881.

SCC

Serial Communications Controller chip, AMD or Zilog 8530.

TOD

Time-Of-Day Clock chip, Intersil 7170.

6

The EEPROM Editing Tool

6

The EEPROM Editing Tool

6.1. Introduction

An EEPROM is an Electronically Erasable, Programmable Read-Only Memory chip. Sun uses this monolithic device for the non-volatile storage of data regarding the configuration of a workstation. It holds such information as the resolution of the console monitor screen, size of memory and where to look for bootable code upon power-up or system reset.

This tool allows you to edit certain fields of the EEPROM by making selections from menus. You don't have to know any memory locations in the EEPROM or any hexadecimal patterns.

6.2. Hardware Requirements

You need a Sun-3 workstation to use this tool, since Sun-2 models don't have an EEPROM. You need to know some things about the workstation, such as how much memory is on the CPU board, because that is the kind of information that goes into the EEPROM.

6.3. Hardware-Related Information

You will need to answer questions about the hardware configuration of the workstation. These questions include which printed circuit boards are installed in which slots, how much memory is on the boards, whether certain options are on the CPU board, the types of disk and tape equipment, and so on.

6.4. Loading And Starting The EEPTOOL

You need the workstation PROM Monitor prompt to begin, which looks like this:

```
>
```

If you have the operating system up and running, use the `/etc/halt` or `/etc/fasthalt` command to shut it down.

Refer to *Chapter 2* for information on bringing up the SunDiagnostic Executive Main Menu.

Select `Diagnostics` from the Main Menu and find the EEPROM Editing Tool in the list of available diagnostics.

6.5. The Main Menu

```
T       Primary Terminal type.
R       Monitor resolution.
S       Board Slots.
B       Boot paths and devices.
I       Initialize things.
Z       Reset EEPROM to all zeros.
SH      Show EEPROM fields.
W       Show all Write counts.
H       High-res monitor cols, rows.*
<esc>   Quit.
EEPROM says memory size = 4 MB.

*This choice appears on systems with a high resolution monitor.
```

This menu looks and behaves like a typical menu under the SunDiagnostic Executive.

6.6. Sub-Menus

Other menus come up as a result of selecting one of the items on the main menu.

Primary Terminal Type

The EEPROM has an area for storing what is the primary means of talking to the workstation user. For a typical workstation, it would be a black-and-white monitor and keyboard.

Here is the menu that comes up when you enter **T** from the main menu:

```
 Primary Terminal

M      B/W Monitor.
A      Serial Port A.
B      Serial Port B.
C      Color monitor.
P      P4 Frame Buffer

Default is M.

Choose one:
```

Monitor Resolution

Sun provides workstations with console monitors that have different resolutions. The typical resolution is 1152 x 900 pixels.

Here is the menu that comes up when you enter **R** from the main menu:

```
            Monitor Resolution

A  1152 x 900  Standard Resolution Display.
B  1024 x 1024 Display.
C  1600 x 1280 High Resolution Display *
D  1440 x 1440 Display.

Choose one:

* available on Sun-3/2xx and Sun-4/xxx only
```

The default EEPROM entry for monitor resolution is 1152 x 900, except for models and Sun-3/260 and Sun-3/280. For them, the default is 1600 x 1280.

Board Slot Data

One of the most important areas in the EEPROM is where it stores information about what type of board is installed in each slot of the workstation backplane. The Boot PROM looks at some of this, as does the operating system. Various application programs might also use this data.

When you select **S** for `Board Slots` from the main menu, a menu from which you may edit or display EEPROM information:

```
D      Display current slot information
C      Choose a slot to edit
```

If you selected `D` from this menu, a menu something like this is offered:

```
Slot 1: CPU 4 MB; WITH 68881; NO DCP; 0 kb cache.
Slot 2: Empty.
Slot 3: Memory Board 4 MB's on it.
Slot 4: Empty.
Slot 5: SCSI Sun 2; MT02 Tape ctlr; Adaptec Disk ctlr: 141 MB.
Slot 6: Empty.
Slot 6 is the last one.

Press Return to continue:
```

If you select `C` from the "Board Slots" menu, the slot assignments will be displayed as shown above, and, instead of the `Press Return` message at the bottom of the screen, you will be prompted for the slot you wish to change:

```
Slot to change (N if none)
```

Your EEPROM just might have some board data for a slot after the last one, since it can hold data for at least 13 slots. If so, the tool will display that information with this message:

```
(Beyond the "last" slot!)
```

You may leave the extra data alone or select that slot and tell the tool that the "slot," which presumably doesn't exist, is empty. The tool does not show empty slots after the last one.

In reply to the `slot to change` query, enter one of the following:

```
N       No board in this slot.
C       CPU Board.
M       Memory Board.
CO      Color Board.
FB      B/W Video Frame Buffer.
FPa     Floating Point Accelerator.
SMd     SMD Disk Controller.
T       Tape Controller.
E       Ethernet Controller.
A       MTI/ALM.
Gp      Graphics Processor.
SCp     SCP Controller.
Scsi    SCSI Host Adaptor.
I       IPC Board.
GB      Graphics Buffer.
SCM     3/75 SCSI.
MAp     MAPKIT Assembly.
END     Slot n is the last one.
```

The *n*, after `END` above, refers to the slot beyond the one this menu is for. The EEPROM does not have a way of storing the number of board slots in its workstation.

Many of the board types don't need any additional information, so the tool will show the updated board slot display and ask you to name a slot if you wish to continue with the board slot configuration.

Some board types, such as CPU and SCSI, will cause the tool to give you further menus or questions to provide more information concerning the board for the slot. If the tool can provide a default answer, it will do so.

When you have finished with that board, the tool will show the updated board slot display and ask you to name a slot if you wish to continue with the board slot configuration. If you do not, enter **n**.

Board Type Defaults

These are the defaults for different boards that the EEPROM tool knows about. Generally, the tool first looks to see if a reasonable choice is already in the EEPROM. If so, that is the default. If not, the default will be as shown here.

Board	*Default Configuration.*
CPU	Model 60, 260, 280: 8 Megabytes RAM, no 68881, no DCP, 64 KB cache. Other Models: 4 Megabytes RAM, no 68881, no DCP, 0 KB cache.
Memory	0 Megabytes.
Color	Type CG3.
SCSI	Type 2: 0 Tape controllers, 0 Disk controllers. If you have a tape controller, default is MT02. If you have a disk controller, default is Adaptec. Default Disk drive: 141 MB.
SMD Controller	Xylogics 451, Ctlr #0, 0 drives, Drive type 10.5" 575 MB.
Tape Controller	Xylogics 472, Ctlr #0, 0 drives.
ALM Serial Line Multiplexer	Systech, 16 lines.

Boot Paths And Devices

If you select **B**, Boot Paths and Devices, from the main menu, this menu is offered:

```
U       Default UNIX Boot device (poll or EEPROM.)

E       EEPROM Unix boot device.

D       Diagnostic boot device and path.
```

The default operating system boot device selection gives you a chance to tell the Boot PROM to either use a list of its own to search for a bootable device, or to find, in the EEPROM, your choice of which device to boot.

For a default operating system and Diagnostic boot device, the tool assigns the device letters `le` to models Sun-3/50 and Sun-3/60, and will the device letters `ie` to all other models.

It assigns zeroes for the control, unit and partition fields of the boot devices.

The tool does not assign a default for the diagnostic boot path.

EEPROM Operating System Boot Device

If you chose EEPROM Boot Device from the Boot menu, the tool displays this sub-menu:

```
L     Change the Unix Device letters.
C     Change the Unix Controller number.
U     Change the Unix Unit number.
P     Change the Unix Partition number.
S     Show current Unix boot device.

ie (0,0,0)
device (controller, unit, partition)
```

Instead of `ie`, your display may show another two device letters and your display may show other numbers in the parentheses. If the EEPROM did not have two letters stored in the proper place, the tool will put a default pair of letters there before displaying this menu. The numbers are, respectively, the boot device controller, unit and partition.

If you choose **L** in this menu, the tool will present this sub-menu:

```
Le  le   (Set boot device to Lance net)
Ie  ie   (Set boot device to Ethernet)
Sd  sd   (Set boot device to SCSI disk).
XD  xd   (Set boot device to Xylogics 7053).
Xy  xy   (Set boot device to Xylogics 450/451).
ST  st   (Set boot device to SCSI Tape).
Mt  mt   (Set boot device to Tapemaster tape).
XT  xt   (Set boot device to Xylogics tape).
```

Choose the boot device you want stored in the EEPROM, for a normal boot, when you choose to boot from that specific device rather than to poll for a device.

If you choose one of the other items from the EEPROM Boot Device menu, such as `Change the Unix Unit number`, the tool will prompt you for a number.

Diagnostic Boot Device

If you chose EEPROM Diagnostic boot device from the Boot menu, the tool displays this menu:

```
L       Change the Diagnostic boot Device letters.
C       Change the Diagnostic boot device Controller number.
U       Change the Diagnostic boot device Unit number.
P       Change the Diagnostic boot device Partition number.
S       Show current Diagnostic boot device and path.

ie (0,0,0) path
device (controller, unit, partition) path
```

Instead of `ie`, your display may show another two device letters and your display may show other numbers in the parentheses. If the EEPROM did not have two letters stored in the proper place, the tool will put a default pair of letters there before displaying this menu. The numbers are, respectively, the boot

device controller, unit and partition.

Your display may have nothing in the *path* area.

If you choose **L** from this menu, the tool will present the tool will present a sub-menu of the available boot devices as shown on the previous page.

Choose the diagnostic boot device you want in the EEPROM, for the case when you boot the workstation with its Diagnostic switch turned on.

If you choose one of the other items from the EEPROM Diagnostic boot device menu, such as `Change the Diagnostic Boot Device Unit number`, the tool will prompt you for a number.

High Resolution Monitor Columns And Rows

If your workstation model has a console monitor resolution of 1600 by 1280 pixels, you should use the main menu **H** command to tell the EEPROM how many columns and how many rows of characters it supports. The defaults are 80 columns and 34 rows.

Initialization

If, from the Main Menu, you select to initialize things, the tool displays this menu:

```
A       Initialize everything.
T       Initialize primary Terminal type.
R       Initialize Monitor resolution.
S       Initialize Board Slots.
B       Boot paths and devices.
H       High-res monitor cols, rows.
P       Initialize the Test Pattern.
Z       Make sure every field except these are zero.
```

Here are its assumptions:

Primary Terminal Type: B/W Monitor and keyboard.

Monitor resolution: 1152 x 900.

(Except models 260 and 280, which it gives a resolution of 1600 x 1280.)

Board slots:

1 CPU with 4* MB RAM, no 68881, no DCP, 0 KB cache.
2 Empty.
3 END (2 is last slot).†
4-12 Empty.

*Models 60, 260 and 280 get 8 MB RAM.

†Applies to Models 50, 60 and 75.
Empty on other models.

Please note that, after this initialization, if you wish to modify the board slot information to describe a larger number of slots, first request to change slot 3. Change it so that it no longer indicates that slot 2 is the last one.

6.7. EEPROM Reset

You can elect to clear the entire EEPROM to all zeros by selecting **Z**, for `Reset`, from the Main Menu. This does not, however, clear the fields that hold the EEPROM write count.

6.8. Show EEPROM Fields

The **SH** item on the Main Menu allows you to see the data now in the EEPROM, displayed in a readable, interpreted format, instead of hex numbers (as it is actually stored).

6.9. Show All Write Counts

Selecting **W** from the Main Menu allows you to see the number of times that the EEPROM was written to, either by this program or any other program that increments the write-count fields. There are four different counters, depending on which part of the EEPROM was written to. The busiest part of the EEPROM is the "Diagnostic" area. The others are called "Reserved," "ROM," and "Software."

6.10. Recommended Procedure

Select `Initialize Things` from the main menu. From the initialization menu, select `Everything`. Normally, that's all you have to do. If your workstation has differences, choose the other menu items, as required, and enter the information relevant to your machine.

7

Sun Ethernet Diagnostic

7

Sun Ethernet Diagnostic

7.1. General Description

The Ethernet Interface provided on Sun-3 CPU Boards connects Sun-3 workstations to an Ethernet communication network. From a test viewpoint, the interface can be subdivided into three major sections: the Ethernet chip set, the control interface, and the memory path. The Sun-3 Ethernet Diagnostic tests the Ethernet Interface on Sun-3 CPU Boards.

7.2. Hardware Requirements

The minimum hardware configuration required is:

1. A Sun-3 cardcage.
2. A Sun-3 power supply.
3. A Sun-3 CPU board.
4. A Sun-3 Memory board.
5. A dumb terminal(TeleVideo, Wyse, etc.) attached to a CPU serial port.
6. A boot device; local disk, local tape or remote disk (over Ethernet.)
7. An Ethernet transceiver(3COM 3C100 or equivalent) with two terminator assemblies.
8. A standard Ethernet transceiver cable connected between Ethernet port on the CPU board and the transceiver box (for external loopback tests.) The cable can be ordered from Sun. It is P/N 530-1241, "Assy., Cable transceiver 15 meter".

Test Overview

The tests are grouped into separate menus to facilitate testing and failure isolation within a subdivision of the Ethernet Interface. The groupings are not entirely independent, however. There is some unavoidable overlap between tests. Here are a few things to keep in mind:

1. Some of the tests only work on a particular chip set or board implementation.
2. Although most tests will be applicable to many different chip sets or board implementations, the actual test implementation details and error message content may differ for different chip sets.

Each test description states:

1. The purpose of the test.
2. The parameters required.
3. The function of the test.

Aborting an Ethernet Test

Most tests display the message

```
enter one character of ! to escape
```

when they begin. This message is intended to show you which character to enter in order to abort a test BEFORE the number of passes specified by the `PASs=` parameter has been reached. Using the exclamation point will not abort a test in the middle of a pass, nor will it abort a sequence of tests.

7.3. The Main Menu

The user interface of the Ethernet diagnostic consists of a Main Menu and four sub-menus. Help options on each menu offer more detailed instructions to the user. All sub-menus allow the user to return to the Main Menu by using the escape character.

NOTE *Do not use a question mark to display a help menu; use*

```
help command_you_want_help_with
```

for help messages.

```
Sun-3 Ethernet Diagnostic  REV x,x xx/xx/xx  Main Menu

i  -  control Interface menu
n  -  etherNet menu
m  -  Memory path menu
A  -  debugging Aids menu
ALl - All test sequence
DEfault - Default test sequence
QUick - Quick test sequence

message line

message line


Command==>
```

i

If the *control interface* menu is selected, the diagnostic displays a sub-menu containing the control interface tests.

n

If the *Ethernet* menu is selected, the diagnostic displays a sub-menu containing the Ethernet chip tests.

m

If the *memory path* menu is selected, the diagnostic displays a sub-menu containing the memory path tests.

A

If the *debugging aids* menu is selected, the diagnostic displays a sub-menu containing the debugging aids.

ALl

The *All tests* command runs all of the tests available on this menu. It executes the following commands in sequence:

```
i ; AL ; n ; AL ; m ; AL ; u ; AL
```

The summary after running ALL tests from the main menu ends after three lines, which isn't enough to show results from all the tests that were executed. If an error occurs near the end of the ALL test sequence, you will be unable to see the error unless you are carefully watching the display during execution.

DEfault

The *Default tests* command runs a sub-set of the tests available on this menu. It executes the following commands in sequence:

```
i ; DE ; n ; DE ; m ; DE
```

QUick

The *Quick tests* command runs a small set of tests that complete in a short time. It executes the following commands in sequence:

```
i ; QU ; n ; QU
```

7.4. The Control Interface Menu

The Control Interface tests test the Sun hardware that interfaces with the Ethernet chip set, excluding hardware specific to the memory path. This hardware includes external control and status registers, interrupt hardware, parity, and protection circuits.

```
Sun-3 Ethernet Diagnostic  REVx.x xx/xx/xx  Control Interface Menu

c  -  Control & status register test
s  -  System error tests
i  -  Interrupt enable/disable test
ALl - All test sequence
DEfault - Default test sequence
QUick - Quick test sequence


message line

message line

Command==>
```

c

The *Control and Status Register* test checks the function of the control and status registers. It attempts to set and clear register bits to verify that register bits do get set and cleared only when they are supposed to.

Use `PAss=)` to set the number of passes. Default number of passes is one.

s

The *System error* test checks the function of the Ethernet specific error reporting circuitry. It creates system errors and protection violation conditions and verifies proper error and violation reporting.

Use `PAss=)` to set the number of passes. Default number of passes is one.

i

The *Interrupt Enable and Disable* test checks the Ethernet interrupt circuitry. It creates interrupt conditions and verifies that interrupts occur when they are enabled and do not occur when they are disabled.

Its parameter is number of passes (`PAss=`). The default number of passes is one.

ALl

The *All tests* command runs all of the tests available on this menu. It executes the following commands in sequence:

```
c ; s ; i
```

DEfault

The *Default tests* command runs a subset of the tests available on this menu. It executes the following commands in sequence:

```
c ; s ; i
```

QUick

The *Quick tests* command runs a small set of tests that complete in a short time. It executes the following commands in sequence:

```
c ; s ; i
```

7.5. The Ethernet Menu

The Ethernet tests check the functionality of the Ethernet chip set.

```
Sun-3 Ethernet Diagnostic  REVx.x xx/xx/xx Ethernet Menu


i  -  Initialization test
A  -  diAgnose test
n  -  Nop test
b  -  internal loopBack test
c  -  enCoder loopback test
f  -  external loopback test
ALl - All test sequence
DEfault - Default test sequence
QUick - Quick test sequence


message line

message line

Command==>
```

i

The *Initialization* test checks the operation of Ethernet chip set when it is subjected to initialization procedures. It initializes the Ethernet control blocks, resets the Ethernet chip set, issues channel attention (for Intel chips), and verifies the chips' correct response to initialization.

Use `PAss=` to set the number of passes. The default number of passes is one.

A

The *Diagnose (Intel only)* test runs the Intel chip's self-diagnosis command and checks the results.

Use `PAss=` to set the number of passes. The default number of passes is one.

n

The *Nop (Intel only)* test executes the Intel chip's NOP command and checks the results.

Use `PAss=` to set the number of passes. The default number of passes is one.

b

The *Internal Loopback* test checks the performance of the chip set's buffer management, address recognition, CRC generation and detection, interrupt,

and retransmit circuits. It configures the chip set to internal loopback mode, transmits a block, receives the same block, analyzes the control blocks to see if they are in the expected state, then compares the transmitted and the received data.

The parameters are:

PASs= FIFOlim= BLOCKsize=.

The defaults are:`number of passes=1 fifolim=8 blocksize=0x2000`

FIFOlim sets the length of the FIFO buffer inside the Ethernet chip. Acceptable entries are 1 through 15.

BLOCKsize sets the size of the buffers used by the Ethernet chip.

NOTE *Do not type @ during this test; doing so leaves the Ethernet chip locked in loopback.*

c

The *Encoder Loopback (Intel only)* test checks the Intel Ethernet chip set by using the Serial Interface Adapter without going to the transceiver. It sets the Serial Adapter Chip to loopback mode, then runs the sequence of tests described for internal loopback.

The parameters are *PASs= FIFOlim= BLOCKsize=*.

The defaults are number of passes = 1, fifolim = 8 and blocksize = 18.

FIFOlim sets the length of the FIFO buffer inside the Ethernet chip. Acceptable entries are 1 through 15.

BLOCKsize sets the size of the buffers used by the Ethernet chip.

f The *External Loopback:* test checks the Serial Interface Adapter and transceiver interface. It sets the Serial Adapter Chip to full external loopback mode then runs the sequence of tests described for internal loopback.

The parameters are *PASs= FIFOlim= BLOCKsize=*. The defaults are number of passes = 1, fifolim = 8 and blocksize = 18.

FIFOlim sets the length of the FIFO buffer inside the Ethernet chip. Acceptable entries are 1 through 15.

BLOCKsize sets the size of the buffers used by the Ethernet chip.

After the Ethernet Diagnostic has been invoked, in order to run the External Loopback Test correctly, the following requirements must be fulfilled: You must connect a "null network" to the CPU board Ethernet connector in place of the normal Ethernet cable. This "null network" must consist of:

- Items 7 and 8 listed under *Hardware Requirements* in this document
- a transceiver cable
- a transceiver with two terminating connectors

If your system boots with the

```
le
```

boot device command, the message

```
ext_chaste: **heartbeat
```
or
```
ext_chaste: **no heartbeat
```

may be printed out. *This is not an error or warning message and does not indicate that the has Ethernet circuitry has failed.*

ALl

The *All tests* command runs all of the tests available on this menu. It executes the following commands in sequence:

```
i ; A ; n ; b ; c ; f ; t
```

DEFault

The *Default tests* command runs a subset of the tests available on this menu. It executes the following commands in sequence:

```
i ; A ; n ; b ; t
```

Quick

The *Quick tests* commands runs a small set of tests that complete in a short time. It executes the following commands in sequence: `i ; A`

7.6. The Memory Path Menu

The Memory Path tests are designed to test the functionality of Sun hardware that provides memory access to the Ethernet chip set. These tests check circuits such as address or data latches that are missed by the other diagnostics. These tests don't test memory functionality; that's done more efficiently by other diagnostics.

```
Sun-3 Ethernet Diagnostic REVx.x xx/xx/xx Memory Path Menu

A  -  Address test
b  -  data test
c  -  address/data independence test
ALl - All test sequence
DEfault - Default test sequence
QUick - Quick test sequence

message line

message line

Command==>
```

A *PASs= BEG= END=*

The purpose of the *Address* test is to check the Ethernet chip set's ability to access memory within its address range. This test accepts these parameters:

PASs=*number of passes*

BEG=*beginning address*

END=*ending address*

The default number of passes is one.

The function of the address test is to create data buffers throughout the Ethernet chip set's memory access range (if possible) and verify that data can be written and read properly.

b *PAss=*

The purpose of the *Data* test is to check the Ethernet chip's ability to write and read data in memory.

The test accepts the *pass=* parameter. The default number of passes is one.

The test writes and reads different data patterns in memory, using the Ethernet chip set, then checks the data's accuracy and lack of pattern sensitivity.

c Address/Data Independence

The purpose of this test is to check the Ethernet chip set's ability to write and read data in memory regardless of the memory address accessed. (Note: both chip sets have multiplexed address and data lines)

This test accepts the parameters

PASs= BEG= END= .

The default settings for these parameters are:

```
1 0x0FF00000 0xFF7800    for Sun-2
```

and

```
1 0x0FF00000 0x0FFD6000  for Sun-3
```

The function of this test is to write and read memory using data patterns different from the addresses accessed, then verify the data.

ALl

The *All tests* command runs all of the tests available on this menu. It executes the following commands in sequence:

```
A ; b ; c
```

DEfault

The *Default tests* command runs a subset of the tests available on this menu. It executes the following commands in sequence:

```
A ; b ; c
```

QUick

The *Quick tests* commands runs a small set of tests that complete in a short time. It executes the following commands in sequence:

```
A ; b ; c
```

7.7. The Debugging Aids Menu

The Debugging Aids are designed to help you isolate hardware problems. These commands provide useful information and line control, but cannot be considered automatic tests. The Debugging Aids Menu has four options.

```
Sun-3 Ethernet Diagnostic  REVx dd/mm/yy  Debugging Aids Menu

r  -  Reset toggle
c  -  Channel attention toggle
D  -  Dump control blocks
s  -  display data Structure addresses

message line

message line

Command==>
```

r

The *Reset Toggle* test toggles the Ethernet chip's set/reset line so it can be checked with an oscilloscope.

The parameter is: *PASs=*, and the default is one.

c

The *Channel Attention (Intel only)* test toggles the Ethernet chip's channel attention line so that it can be checked with an oscilloscope.

The parameters are *PASs= MOde=*. The default settings are:`1 0`.

If the *mode* parameter equals 1, the test executes a very tight loop, displaying no messages.

If the *mode* parameter equals 0, all messages are printed.

D

Dump Control Blocks

This test displays for analysis the contents of the control and status registers, the control blocks, and any buffers.

The parameters are *LEVEL= INDex=*. The defaults are `-1 -1`.

The test displays the specified control blocks on the screen. The flag bits described below allow you to control the contents of the display (the default `level` value is `ALL = -1`. The second parameter, *index*, optionally specifies a particular block, descriptor, or buffer to be dumped. *index* is a 4 bit field, that is set by giving it an integer from 1 to 16; the default value is -1 (all bits on, which means "display everything").

Table 7-1 *Intel Ethernet Chip Status Levels*

Intel Levels	
0	control & status register
1	system configuration pointer
2	intermediate system configuration pointer
3	system control block
4	command block(s)
5	transmit buffer descriptor(s)
6	transmit buffer(s)
7	receive frame descriptor(s)
8	receive buffer descriptor(s)
9	receive buffer(s)
a	system control block statistics
b	tdr status

Table 7-2 *AMD Ethernet Chip Status Levels*

AMD Levels	
0	control & status registers
3	initialization block
5	transmit descriptor(s)
6	transmit buffer(s)
8	receive descriptor(s)
9	receive buffer(s)

s

Display Data Structure Addresses

This selection displays data structure addresses to assist debugging done with logic analyzers. This command indicates what locations the Ethernet chip set is attempting to access.

The parameters are: *LEVEL= INDex=*

The default settings are: `-1 -1`.

An **s** selection displays the specified data structure addresses. Parameter definitions and defaults are the same as in the `Dump` command above.

8

Sun-3 FPA Diagnostic

8

Sun-3 FPA Diagnostic

8.1. Required Hardware

The FPA board is only used with certain Sun-3 systems. You need the following hardware to run the diagnostic successfully:

- A working Sun-3 CPU board
- A Floating Point Accelerator board (to be tested)
- A working Sun-3 monitor
- A working Sun-3 keyboard
- A working boot device (disk, tape, or Ethernet)

8.2. Tests

The tests in the diagnostic cover about 98% of the FPA board. The tests can be divided into 4 functional groups:

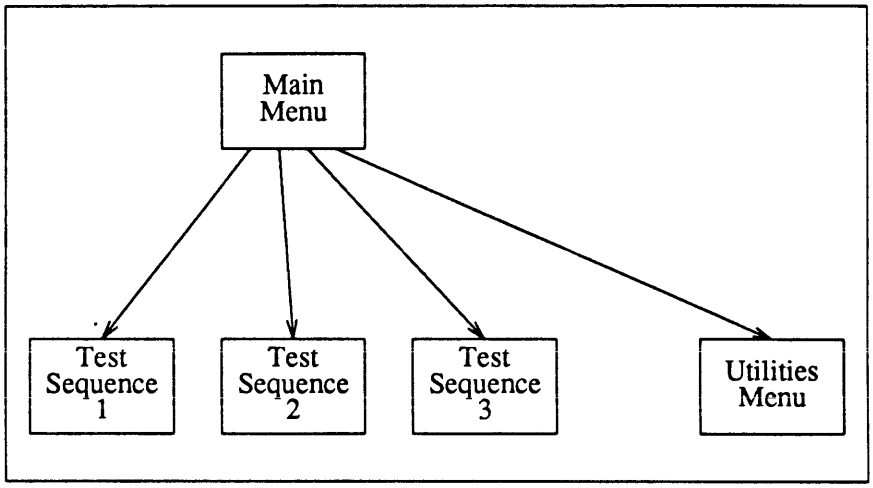
- register tests
- Weitek tests
- pipeline tests
- microcode controller tests

There are a number of tests for each functional section. The tests in the diagnostic are divided by coverage, not functionality. The tests reside in three menus; Test Sequence 1, Test Sequence 2, and Test Sequence 3. Each test sequence relies on its predecessors and covers the same area more thoroughly. Simpler tests that exercise less circuitry are run first.

In addition to the three test menus, there is a utility menu that is used to directly access some parts of the FPA board for troubleshooting purposes.

The following figure diagrams the menus in the diagnostic and their relationship to each other:

Figure 8-1 *The FPA Diagnostic Menu Hierarchy*



Test Syntax

Use the command syntax described in *Chapter 2* in order to run these tests. You need enter only the letters shown in upper case in order to invoke tests and parameters. When an "equals" sign is shown, you must enter it also.

Default Parameters

Commands in the diagnostic that accept parameters also have a set of default parameters that can be invoked.

Three special characters are used to invoke a default parameter for a given command. Instead of entering a value for a parameter, enter one of the following:

- null (leave out the parameter value) " "
- period (a single dot) " . "
- asterisk (a single star) "* "

Each command parameter has two default values. Entering null or period invokes the command's first default value. Using an asterisk invokes the other default value, which is usually the largest legal value. The particular value invoked by a default character varies with each command and parameter.

Batching Commands

A series of commands can be "batched", or automatically run in sequence, by entering a series of them on the command line, each one separated by semicolons (;). Each command in the series is entered with its proper parameters or defaults. The commands are run in the order they were entered on the command line. Commands that reside on different menus can be batched together by including the sub-menu and u options as part of the command sequence. Refer to *Chapter 2* for more information on command line syntax.

Test Menus

The tests and commands in the diagnostic are organized into a hierarchy of menus and sub-menus. In addition to the tests themselves, there are commands that move you up or down in the menu tree.

The following commands are common to all of the diagnostic's test menus:

`All`
Run all tests. This command runs all of the tests in sequence in the current menu, and any menus below it. The tests are run in the order they appear in the menu, with their default parameters.

`?` Display the help menu. This command displays the current help menu. This menu shows the commands of the current menu, along with the names of the parameters expected by each command. This menu is useful if you forget what a given command's parameters are. For a more detailed description of a particular test, refer to it in this manual.

8.3. Main Menu

When you boot up the diagnostic, it displays the main menu. This menu is at the top of the command menu hierarchy. The `b`, `c`, `d`, and `u` commands call sub-menus.

```
Sun 3 FPA Diagnostic  x.x  xx/xx/xx  FPA Main Menu

   All             All test Sequence
   Default         Default Test Sequence
   Options         Display the Local Options Menu
   1               Test Menu #1
   2               Test Menu #2
   3               Test Menu #3
   Utilities       Utilities Menu
   ?               Display Help Menu

Command ==>

FPA COMMAND:
```

NOTE *Do not use the* `Utilities`*Menu selection; it is not functional at this time.*

`All`

The *all tests* command executes every test in the diagnostic in sequence; running the test sequences in order, going from top to bottom, in each sequence menu. The commands executed by the all tests command are as follows:

- Test Sequence 1

  ```
  C ; R ; N ; M ; L ; I ; T ;
  ```

- Test Sequence 2

  ```
  P ; P1 ; I ; Ptr ; P5 ; L ; LF ;
  ```

- Test Sequence 3

  ```
  DX1 ; R ; S ; RE ; M ; D ; WO ; WS ; J ; P
  ```

`Default` *Pass=*

The *default test* command runs a subset of the diagnostic's tests, providing reasonable coverage of the FPA circuitry.

`Options`

The *options* command displays the local options menu, which allows you to modify the default options of the diagnostic.

`1`

The *test sequence 1 menu* command displays the test sequence 1 sub-menu. This command must be executed before running test sequence 1 commands.

`2`

The *test sequence 2 menu* command displays the test sequence 2 sub-menu. This command must be executed before running test sequence 2 commands.

3

The *test sequence 3 menu* command displays the test sequence 3 sub-menu. This command must be executed before running test sequence 3 commands.

Utilities

This command is not functional at this time.

?

The *help* command displays the current menu with the addition of the expected command parameters.

Test Sequence 1 Menu

The tests in this sequence provide first pass coverage of FPA functionality. Sequences 2 and 3 assume that the functionality tested in sequence 1 is working. Each test in the menu depends on the success of the previous test in order to provide accurate error messages.

```
Sun 3 FPA Diagnostic  1.0  mm/dd/87  FPA Test #1 Menu

   All              Execute All tests Sequence
   Default          Execute Default Test Sequence
   Options          Display the Local Options Menu
   Configure        FPA Configuration Test
   Register         Register Test(s)
   Nack             Nack (Negative Acknowledge) Test
   Map              Mapping Ram Test(s)
   MIcrostore       MicroStore Ram Test(s)
   Loop             Loop Counter Test
   Instruction      Execute Simple Instruction Test
   Timeout          Timeout Retry Test
   ?                Display Help Menu


Command ==>
```

All *Pass=*

The *all tests* command runs all of the tests in the current menu in sequence the number of times specified after *Pass=*

Default

The *default tests* command runs a default subset of the FPA diagnostic tests.

Options

The *options* command brings up a menu that provides for modification of the diagnostic's default options.

Configure *Pass=*

The *FPA Configuration Test* test reads the version number of the FPA board. This insures that the correct version of the microcode is loaded into the FPA hardware. The test has one argument, *Pass=* which controls how many times the test is run.

Register *Index= Pass=*

The *register* test writes and reads a selected register the number of times specified by *Pass=*. After each write/read cycle, the results are checked to

make sure the register is holding values properly. The parameter *Index* selects the register to check. It can have one of 4 values:

- 0 - all registers
- 1 - Immediate error (IERR) register
- 2 - State register
- 3 - Imask register
- 4 - Load Pointer (Ldptr)

The second parameter, *Pass=* controls the number of times the test is executed. The default value for the number of passes is 1.

Nack *Pass=*

The *negative acknowledgement test* deliberately creates errors to check the IERR (Immediate ERRor) register. The test writes to read-only locations, and reads from write-only locations to generate bus errors. These bus errors should set a bit pattern in the IERR register. If they don't, the test displays an error message. The test has one parameter, *Pass=*, which controls the number of times the test is executed. The default value for the number of passes is 1.

Map *Index= Pass=*

The *mapping RAM test* accepts two arguments: *Index* and *Pass=*. The *Index=* argument selects between the fast RAM test, the medium RAM test, or the burn-in RAM test. Each test provides more thorough coverage than the previous one. The values for *Index=* are shown in the following table:

Table 8-1 *Index Values*

Index	*test to run*
0	all tests
1	fast RAM test
2	medium RAM test
3	burn-in RAM test

Every test writes patterns of data into the mapping ram and reads it back, comparing the result with the original data.

The `fast ram` test writes and reads the first 10 locations in the mapping ram. It uses four hexadecimal data patterns; 0000, FFFF, 5A5A, and A5a5.

The `medium ram` test runs five different subtests; a data bus check, three address bus checks, and a rotating pattern check.

The data bus check writes every location in RAM with a pattern, then reads it back. If the pattern is corrupted, the test returns an error message. This check uses the following hexadecimal data values for the pattern:

Patterns	
(in hex)	
0000	1818
FFFF	E7E7
5555	7171
AAAA	8E8E
6666	C3C3
9999	3C3C

The first address bus check writes a unique value into every memory location, then reads it back, checking for errors. If an error is found, the test prints an error message.

The second address bus check writes a *walking pattern* into memory. The test sets the RAM to all zeros, then writes all ones into every other memory location. After checking the ram, it repeats the process, this time writing zeros into every location on a background of ones.

The third address bus check clears the RAM then writes into one location repeatedly. The test then examines the memory, to see if any other locations were disturbed by the multiple writes. Once finished, the test repeats at a new location. The locations written to are shown, in order, in the following table:

Write Locations	
(addresses in hex)	
0000	0181
0FFF	0E7E
0555	0717
0AAA	08E8
0666	0C3C
0999	03C3

The rotating pattern check writes a pattern value into every location into memory, then reads back the values to make sure they haven't been corrupted. Then the test does a single left-rotate one the pattern and repeats the process. After two passes, the test uses a new pattern. The patterns used are listed, in order, in the following table:

Rotating Patterns
(values in hex)
0000
1111
FFFF
5555
AAAA

The burn-in RAM test writes zeros to the entire RAM space. Then it writes a single location with all ones and the entire RAM is read to see if any values changed as a result. This process is repeated for every location in the RAM. When its finished, the burn-in test repeats, this time writing a location of zeros on a background of ones.

MIcrostore *Index Pass=*

The *microstore RAM test* runs the same test sequence used for the mapping ram test, above. Use the mapping ram test description for the microstore ram test. The microstore RAM is accessed through the LD_PTR register on the FPA board.

Loop *Pass=*

The *loop* command controls the number of times a command sequence is executed. The loop command is the last command in the sequence that is repeated. The *Pass=* argument controls the number of times the sequence is executed.

Loop *Pass=*

The *loop counter test* checks the loop counter register and the loop counter jump instructions. The test loads the loop counter with different values, then runs a microcode program that loops until the loop counter reaches zero, then jumps to a specified location. The test examines the values in the loop counter register while it is counting down, to make sure it is working correctly. The test cannot distinguish between a loop counter failure and a problem with the conditional jump instruction.

The test accepts one parameter, *pass=*, which controls the number of times the test is run.

Instruction *Pass=*

The *simple instruction* test makes sure that the FPA board can execute simple microcode instructions. The test writes single microcode instructions into program memory, executes them, then checks the status register for error conditions. The test uses NOP instructions for the test; the single precision NOP, the double precision NOP, and the single precision, unimplemented NOP instruction are executed.

The test accepts one argument, *Pass=* which controls the number of times the test is run.

Timeout *Pass=*

The *timeout/retry test* tests the FPA board's timeout circuitry using a microcode program. The test runs a set of microcode instructions, one after another, in an infinite loop program, to test the timeout circuitry. The instructions tested are; pipe write, pipe read, context write, context read, and shadow read. The test checks to see if a "256 tries" error occurs after the program is started. The test accepts one argument, *Pass=*, which controls the number of times the test is run.

?

The *help* command displays the current menu with the addition of the expected command parameters.

Test Sequence 2 Menu

These tests provide more thorough coverage of the FPA, with each test exercising more circuitry. They assume that all of the tests in `Sequence 1` have passed without errors. Each test in this menu assumes that the FPA board has passed all the tests above it.

```
Sun 3 FPA Diagnostic  1.0  10/6/86  FPA Test #2 Menu

All                 Execute All tests Sequence
Default             Execute Default Test Sequence
Options             Display the Local Options Menu
Pipe                Pipe Test
P1                  Pointers 1 through 4 Test
Immed               Immed(2, 3) Pointer Test
Ptr                 Pointer Increment/Decrement Test
P5                  Pointer 5 Test
Lock                Lock Test
LF                  L+/F+ Test
?                   Display Help Menu

Command ==>
```

All *Pass=*

The *all tests* command runs all of the tests in the current menu in sequence the number of times specified with *Pass=*.

Default

The *default tests* command runs a default subset of the tests in this menu.

Options

The *options* command brings up a menu that allows the user to modify the diagnostic's default options.

Pipe *Pass=*

The *pipe test* checks the instruction and data pipeline hardware. The test is composed of three parts that test the pipeline control circuitry, the data pipeline, and the instruction pipeline. The test sends instructions and data through both pipelines, then reads the pipes to make sure the data, instructions, and the pipe status information is correct. The test accepts one argument, *Pass=*, which controls the number of times the test is run.

P1 *Pass=*

The *pointers (1-4) test* checks the load pointer and pointers 1 through 4. The pointer registers are tested to insure they can hold legal values, and that they can be used in instructions to address RAM on the FPA board. Each register is successively loaded with an address value, then read and compared against the original value. If the values don't match, the test displays an error message. The test accepts one argument, *Pass=*, which controls the number of times the test is run.

Immed *Pass=*

The *immediate(2,3) test* is very similar to the pointer (1-4) test, except that the pointer registers must be loaded in a different manner. Since the `immed2` and `immed3` pointer registers get their values directly from the floating point instructions sent to the FPA board, the test sends instructions that use the pointers to the board, then reads the values of the pointers. If the pointer values are incorrect, the test displays an error message. The test accepts one argument, *Pass=*, which controls the number of times the test is run.

Ptr *Pass=*

The *pointer increment/decrement test* tests the ability of the pointer registers to increment and decrement their values. The test exercises all of the pointer registers on the FPA board. The test programs the board with a sequence of microcode instructions which increment a pointer register through its range of values, then decrement it back down to zero. After each increment or decrement step, the test checks the value of the register being tested. If the value in the pointer register is incorrect, the test displays an error message. Each pointer register on the FPA board is tested in turn. The test accepts one argument, *Pass=*, which controls the number of times the test is run.

P5 *Pass=*

The *pointer five test* insures that `pointer five` can be loaded with, as well as increment and decrement through, all of its legal values. The test first programs the FPA board with a series of instructions that load the `pointer five register` with all of its legal values, one at a time. The FPA board then runs the program. After each new value is loaded, the program stops, and the test checks the pointer value to insure its correct. After this step is completed, the test reloads the FPA board with a new program, which increments, then decrements the `pointer five` register through its range of legal values. The program is run as before, and the test checks the register value each time it is incremented or decremented. If the test discovers that the register cannot hold a loaded value, or cannot increment or decrement properly, it displays an error message. The test accepts one argument, `Pass=`, which controls the number of times the test is run.

Lock *Pass=*

The *lock test* checks the interlock capability of the `shadow` registers on the FPA board. Each class of instruction sent to the FPA interlocks different `shadow` registers. The test sends a representative instruction from each class to the FPA board, then checks to make sure the proper registers have interlocked. If they didn't, the test displays an error message.

LF *Pass=*

The *L+ F+ test* checks the L+ and F+ circuitry on the FPA board. L+ and F+ refer to bit fields that control the Weitek Chip set. The F+ field determines what operation the Weiteks perform. The L+ field specifies what part of an operand is being loaded to the chips. These fields are generated by the Mapping RAM (decoded directly from a microcode instruction) or by the micromachine itself. The L+ F+ test is composed of three subtests; the mapping RAM test, the microstore read test, and the microstore generation test.

The mapping RAM test

This test makes sure that L+ and F+ fields generated by the mapping ram can be loaded into the Weitek chips. The microstore is loaded with instructions that select the L+ and F+ fields from the mapping ram. The mapping ram is loaded with every possible combination of L+ and F+ values. The test then executes a microstore instruction, and then checks the Weitek chips to see if the proper L+ and F+ values were sent. The test repeats until every value in the mapping ram has been sent to the Weiteks.

The microstore read test

This test ensures that the data path between the microstore and the Weitek chips work. The microstore is filled with instructions containing all possible L+ and F+ values. The test then reads a location in the microstore and sends the values to the weitek chips. It then checks the chip registers to make sure the proper values were received. This cycle is repeated until every location in microstore is read.

The microstore generation test

In this test, the micromachine itself sends F+ codes to the Weitek chips. The microstore is filled with instructions that send F+ values. The instructions are then executed, one at a time. After each instruction, the test reads the weitek registers to ensure the proper value was received. This cycle is repeated until all possible F+ values have been sent, and every location in microstore has been executed.

?

The help command displays the current menu with the addition of the expected command parameters.

Test Sequence 3 Menu

This sequence contains the most complex and comprehensive tests of the three sequences. All of the tests in this sequence assume the FPA board has passed all of the tests in the previous two sequences. Every test in this menu assumes the FPA board has passed all of the tests above it.

```
Sun 3 FPA Diagnostic  X.X  mm/dd/87  FPA Test #3 Menu

   All          Execute All tests Sequence
   Default      Execute Default Test Sequence
   Options      Display the Local Options Menu
   DX1          Dx1/Dx2 Operand Data Path
   Ram          Register Ram Test
   Shadow       Shadow Ram Test
   REgister     Status Register Test
   Mode         Mode Register Test
   Data         Weitek Data Path Test
   WOp          Weitek Overall Operation Test
   WStatus      Weitek Overall Status Test
   Jump         Jump Conditions Test
   Pipeline     Pipline Control(Timing) Test
   ?                 Display Help Menu


Command ==>
```

All *Pass=*

The *all tests* command runs all of the tests in the current menu in sequence *Pass=* times.

Default

The *default tests* command runs a default subset of the tests in this menu.

Options

The *options* command brings up a menu that allows you to modify the diagnostic's default options.

DX1 *Pass=*

The *Dx1/Dx2 Operand data path test* checks the op.d and rr.d data buses on the FPA board. These buses transfer operand values of floating point instructions between different registers on the board. The test uses special diagnostic microcode instructions to write values to the data buses, then read the bus latchs to make sure they have the correct value. The test uses a "walking ones" pattern to generate the data it sends over the bus. If the data read off the bus latch does not match the original data written to the bus, the test displays an error message. The test accepts one argument, *Pass=*, which controls the number of times the test is executed.

Ram *Index Pass=*

The *register ram test* is very similar to the `mapping` RAM test. The only difference is that it accesses the RAM indirectly, by using the load pointer register. The test writes various values into memory, then reads them back. If the value read does not match the value written, the test displays an error message. The test accepts two arguments, *Index* and *Pass=*. The *Index*

argument controls what level RAM test is run. Here are the possible *Index* values:

Table 8-2 *Testnum Values*

Index	*test to run*
0	all tests
1	fast ram test
2	medium ram test
3	burn-in ram test

Each test checks the memory with greater thoroughness; the fast ram test the least, the burn-in test the most. The default value for *Index* is 0, which runs all of the tests in order.

The second argument, *Pass=*, controls the number of times the test is run.

Shadow *Pass=*

The *shadow RAM test* ensures that the shadow RAM and associated circuitry keeps an accurate copy of the values in the register RAM. The shadow RAM is supposed to duplicate the values stored in the register RAM. This faster memory is used to speed up CPU ability to read the results of floating point instructions. The test writes a value into the register RAM, then reads the shadow RAM to insure that the value was correctly copied. This sequence is repeated for all legal values. If the test finds a discrepancy between the two RAMs, it displays an error message. The test accepts one argument, *Pass=*, which controls the number of times the test is run.

REgister *Pass=*

The *status register test* insures that the status register on the Weitek chip can be written to, and that the written values can be accurately read from the `Wstatus`, `clear` and `stable` registers. A four-bit pattern is written to the `Wstatus` register, then read from the `clear` and `stable` registers. This sequence is repeated for all possible four bit values. If the values read do not match the values written, the test displays an error message. The entire sequence is done twice; first with the Weitek `Imask` register set to zero (errors disabled), then with the `Imask` register set to one (errors are enabled).

The test accepts one argument, `Pass=`, which controls the number of times the test is run.

Mode *Pass=*

The *mode register test* checks the operation of the three mode registers on the FPA board: one write-only register connected to two read-only registers. The test makes sure four-bit values written to the `mode` register (write-only) get copied to the `mode stable` and `mode clear` (read-only) registers. The test accepts one argument, *Pass=*, which controls the number of times the test is performed.

Data *Pass=*

The *Weitek data path test* ensures that the data bus to the Weitek chips is

working correctly. The test sends values over the bus by sending `add 0 to value` and `multiply value by 1` commands to the Weitek chips. Using both add and multiply instructions checks the data path to both chips. The value resulting from the instruction is compared to the original value. If the values differ, the test displays an error message. The test accepts one argument, *Pass=*, which controls the number of times the test is performed.

WOp *Pass=*

The *Weitek Overall Operation Test* makes sure the Weitek chip can accurately perform all its floating point operations. This test checks command register single and double precision, extended single and double precision, and single and double precision short operations. The test loads the Mapping, and Microstore RAM, and loads in the appropriate constants. It then sets up the FPA registers to drive the Weitek chips to perform an operation. The test then reads the results and compares them again the correct answer. This cycle is repeated for every Weitek instruction.

WStatus *Pass=*

The *Weitek Overall Status Test* insures that the Weitek status register can properly display the status of floating point operations. The test sends a special set of floating point instructions to the Weitek chips, which should generate all of the possible floating point status conditions. After each instruction is sent, the test reads the status register to ensure that the proper status is reported. If the status register reports the wrong condition, the test displays an error message. The test accepts one argument, *Pass=*, which controls the number of times the test is performed.

Jump *Pass=*

The *jump conditions test* makes sure that the conditional branch microcode instruction works correctly. The test initializes Mapping Microcode and Constant RAM, and then sets up registers for branching operations. After a microcode conditional branch instruction is executed, the test checks that the program has jumped to the right place. The test is repeated for all types of branch instructions, setting the conditions so both branch and don't branch conditions are tested.

Pipeline *Pass=*

The *pipeline control timing test* makes sure the instruction pipeline can handle a high rate of FPA instructions. The Mapping, Microstore and Constant RAM are initialized, and a series of instructions are sent to the board. After each sequence, the test reads the register to see if the results are correct. A new instruction sequence is sent, until every possible instruction has been sent to the board.

?

The help command displays the current menu with the addition of the expected command parameters.

8.4. Utilities Menu

NOTE *This option is not functional at this time.*

The utilities menu provides the user limited access to the FPA board in order to troubleshoot it.

```
Sun 3 FPA Diagnostic  1.0  mm/dd/87 FPA Utility Menu

   Options      Display the Local Options Menu
   Dump         Dump The Ram Array
   Edit         Edit Ram Array
   Fill         Fill Ram Array
   Download     Download Ram Array
   ?            Display Help Menu

Command ==>
```

Options

The *options* command brings up a menu through which you may modify the diagnostic's default options.

Dump *Index= Offset= Size=*

The *dump ram* command allows you to display the contents of a section of the FPA board RAM. The command accepts three arguments: *Index=*, *Offset=*, and *Size=*. The *Index=* argument determines which area of memory is dumped. The table below shows the possible *Index* values and their meanings.

Table 8-3 *Index Values*

Value	*Location*
1	Mapping RAM
2	Microstore RAM
3	Register RAM

The *Offset=* parameter is the start address of the memory to be dumped; *Size* is the size (in hexadecimal) of the memory to be dumped.

Edit *Index= Offset=*

The *edit ram* FPA command provides for modification of the values in the board's memory. The command accepts two arguments: *Index=* and *Offset=*. The *Index=* argument determines which area of memory is dumped. The table at the beginning of this section shows the possible *Index=* values and their meanings.

The *Offset=* parameter is the start address of the memory to be modified. The address is in hexadecimal, and starts at 0 at the beginning of each RAM area.

Fill *Index= Begin= End=*

The *fill ram* command allows you to fill selected areas of memory on the

FPA board with a data pattern. The command accepts three arguments: *Index=*, *Begin=*, and *End=*. The *Index=* argument determines which area of memory is filled; the table at the beginning of this section shows the possible *Index* values and their meanings.

The *Begin=* parameter is the start address of the memory to be filled; *End=* is the address of the end of the region to fill. Both addresses are hexadecimal, and start at 0 at the beginning of each RAM area. When the fill command is executed, it interactively asks you to enter the value for each memory location to fill. Enter a hexadecimal value of the appropriate size for the memory being filled.

`Download` *Index*

The *download RAM* command displays the contents of an area of RAM to the console. The command accepts one argument, *Index*. This argument determines which area of memory is dumped; the table at the beginning of this section shows the possible *Index* values and their meanings.

`?`

The help command displays the current menu with the addition of the expected command parameters.

9

Graphics Processor1 Diagnostic

9

Graphics Processor1 Diagnostic

9.1. Introduction

The Sun GP1 and GB (Graphics Buffer Board) are two, full-size VME boards intended for use in any Sun color workstation that has a VME bus. It could also be used in a monochrome workstation that supports grayscale imaging (and has a modified color frame buffer board to do that)

Due to user requirements, this diagnostic does not name failing parts of the board. It displays actual and expected values of each test. See the section showing error messages for tips on how to interpret them to locate hardware faults.

The GP1 and GB only work in certain adjacent slots of a Sun workstation. For details, see a Cardcage Backplane and Slot Assignment document.

Read *Chapter 2* for information on booting the Diagnostic Executive, then select the GP1 test from the Diagnostics menu.

This diagnostic consists of a series of tests that run automatically with no user intervention, except in the case of the "scope loop" tests, which present submenus of options.

9.2. The Main Menu

The main menu provides these choices; you need only enter the letter(s) shown in upper case to make a selection.

```
GP1 Diagnostic    Rev x.x   Date      Main Menu

Gp    Run all GP tests.
S     Run slave tests.
Vp    Run the Viewing Processor tests.
Fp    Run Floating Point tests.
Pp    Run the Painting Processor tests.
Sfv   Move data from Shared Mem to FIFO to VME.
GB    Run all GB tests.
D     Test the GB DRAM.
I     Test the GB Integer Multiplier.
SL    Run Scope Loops.

Command ==>
```

The `Gp, S, Vp, Fp, Pp` and `Sfv` selections operate on the Graphics Processor Board, while the `GB, D` and `I` selections test the Graphics Buffer Board. If no Graphics Buffer Board is present, the test will query you about it

and then return you to the Main Menu.

Selecting G is the equivalent of selecting S, V, F, P and SF.

Selecting GB is equivalent to selecting D and I.

Selecting SL gives you hundreds of choices of writing/reading patterns throughout the GP1 and GB. The menus are shown later in this text. Once in a scope loop, you exit it by pressing "Control C."

9.3. The Scope Loop Menus

The **SL** selection from the main menu brings up this menu:

```
GP1 Diagnostic   Rev x.x    Date   Scope Loop Menu

S      Shared Memory from VME Scope Loops
M      Microstore from VME Scope Loops
A      Microstore Address Register
V      VP All Source and Destination Loop
P      PP and GB All Source and Destination Loop
D      Graphics Buffer DRAM Scope Loop

Command ==>
```

Each of the choices shown above brings up a sub-menu, described on the pages that follow, along with the sub-menu options.

S — Shared Memory

The **S** selection from the Scope Loop Menu brings up the following sub-menu of test choices:

```
GP1 Diagnostic  Revx.x   Date   Shmem Scope Loop Menu

FPL      Fixed pattern to 1 location
FPA      Fixed pattern to all locations
I        Incrementing pattern to incr loc
R        Read 1 location
RA       Read all locations

Command ==>
```

The FPL, FPA and R selections prompt you for a location, and then, if needed, a pattern:

```
Please select the location for doing scope loop:

if you just press [Return] , the program enters location 0x00 for you.
You are now prompted for a pattern:

Using location 0 (or the value you entered)
Please give a pattern to write (4 hex digits):
The range is 0 to 3fff (hex; don't use "0x" prefix):

If you do not enter a pattern, the default "0" is used and the test proceeds.
```

The **I** command writes incrementing patterns to incrementing locations.

The **RA** command reads every shared memory location.

M — Micstor Scope Loop

The **M** selection from the Scope Loop Menu brings up this test selection:

```
GP1 Diagnostic  Rev x.x  Date  Micstor Scope Loop Menu

FPL   Fixed Pattern to Microword Location
FPA   Fixed Pattern to all Microword Locations
I     Incrementing Pattern to incr locs
R     Read 1 Microword Location
RA    Read All Microword Locations

Command ==>
```

As described for the Shared Memory Scope Loop menu selection, the `FPL`, `FPA` and `R` selections from the menu above ask you to enter a location and pattern:

```
Please select the location for doing the scope loop

The range is 0 to 1fff (hex; don't use "0x" prefix)

and then...

Please enter your desired microword pattern (hex) in this format:

            xx  xxxx  xxxx  xxxx
```

If you just press [Return] without entering values, the default used is 0.

The **I** Microstore Scope Loop Menu choice writes incrementing patterns to incrementing microstore locations.

The **RA** selection reads all microstore locations.

A — Microstore Address Register

Selecting **A** from the Scope Loop Menu brings up the following sub-menu:

```
P      Write a pattern
R      Write a pattern then read forever
WTI    Write a pattern then increment
```

P If you select P from the Microstore Address Register test sub-menu, the program will prompt you for a pattern:

```
Please give a pattern to write (4 hex digits):
```

If you just enter [Return], a default 0x0000 pattern is used, and then the test announces:

```
NOW WRITING 0000 TO MICROSTORE ADDRESS REGISTER
```

R If you select R from the Microstore Address Register test sub-menu, the program prompts you for a pattern as shown above, and then announces:

```
NOW READING MICROSTORE ADDRESS REGISTER, EXPECT pattern you entered
```

WTI

If you select WTI from the Microstore Address Register test sub-menu, you are prompted for a pattern and the test announces:

```
USING pattern
INITIALIZED MICROSTORE ADDRESS REG WITH pattern
NOW INCREMENTING IT BY READING DATA REG
```

V — VP Scope Loop Menu

If you select V from the Scope Loop Menu, this sub-menu is displayed:

```
GP1 Diagnostic    Rev x.x   Date   VP Scope Loop Menu

I     Interprocessor Flag #2 Register
A     AM29116
F     FIFO #2
S     Shared Memory
P     VPPROM
G     General Field
F     Floating Point Register
FS    Floating Point Status Register

Command ==>
```

These menu choices, except for the I selection, each offer a list of possible destinations for the selected component. The selection simply moves the Interprocessor Flag#2 Register to the AM29116, which is the only possible destination for that register.

A If you select A, the program asks you to select a pattern after you choose one of these destinations for the AM29116 pattern:

```
                                        VP29116 Loop Menu

  Choose the destination for VP29116 pattern

  L     Status LED Register
  N     N Register
  I     Interprocessor Flag #1 Register
  F     FIFO #1
  A     AM29116
  B     Branch Register
  P     VPPROM Register
  S     Shared Memory
  FP    Floating Point Register
  FA    Floating Point Source A Register
  FB    Floating Point Source B Register
  FD    Floating Point Destination Pointer
  SP    Shared Memory Pointer
```

If, for example, you chose **A** from the VP29116 Loop Menu, the program would prompt you to enter a pattern and then announce:

```
MOVING AM29116 (pattern) TO STATUS LED REGISTER
```

F If you select F from the VPScope Loop Menu, these destination choices are offered:

```
  A     AM29116
  B     Branch Register
  V     VPPROM Register
  S     Shared Memory Pointer
  N     Never mind doing this

  Choose the destination for FIFO#2
```

If you select A, B, V or S, the program prompts you to enter a pattern and then moves that pattern from the FIFO#2 to your destination choice, announcing that it is doing so.

If you select N, you are returned to the VP Scope Loop Menu.

S If you select S from the VP Scope Loop Menu, this selection of destinations for the Shared Memory data is displayed:

```
command ==>s

F     FIFO #1
A     AM29116
B     Branch Register
P     VPPROM Register
R     Floating Point Register
V     Floating Point Source A Pointer
T     Floating Point Source B Pointer
D     Floating Point Destination B Pointer
S     Shared Memory Pointer
N     Never mind doing this

Choose the destination for Shared Memory Data
```

Any selection, excepting N, moves Shared Memory Data to the destination represented by the selection.

N returns you to the VP Scope Loop Menu.

P Selecting P from the VP Scope Loop Menu brings up these destination choices:

```
command ==>p

F      FIFO#1
A      AM29116
S      Shared Memory
R      Floating Point Register
N      Never mind doing this

Choose the destination where VPPROM goes to.
```

The N selection brings you back to the VPScope Loop Menu; all the other selections move the VPPROM to the destination represented by that choice.

FS An FS selection from the VP Scope Loop Menu brings up these destination choices:

```
command ==>fs

A      AM29116
S      Shared Memory
F      Floating Point Register

Choose the destination where FP Status Register goes to.
```

If you enter N, the VP Scope Loop Menu returns; any other choice moves the FP Status Register to the destination represented by the selection.

P — PP Scope Loop Menu

If you select P from the Scope Loop Menu, the PP Scope Loop Menu is offered:

```
GP1 Diagnostic   Rev x.x  Date   PP Scope Loop Menu

I     Interprocessor Flag#1 Register
A     AM29116
P     PPPROM
G     GB Read Data Register
V     VME Read Data Register
S     VME Status Register
GF    General Field
S     Scratchpad Memory
F     FIFO #1
M     Multiplier Result
```

All the selections shown above bring up sub-menus, with the exception of the first choice, I, which requires no user input. The sub-menus under the Pixel Processor (PP) Scope Loop Menu are shown on the following pages, in the order that they appear on the menu.

A The AM29116 selection from the PP Scope Loop Menu presents this sub-menu:

```
   L     Status LED Register
   N     N Register
   B     Branch Register
   SP    Scratchpad Register
   I     Interprocessor Flag#2 Register
   P     PPPROM
   F     FIFO
   A     AM29116
   S     Scratchpad Memory
   GD    GB Write Data Register
   VD    VME Write Data Register
   MX    Multiplier X Operand
   MY    Multiplier Y Operand
   MM    Multi-Mode
   ID    Internal ID Register
   VH    VME High Address
   VL    VME Low Address
   VC    VME Control
   GH    GB High Address
   GL    GB Low Address
```

P Selecting P from the PP Scope Loop Menu brings up this sub-menu:

```
F     FIFO#2
A     AM29116
S     Scratchpad Memory
G     GB Write Data Register
V     VME Write Data Register
X     Multiplier X Operand
Y     Multiplier Y Operand
N     Never mind doing this

Choose the destination where PPPROM data goes.
```

G A G selection brings up this sub-menu:

```
Choose the destination for GB Read Data Register:

A     AM29116
H     VME High Address Register
L     VME Low Address Register
B     Branch Register
S     Scratchpad Register
P     PP PROM Pointer
```

V Selecting V from the PP Scope Loop Menu brings up this sub-menu:

```
A     AM29116
V     VME Write Data Register
X     Multiplier X Operand
Y     Multiplier Y Operand
N     Never mind doing this

Choose the destination where VME Read Data Register goes to.
```

S Selecting S from the Scope Loop Menu brings up this sub-menu:

```
F     FIFO#2
A     AM29116
S     Scratchpad Memory
G     GB Write Data Register
N     Never mind doing this

Choose the destination for VME Status Register:
```

GF Selecting GF from the PP Scope Loop Menu brings up this sub-menu:

```
GF General Field

1       Branch Register
2       Scratchpad Pointer
3       Interprocessor Flag#2
4       PPPROM Pointer
5       FIFO #2
6       AM29116
7       Scratchpad Memory
8       GB Write Data Register
9       VME Write Data Register
10      Multiplier x Operand
11      Multiplier y Operand
12      Multiplier Mode Register
13      Interrupt ID Register
14      VME High Address Register
15      VME Low Address Register
16      VME Control Register
17      GB High Address Pointer
18      GB Low Address Pointer

Choose the destination for General Field Data
```

After you have made a destination choice from the General Field menu, you will be prompted to provide a pattern.

M When you select **M** from the PP Scope Loop menu, this sub-menu is displayed:

```
F       FIFO #2
A       AM29116
S       Scratchpad Memory
G       GB Write Data Register
V       VME Write Data Register
X       Multiplier X Operand
Y       Multiplier Y Operand
N       Never mind doing this

Choose the destination for the Multiplier Result Data.
```

If you select N, you will be returned to the PP Scope Loop Menu.

D — DRAM Scope Loop Menu

When you select D from the Scope Loop Menu, this sub-menu is displayed:

```
GP1 Diagnostic  Rev x.x  Date  DRAM Scope Loop Menu

W     Write to one word address
WR    Write to one word address, then read it back forever
WA    Write address-unique patterns to 4K-word block
R     Read one word address

command ==>
```

W If you select w from the DRAM Scope Loop Menu, the program prompts you to select a location and a pattern for the test:

```
Please select the location for doing the scope loop:
```

you enter an address

```
Please give pattern to write (4 hex digits):
```

you enter a 4-digit hex value

NOW WRITING *pattern* TO ADDR HIGH= *hex value* ADDR LOW= *hex value* IN DRAM

WR

To be added

WA If you select WA from the DRAM Scope Loop Menu, you are prompted for a location (as shown above), and, after you enter one, the test announces:

```
NOW WRITING ADDRESS UNIQUE COUNTING PATTERNS,
STARTING FROM 0 TO 4-K WORD BLOCK
STARTING ADDR HIGH=hex value LOW=hex value IN DRAM
```

9.4. Error Messages

Messages from the GP1/GB diagnostic have a number at their beginning, which helps you find them in this text. Here they are, with a brief discussion of the problem.

Where the message examples show a value with a percent sign, it indicates how the value will be presented under running conditions.

%d a decimal value (0's - 9's).
%x a hexadecimal value (0's - f's).
%b a binary value (0's and 1's).

NOTE: Following these error messages, see a discussion of ABORTION messages, such as this:

30 VP Status Register test aborted because of the following:
Couldn't stop the VP in order to load microcode. Try the slave reg tests.
Couldn't start the VP after loading microcode. Try the slave reg tests.

1 CAUTION: GP diag had trouble determining the size of GP microstore. That could mean trouble with VME slave interface with the board.

The two different types of GP1 board are usually called GP and GP+. The main difference between them is the microstore size. Since this diagnostic can handle both kinds of board, it does a simple test at the beginning to determine the size of microstore. This message is a warning that something prevented this simple test from getting expected results. Expect a subsequent test to reveal more information, unless the VMEbus interface is quite seriously broken.

5 Board id error found, should have value other than 0.

The board ID is one read-only byte at the first address of the GP. The hardware documentation on that byte says only that it is not zero; therefore the test can only check for a non-zero value. This message says that when the diagnostic read the board ID, it was zero.

6 Status Register error found. Did board reset, but certain bits in SR not zeroed. Status Register ANDed by 0x%x = %x

The status register test does very little because the GP Status Register is read-only. It uses the write-only GP Control Register to generate a board reset, then reads the GP Status Register and expects bits 8, 9, 10 and 14 to be zeros. This message indicates a difference between observed and expected results. Suspect the reset and status register hardware.

7 Status Reg : xxxx xxxx xxxx xxxx
BAD: Expected bit 14, Interrupt Enabled, to be zero.

The test tried to change the value of this bit to zero by using the write-only Control Register. Since that didn't work, suspect both the control and status registers.

8 Status Reg : xxxx xxxx xxxx xxxx
BAD: Couldn't TOGGLE bit 14, Interrupt Enabled, from x to x.

The test tried to toggle this bit by using the write-only Control Register. Since that didn't work, suspect both the control and status registers.

10 Status Reg : xxxx xxxx xxxx xxxx
BAD: Couldn't change bit 14, Interrupt Enabled, from x to x.

The test tried to change this bit by using the write-only Control Register. Since that didn't work, suspect both the control and status registers.

11 BAD: Expected bit 10, Reset, to be zero.

Following a reset attempt using the Control Register, this bit should show zero, but it doesn't. Suspect the reset, control and status register hardware.

12 Status Reg : xxxx xxxx xxxx xxxx
BAD: Couldn't change bit 10, Reset, from x to x.

The test tried to change this bit by using the write-only Control Register. Since that didn't work, suspect both the control and status registers.

14 BAD: Expected bit 9, to be zero.

Following a reset attempt using the Control Register, this bit should show zero, but it doesn't. Suspect the reset, control and status register hardware.

15 Status Reg : xxxx xxxx xxxx xxxx BAD: Couldn't change bit 9 from x to x.

The test tried to change this bit by using the write-only Control Register. Since that didn't work, suspect both the control and status registers.

16 BAD: Expected bit 8, to be zero.

Following a reset attempt using the Control Register, this bit should show zero, but it doesn't. Suspect the reset, control and status register hardware.

17 Status Reg : xxxx xxxx xxxx xxxx BAD: Couldn't change bit 8 from x to x.

The test tried to change this bit by using the write-only Control Register. Since that didn't work, suspect both the control and status registers.

31 Couldn't get expected pattern in VP status register.
Observed xxxx Expected xxxx

The diagnostic used VP microcode to try and put a certain 4-bit pattern into the VP field of the GP Status Register, but failed. Suspect the VP 29116, the GP status register and the paths between them.

36 VP's interprocessor flags bit 8 is a 1, but it should always be a 0.

As a requirement for the two processors to distinguish their unique code in a shared microstore, the VP's interprocessor flag number 8 must always be a 0. If this message comes up, suspect the `ipflags` buffer UJ4, the VP 29116, and the path between them.

38 VP N Register pattern test error found.
OBS (binary) = xxxx EXP (binary) = xxxx

The test moved a pattern to the 4-bit N Register, then used that to make a binary calculation. This message says that the calculation gave wrong results. Suspect the N Register, the VP 29116 and the paths between them.

41 VP AM29116 n field assembly constant test error found.
OBS (binary) = xxxx EXP (binary) = xxxx

This test used a pattern in the microcode instruction to make a binary calculation in the VP 29116. This message says that the calculation gave wrong results. Suspect the Microcode decoding hardware for the N field, the VP 29116 and the path between them.

46 VP General Field Immediate Value Test error found.
OBS = 0x%x EXP = 0x0

The test put a pattern into the general field of the micro-instruction and moved it to shared memory for comparison with the expected pattern. Since shared memory and the VP bus to the shared memory path were already tested, suspect

the general field data path from microstore to the VP bus.

49 VP General Field Addressing Test error found.
Couldn't jump to microstore 56-bit word location %x(hex)

The test put a pattern into the general field of the micro-instruction for use as a branch address. Since the microcode jumped to a different address, suspect the general field data path from microstore to the 2910A.

51 VP Branch Register error found.
Couldn't jump to microstore 56-bit word location %x(hex)

The test put a pattern into the VP branch register, then tried to branch to that as an address. Since the microcode jumped to a different address, suspect the branch register and its connection to the VP bus.

56 Couldn't load microcode into location 0x%x. Happens if very small microcode size.

The microstore banks select test, during setup, tried to load some microcode into a high microstore location, but couldn't verify it. That could mean that your GP1 has very a very small microstore. This message is very unlikely to show up.

57 VP Bank Select and address buffer to 2nd bank error found.
Couldn't jump to second bank (ustore 56-bit word loc 0x%x

The VP microstore bank select test was unable to jump to a microstore location that needs the bank select hardware to carry out the jump. Suspect the bank select hardware.

58 VP Bank Select error found.
Couldn't jump back to first bank (ustore 56-bit word loc 0x%x)

The VP microstore bank select test was unable to jump from a high bank of microstore to an instruction in bank 0. Suspect the bank select hardware.

61 VP Shared Memory address unique test error found.
Word Address 0x%x OBS = 0xXXXX EXP = 0xXXXX
If you already tested shared memory with the slave test, suspect the VP's Shared Memory Pointer.

The message is self explanatory.

62 VP Shared Memory read/write test (shmem[i]->am29116->shmem[i-1]).
Word Loc 0x%x OBS = 0x%x EXP = 0x%x
(NOTE: if you continued after a previous error, expect this error.)

This test used the VP to read shared memory and write back to shared memory in a different location. If no previous shared memory test failed, suspect the VP Shared Memory Pointer or the VP bus between the 29116 and the shared memory pointer.

64 GP microstore address reg error found.
OBS = 0x%x EXP = 0x%x

The address register should be capable of holding any pattern. This test wrote the expected pattern to it but read back a different one. Suspect the address

register or the VME bus path to it.

65 GP microstore address reg test error found.
16-bit word window movement in GP Status Reg isn't right.

The GP Status Register has two bits that show which of four "window" locations are currently connected to the microstore Data Register. (A microword is 56 bits wide but you can only access it 16 bits at a time.) This test kept track of which part of the microword it was writing to and compared it with the number in those two bits of the GP status register. Because they disagreed, suspect the GP status register and the path between it and the microstore hardware.

67 GP microstore address unique test error found.
ustore loc = 0x%x OBS = 0xhh hhhh hhhh hhhh
EXP = 0xhh hhhh hhhh hhhh

The test wrote different patterns to each location of microstore, then read them back and found an error. Suspect the Address Register, the microstore RAM addressing circuitry and the paths between them.

68 GP microstore random data test error found.
ustore loc = 0x%x OBS = 0xhh hhhh hhhh hhhh
EXP = 0xhh hhhh hhhh hhhh

The test wrote random patterns to microstore, then read them back and found an error. Suspect the microstore RAM and the path between it and the VME bus.

69 GP microstore pattern test error found.
ustore loc = 0x%x OBS = 0xhh hhhh hhhh hhhh
EXP = 0xhh hhhh hhhh hhhh

The test wrote a constant pattern to microstore. Upon read-back, it found an error. Suspect the microstore RAM and the path between it and the VME bus.

71 PP FIFO Test did a control register reset.
High byte of PP's ipflag register: xxxx xxxx
PP ipflags bit 9 says FIFO direction is PP to VP; a reset should have changed that.

Suspect the PP Interprocessor Flags Register 2 hardware, the FIFO Direction Control hardware, the path between them, the reset circuitry and its path to the FIFO Direction Control hardware.

72 PP FIFO Test did a control register reset. High byte of PP's ipflag register: %xxxx xxxx. PP ipflags bit 9 says FIFO direction is VP to PP, but VP supposedly changed it to the other way.

Suspect the PP Interprocessor Flags Register 2 hardware, the FIFO Direction Control hardware and the path between them.

74 High byte of PP's ipflag register: xxxx xxxx. PP ipflags bit 10 says FIFO is not empty, but reset should have cleared FIFO.

Suspect the PP Interprocessor Flags Register 2 hardware, the FIFO Synchronizer hardware, the path between them, the reset circuitry and its path to the FIFO Synchronizer hardware.

75 PP ipflags say FIFO is empty, but PP wrote some words to FIFO.

Suspect the PP Interprocessor Flags Register 2 hardware, the FIFO Synchronizer hardware, the FIFO Write Control hardware and the paths among them.

77 Test expected to read xxxx in the fifo but instead read xxxx.

Suspect the FIFO and its connection with the VP bus.

80 FIFO control is bad. Did series of PP writes to empty FIFO; expected to reach FIFO-full condition after %d times but instead wrote %d times. (Note: used cond code to determine full state, so check that hardware.)

Suspect the PP Interprocessor Flags Register 2 hardware, the FIFO Synchronizer hardware, the FIFO Write Control hardware and the paths among them. Also suspect the PP Condition Code Select hardware. Your GP1 board may have a different size of FIFO than the diagnostic expected.

81 PP FIFO Test. For read # %d, FIFO showed %x. Expected %x.

The test filled the FIFO with positionally unique data. Upon read-back, the data did not compare. Suspect the FIFO, the FIFO Write Registers, the FIFO Read Buffers and their connections to the PP bus.

82 PP FIFO control is bad. Did series of VP reads of full FIFO; expected to reach FIFO-empty condition after %d times but instead read %d times. (Note: used cond code to determine full state, so check that hardware.)

Suspect the PP Interprocessor Flags Register 2 hardware, the FIFO Synchronizer hardware, the FIFO Write Control hardware and the paths among them. Also suspect the PP Condition Code Select hardware. Your GP1 board may have a different size of FIFO than the diagnostic expected.

86 Couldn't get expected pattern in PP status register. Observed xxxx Expected xxxx

The PP Status Register was supposed to show the expected pattern, but held another pattern. Suspect the PP Status Register, the GP Status Register, PP Destination Control (UP16) and the paths among them.

89 PP's interprocessor flags bit 8 is a 0, but it should always be a 1.

As a requirement for the two processors to distinguish their unique code in a shared microstore, the PP's interprocessor flag number 8 must always be a 1. Suspect the `ipflags` buffer UJ4, the PP 29116 and the path between them.

91 PP N Register pattern test error found
OBS (binary) = xxxx EXP (binary) = xxxx

The test moved a pattern to the 4-bit N Register, then used that to make a binary calculation. This message says that the calculation gave wrong results. Suspect the N Register, the PP 29116 and the paths between them.

95 PP AM29116 n field assembly constant test error found.
OBS (binary) = xxxx xxxx xxxx xxxx EXP (binary) = xxxx xxxx xxxx xxxx

This test used a pattern in the microcode instruction to make a binary calculation in the PP 29116. This message says that the calculation gave wrong results. Suspect the Microcode decoding hardware for the N field, the PP 29116 and the path between them.

98 PP General Field Immediate Value Test error found
OBS = 0x%x EXP = 0x%x

The test moved an immediate value from a field in the microcode word, through the FIFO to shared memory. Because you already tested microstore, the FIFO, the VP bus and shared memory, suspect the PP bus between microstore general field and the FIFO.

101 PP General Field Branch Function error found
Couldn't jump to u-store 56-bit word location %x(hex)

The test put a pattern into the general field of the micro-instruction for use as a branch address. Remember that "56-bit word location" means the location of the 56-bit word, not that the address is 56 bits long. Since the microcode jumped to a different address, suspect the general field data path from microstore to the PP 2910A.

105 IP Flag #2 pattern test error found
OBS = 0x%x EXP = 0x%x

The test moved a pattern from the PP to the Interprocessor Flags Register 2 and from there to shared memory. Since you already checked the PP immediate field and all of the VP hardware, suspect the PP IPflag hardware, the bus between the PP general field and that hardware and the VP path between the IPflag hardware and shared memory.

106 IP Flag #1 pattern test error found
OBS = 0x%x EXP = 0x%x

The test moved a pattern from the VP to the Interprocessor Flags Register 1 to the Interprocessor Flags Register 2 and from there to shared memory. Suspect the path between the VP microcode general field and IPflag 1, IPflag 1, the path between IPflag 2 and the PP 29116 and the path between PP 29116 and IPflag 2.

109 Got a timeout while accessing GB board. Is one installed?
This test requires a Jumper J7 on GP and a working GB in the system.

The test tried to use the PP to communicate with the GB, but a timeout indicated that this was unsuccessful. If you really have a GB installed, verify that the GP and GB are in suitable slots of the workstation. They communicate using the P2

connectors. However, not all adjacent slot pairs are connected together on the backplane. If the boards are in the correct slots, suspect the PP Bus Extension Transceivers and Decoders, the GP/GB Interface Signal Decoders on the GB and the backplane of the workstation.

110 GP GB interface test error found
OBS = 0x%x EXP = 0x%x
This test requires a Jumper J7 on GP and a working GB in the system.

The test wrote patterns to DRAM on the GB, read them back and found a miscomparison. If you have not yet tested DRAM, suspect that. Also suspect the PP Bus Extension Transceivers and Decoders, the GP/GB Interface Signal Decoders on the GB and the backplane of the workstation.

112 PP continue not on zero error found.
Could not start PP after halted it.

The test used a control bit of the GP Control Register that allows software to start the PP from the halted state without first setting its program sequencer to location zero of microcode. Since the diagnostic never does that under normal operation, this test does it. Suspect the PP 2910A Sequencer, the GP Control Register hardware, the PP Run/Halt hardware and the paths among them.

113 PP continue not on zero error found.
Could not start PP not on zero.

The test used a control bit of the GP Control Register that allows software to start the PP from the halted state without first setting its program sequencer to location zero of microcode. Since the diagnostic never does that under normal operation, this test does it. Suspect the PP 2910A Sequencer, the GP Control Register hardware, the PP Run/Halt hardware and the paths among them.

120 FIFO control is bad. Did control reg reset; the VP ipflag should have said FIFO direction is VP to PP, but it didn't.

Suspect the PP Interprocessor Flags Register 1 hardware, the FIFO Direction Control hardware, the path between them, the reset circuitry and its path to the FIFO Direction Control hardware.

121 FIFO control is bad. Did control reg reset; the VP ipflag should have said FIFO is empty, but it didn't.
Suspect the PP Interprocessor Flags Register 1 hardware, the FIFO Synchronizer hardware, the path between them, the reset circuitry and its path to the FIFO Synchronizer hardware.

122 FIFO control is bad. Wrote two words to empty FIFO; the VP ipflag should have said FIFO not empty, but it didn't.
In fact, it gave a fifo1 full cond code to the microcode!

The last line won't always appear. Suspect the PP Interprocessor Flags Register 1 hardware, the FIFO Synchronizer hardware, the FIFO Write Control hardware and the paths among them. If the last line is there, suspect the VP Condition Code Select hardware.

123 FIFO control is bad. Tried to set FIFO direction to PP to VP, but VP ipflag says direction is VP to PP.

VP software can change the FIFO direction. The test tried to do that but failed. Suspect the PP Interprocessor Flags Register 1 hardware, the FIFO Direction Control hardware and the path between them.

124 FIFO data error. Wrote two words to FIFO from VP; read them back in VP. Second word: actual %x expected %x

Suspect the VP bus, the FIFO and the path between them.

125 FIFO control is bad. Did a FIFO direction toggle; after that, the VP ipflag should have said FIFO direction is VP to PP, but it didn't.

Suspect the VP Interprocessor Flags Register 1 hardware, the FIFO Direction Control hardware and the path between them.

126 FIFO data path is bad. Did a VP write to FIFO and read-back. Actual = %x Expected = %x

Suspect the FIFO and its connection to the VP bus.

127 FIFO control is bad. Did series of VP writes to empty FIFO; expected to reach FIFO-full condition after %d times but instead wrote %d times. (Used cond code to determine full state, so check that hardware too.)

Suspect the VP Interprocessor Flags Register 1 hardware, the FIFO Write Control hardware, the FIFO Read Control hardware, the FIFO Synchronizer and the paths among them. Also check the Condition Code Select hardware and the 2910A.

128 Tested VP PROM checksum, using fifo. Actual = %x Expected = %x.

If you already trust the VP PROM and the FIFO, suspect VP bus control. This is the only time the diagnostic moves data from the VP PROM to the FIFO, but both of those have used the VP bus before, so their data paths should be okay. Suspect VP Source and VP Destination hardware.

131 FP A Source Register Pattern Test error found
LOC(of 16-bit word) = 0xhhhh
OBS = 0xhhhh EXP = 0xhhhh

The test wrote and read back patterns in the Floating Point Registers Set A and found a miscomparison. Suspect the registers and their connection to the VP bus.

132 FP B Source Register Pattern Test error found
LOC(of 16-bit word) = 0xhhhh
OBS = 0xhhhh EXP = 0xhhhh
The test wrote and read back patterns in the Floating Point Registers Set B and found a miscomparison. Suspect the registers and their connection to the VP bus.

133 FP A Source Register Addressing Test error found
LOC(of 16-bit word) = 0xhhhh
OBS = 0xhhhh EXP = 0xhhhh

The test wrote and read back address-unique patterns in the Floating Point Registers Set A and found a miscomparison. Suspect the set A Floating Point Addressing Source and Destination Pointers and their connections to the VP bus.

133 FP B Source Register Addressing Test error found
LOC(of 16-bit word) = 0xhhhh
OBS = 0xhhhh EXP = 0xhhhh

The test wrote and read back address-unique patterns in the Floating Point Registers Set B and found a miscomparison. Suspect the set B Floating Point Addressing Source and Destination Pointers and their connections to the VP bus.

135 FP Source Register pointer uniqueness test error found
OBS = 0x%x EXP = 0x%x, Should check A and B pointers

The message is self explanatory.

136 FP Source Register pointer test error found
32-bit Addr = 0x%x OBS = 0x%x EXP = 0x0, Check D and A ptr

The message is self explanatory. "D ptr" means Destination Pointer.

140 FP Flowthru ALU operation Float/Fix error found
OBS = 0x%x EXP = 0x%x
FP Status Register = xxxx xxxx xxxx xxxx

The test used the Weitek ALU chip to convert fixed-point data to floating-point data and back again. Since that failed, suspect the chip and the path between it and the VP bus.

141 FP Flowthru ALU operation A+B error found
OBS = 0x%xEXP = 0x%x
FP Status Register = xxxx xxxx xxxx xxxx

Suspect the Weitek ALU chip and the path between it and the VP bus.

142 FP Flowthru ALU operation A-B error found
OBS = 0x%xEXP = 0x%x
FP Status Register = xxxx xxxx xxxx xxxx

Suspect the Weitek ALU chip and the path between it and the VP bus.

143 FP Flowthru ALU operation -A+B error found
OBS = 0x%xEXP = 0x%x
FP Status Register = xxxx xxxx xxxx xxxx

Suspect the Weitek ALU chip and the path between it and the VP bus.

144 FP Flowthru ALU operation |A| + |B| error found
OBS = 0x%xEXP = 0x%x
FP Status Register = xxxx xxxx xxxx xxxx

Suspect the Weitek ALU chip and the path between it and the VP bus.

145 FP Flowthru ALU operation |A-B| error found
OBS = 0x%xEXP = 0x%x
FP Status Register = xxxx xxxx xxxx xxxx

Suspect the Weitek ALU chip and the path between it and the VP bus.

146 FP Flowthru ALU operation |A+B| error found
OBS = 0x%xEXP = 0x%x
FP Status Register = xxxx xxxx xxxx xxxx

Suspect the Weitek ALU chip and the path between it and the VP bus.

147 FP Flowthru Multiplier operation AxB error found
OBS = 0x%xEXP = 0x%x
FP Status Register = xxxx xxxx xxxx xxxx

Suspect the Weitek Multiplier chip and the path between it and the VP bus.

148 FP source reg high/low word order control error found
OBS high word = 0x%x low word = 0x%x
EXP high word = 0x0 low word = 0x%x
FP Status Register = xxxx xxxx xxxx xxxx

The VP bus is 16 bits wide, but the Weitek chips give 32-bit results. Suspect the hardware which correctly orders the two words from the Weitek chips to the VP bus. This includes UA18 and UB18.

151 FP Pipeline, using result register, operation (A+B)*B error found
OBS = 0x%xEXP = 0x%x
FP Status Register = xxxx xxxx xxxx xxxx

Pipelining is a feature of the two Weitek floating-point processor chips. Suspect them both and the result registers, UC25 and UD25.

155 FP Pipeline Matrix Multiplication, (1 by 4) . (4 by 1), error found
OBS = %dEXP = 20

Suspect the Weitek multiplier chip.

158 FP Status Reg error found, can't force ainv to occur.

Suspect both Weitek chips, FP Status Register chip UD21 and the paths among them.

159 FP Status Reg error found, can't force inv to occur.

Suspect both Weitek chips, FP Status Register chip UD20 and the paths among them.

160 FP Status Reg error found, can't force ainx to occur.

Suspect both Weitek chips, FP Status Register chip UD21 and the paths among them.

161 FP Status Reg error found, can't force aovr to occur.

Suspect both Weitek chips, FP Status Register chip UD21 and the paths among them.

162 FP Status Reg error found, can't force inx to occur.

Suspect both Weitek chips, FP Status Register chip UD20 and the paths among them.

163 FP Status Reg error found, can't force ovr to occur.

Suspect both Weitek chips, FP Status Register chip UD20 and the paths among them.

164 FP Status Reg error found, can't force asgn to occur.

Suspect both Weitek chips, FP Status Register chip UD21 and the paths among them.

165 FP Status Reg error found, can't force aund to occur.

Suspect both Weitek chips, FP Status Register chip UD21 and the paths among them.

166 FP Status Reg error found, can't force sgn to occur.

Suspect both Weitek chips, FP Status Register chip UD20 and the paths among them.

167 FP Status Reg error found, can't force und to occur.

Suspect both Weitek chips, FP Status Register chip UD20 and the paths among them.

170 VP PROM Checksum Test error found
OBS = %x(hex)EXP = %x(hex)
VP PROM word location 0x%x OBS = 0x%x EXP = xxxx

Suspect the VP PROM and the path between it and the VP bus.

175 VP Two Address Operation, r[%d] + acc = r[%d], error found
OBS r[%d] = 0x%x EXP r[%d] = 0x%x

Suspect the VP 29116, the 29116 RAM Address Select hardware and the path between them. Also suspect the path for micro-word bit 51 to the 29116 RAM Address Select hardware.

179 VP continue not on zero error found.
Could not start VP after halted it.

The test used a control bit of the GP Control Register that allows software to start the VP from the halted state without first setting its program sequencer to location zero of microcode. Since the diagnostic never does that under normal

operation, this test does it. Suspect the VP 2910A Sequencer, the GP Control Register hardware, the VP Run/Halt hardware and the paths among them.

180 VP continue not on zero error found.
Could not start VP not on zero.

The test used a control bit of the GP Control Register that allows software to start the VP from the halted state without first setting its program sequencer to location zero of microcode. Since the diagnostic never does that under normal operation, this test does it. Suspect the VP 2910A Sequencer, the GP Control Register hardware, the VP Run/Halt hardware and the paths among them.

189 PP Branch Register error found.
Couldn't jump to u_store 56-bit word location %x(hex)

The test put a pattern into the PP branch register, then tried to branch to that as an address. Since the microcode jumped to a different address, suspect the branch register and its connection to the PP bus.

193 PP Bank Select test couldn't load microcode into location 0x%x.

The microstore bank select test, during setup, tried to load some microcode into a high microstore location, but it couldn't verify it. That could mean that your GP1 has very a very small microstore. This message not likely to show up.

194 PP Bank Select and address buffer to 2nd bank error found.
Couldn't jump to second bank (ustore 56-bit word loc 0x%x)

The PP microstore bank select test was unable to jump to a microstore location that needs the bank select hardware to carry out the jump. Suspect the bank select hardware.

195 PP Bank Select error found.
Couldn't jump back to first bank (ustore 56-bit word loc 0x%x

The PP microstore bank select test was unable to jump from a high bank of microstore to an instruction in bank 0. Suspect the bank select hardware.

199 PP Scratchpad constant pattern write/read test error found.
Word Loc: 0x%x OBS = 0x%x EXP = 0x0

The test wrote all zeros to the PP Scratchpad RAM and read it back. Since the value read back was not zero, suspect the Scratchpad RAM and the path between it and the PP bus.

200 PP Scratchpad test word Loc: 0x%x OBS = 0x%x EXP = 0xhhhh

The test wrote the expected pattern to the PP Scratchpad RAM and read it back. Since the value read back was wrong, suspect the Scratchpad RAM and the path between it and the PP bus.

201 PP Scratchpad address unique write/read test error found.
Word Loc: 0x%x OBS = 0x%x EXP = 0x%x

Suspect the Scratchpad Pointer hardware and the path between it and the PP bus.

205 VME Bus Master word write to shmem [0x%x] error found.
OBS = 0x%x EXP = 0x%x

The test wrote to slave shared memory using the PP Bus Master hardware. Suspect the VME Master Data Out Register hardware and its connections to the VP bus, the VME Bus Master Data Transfer Controller and the Miscellaneous VME Master Logic.

206 VME Bus Master byte write to high or low byte in shmem word location 0x%x OBS word = 0x%x EXP high byte = 0x%x low byte = 0x%x

The test wrote to slave shared memory using the PP Bus Master hardware. Suspect the VME Master Data Out Register hardware and its connections to the VP bus, the VME Bus Master Data Transfer Controller and the Miscellaneous VME Master Logic.

207 VME low address register error found.
Low Addr = 0x%x OBS data = 0x%x EXP data = 0xffff

This test used walking 1 locations in shared memory, as accessed over the VME bus, to write all ones. Since it failed, suspect the VME Address Registers UP7 and UP12 and their connections to the PP bus.

211 VME Bus Master word read from shmem [0x%x] error found.
OBS = 0x%x EXP = 0x%x

The test read slave shared memory using the PP Bus Master hardware. Suspect the VME Master Data Out Register hardware and its connections to the VP bus, the VME Bus Master Data Transfer Controller and the Miscellaneous VME Master Logic.

212 VME Bus Master byte read from byte loc shmem[0] error found.
OBS = 0x%x EXP = 0x%x

The test read slave shared memory using the PP Bus Master hardware. Suspect the VME Master Data Out Register hardware and its connections to the VP bus, the VME Bus Master Data Transfer Controller and the Miscellaneous VME Master Logic.

213 VME Bus Master byte read from byte loc shmem[1] error found.
OBS = 0x%x EXP = 0x%x

The test read slave shared memory using the PP Bus Master hardware. Suspect the VME Master Data Out Register hardware and its connections to the VP bus, the VME Bus Master Data Transfer Controller and the Miscellaneous VME Master Logic.

214 VME bus master address counter error. At color board location 0xhh hhhh
expected xxxx xxxx xxxx xxxx
actual xxxx xxxx xxxx xxxx

This test used walking 1 locations on the color board, as accessed over the VME bus, to write all address-unique data. Since it failed, suspect the VME Address Register UP6 and its connections to the PP bus.

216 PP CC Select Test generated 29116 negative condition code, but cc select logic didn't pass that signal to 2910A

Suspect the PP 2910A and the PP Condition Code Select hardware. Also suspect the path from microstore to the PP 2910A for bit 55 (including the 3-Way Branch Control hardware).

218 PP CC Select test generated carry, fifo not full, fifo not empty cc's. The PP incorrectly detected a 29116 NEGATIVE cc. Check cc select logic.

Suspect the PP 2910A and the PP Condition Code Select hardware.

219 PP CC Select test didn't detect high bit of dreg field (microcode") bit 39), programmable negation bit.

Suspect chip UD14 and the bit 39 path from microstore to it.

220 PP CC Select test generated pp 29116 carry condition code, but CC select logic didn't pass that signal to 2910A.

Suspect the PP 2910A and the PP Condition Code Select hardware. Also suspect the path from microstore to the PP 2910A for bit 55 (including the 3-Way Branch Control hardware).

221 PP CC Select test generated negative, fifo not full, fifo not empty cc's. The PP incorrectly detected a 29116 CARRY cc. Check cc select logic.

Suspect the PP 2910A and the PP Condition Code Select hardware.

222 PP CC Select test generated pp 29116 zero condition code, but cc select logic didn't pass that signal to 2910A.

Suspect the PP 2910A and the PP Condition Code Select hardware. Also suspect the path from microstore to the PP 2910A for bit 55 (including the 3-Way Branch Control hardware).

223 PP CC Select test generated carry, fifo not full, fifo not empty cc's. The PP incorrectly detected a 29116 ZERO cc. Check cc select logic.

Suspect the PP 2910A and the PP Condition Code Select hardware.

224 PP CC Select test generated pp 29116 overflow condition code, but cc select logic didn't pass that signal to 2910A.

Suspect the PP 2910A and the PP Condition Code Select hardware. Also suspect the path from microstore to the PP 2910A for bit 55 (including the 3-Way Branch Control hardware).

225 PP CC Select test generated carry, fifo not full, fifo not empty cc's. The PP incorrectly detected a 29116 OVERFLOW cc. Check cc select logic.

Suspect the PP 2910A and the PP Condition Code Select hardware.

226 PP CC Select test moved a zero in 29116 with STATUS ENABLED. CC logic behaved as if zero cc weren't set.

Suspect the PP 2910A and the PP Condition Code Select hardware. Also suspect the path from microstore to the PP 2910A for bit 55 (including the 3-Way Branch Control hardware).

227 PP CC Select test moved a zero in 29116 with status enabled; then moved a 1 with STATUS NOT ENABLED. CC logic said zero cc wasn't still set. Check cc select logic.

Suspect the PP 2910A and the PP Condition Code Select hardware.

228 PP CC Select test did a reset, which should have emptied the fifo. The PP 2010A did not detect fifo2 not full. Check cc select logic.

Suspect the reset circuitry and its path to the FIFO Synchronizer hardware and the PP 2910A and the PP Condition Code Select hardware.

229 PP CC Select test filled the FIFO, but 2910A detected 'fifo2 not full cc. Check cc select logic.

Suspect the FIFO Control hardware and its connections to the cc select logic.

230 PP CC Select test cleared the FIFO but the 2910A still detected 'fifo1 not empty' cc. Check cc select logic.

Suspect the FIFO Control hardware and its connections to the cc select logic.

231 PP CC Select test wrote to FIFO, but PP 2910A failed to detect 'fifo1 not empty' condition code. Check cc select logic.

Suspect the FIFO Control hardware and its connections to the cc select logic.

232 PP CC Select test. 2910A failed to detect 'vme ready' condition code. Check cc select logic.

Suspect the VME Bus Requester hardware, the cc select logic and the path of the VMERDY signal between them.

233 PP CC Select test. 2910A detected 'vme ready' condition code when it shouldn't have been ready. Check cc select logic.

Suspect the VME Bus Requester hardware, the cc select logic and the path of the VMERDY signal between them.

234 PP 2910A failed to detect 'gb ready' condition code. Check cc select logic.

Check that you have a GB installed in your system. If you really have a GB installed, verify that the GP and GB are in suitable slots of the workstation back-plane. They communicate using the P2 connectors. However, all the slots are not connected together there. If the boards are in the correct slots, suspect the Z Buffer Flag hardware, the Condition Code Select hardware and the path between them.

236 3-way branch didn't expect the VME busy condition, but that's what happened. (Microcode primitive p99.)

Three-way branching gives the PP three possible addresses to branch to, depending on a programmed condition code or the VME-busy state of the workstation. The microcode primitive message is there to assist debugging by engineering. This message means that the condition code hardware unexpectedly detected the VME Busy condition.

Suspect the PP Condition Code Select hardware, the VME Bus Requester hardware, the VME Bus Master Data Transfer Controller and the paths among them.

237 3-way branch failed to detect zero condition code. It correctly detected that VME was not busy, however.

Suspect the PP 2910A and the PP Condition Code Select hardware.

238 PP 3-way branch hardware didn't expect the VME busy condition, but that's what happened. Microcode primitive p101.)

Three-way branching gives the PP three possible addresses to branch to, depending on a programmed condition code or the VME-busy state of the workstation. The microcode primitive message is there to assist debugging by engineering. This message means that the condition code hardware unexpectedly detected the VME Busy condition.

Suspect the PP Condition Code Select hardware, the VME Bus Requester hardware, the VME Bus Master Data Transfer Controller and the paths among them.

239 3-way branch detected zero condition code when that cc wasn't set. It correctly detected that VME was not busy, however.

Suspect the PP 2910A and the PP Condition Code Select hardware.

240 3-way branch failed to detect vme busy.

Three-way branching gives the PP three possible addresses to branch to, depending on a programmed condition code or the VME-busy state of the workstation. Suspect the PP Condition Code Select hardware, the VME Bus Requester hardware, the VME Bus Master Data Transfer Controller and the paths among them.

241 Couldn't move data via FIFO and VME from/to Shared Memory. Exp xxx at %x Obs xxxx at %x

The Shared memory to FIFO to VME and back to shared memory (SFV) test uses much of the graphics processor hardware to accomplish its task of copying data from one part of shared memory to another. It uses the VP to write the pattern to the FIFO, then it uses the PP to move it from the FIFO back to shared memory via the VME bus. So the PP becomes the bus master and shared memory is the bus slave. This test will uncover bugs that other tests miss, but it cannot isolate that fault to a certain area of the hardware. This is where user-feedback can enhance this user guide. When you fix a bug that this test detected,

please use the bug report form at the back of this manual to let us know what you learned.

243 PP Two Address Operation, r[%d] + acc = r[%d], error found.
OBS r[%d] = 0x%x EXP r[%d] = 0x%x

Suspect the PP 29116, the PP 29116 RAM Address Select hardware and the path between them. Also suspect the path for micro-word bit 51 to the PP 29116 RAM Address Select hardware.

246 just tried to clear interrupts via the GP Control Reg, but the GP Status Reg %s says rupts pending.
GP Status Reg: xxxx xxxx xxxx xxxx

This is a test of bit 15 in the GP Status Register. That bit indicates whether the PP microcode tried to generate a VME bus interrupt when they were disabled by the GP Control Register. It should always be cleared after a board reset, but the test detected a 1. Suspect GP Status Register chip UP1, GP Control Register chip UN3, VME Bus Interrupter chip UF4, the reset hardware and the paths among them.

248 VME Status Reg bit 15 says an interrupt is pending.
VME Status Reg: xxxx xxxx xxxx xxxx
(Note that the VME Status Reg uses reverse logic, so leftmost bit 15 should be 1.

This is a test of bit 15 in the PP's VME Status Register. That bit indicates whether the PP microcode tried to generate a VME bus interrupt when they were disabled by the GP Control Register. It should always be cleared after a board reset, but the test detected a 0, not cleared. Suspect VME Status Register chip UN5, GP Control Register chip UN3, VME Bus Interrupter chip UF4 and the paths among them.

249 The PP's VME Status Register Rupt Pending bit, 15, fails to show an interrupt pending. It is supposed to be a copy of GP Status Reg bit 15, Interrupt Pending, which does show a rupt pending.
GP Status Reg: xxxx xxxx xxxx xxxx
VME Status Reg: xxxx xxxx xxxx xxxx
(Note that the VME Status Reg uses reverse logic, so leftmost bit 15 should be 0.

Since the GP Status Register is good, suspect VME Status Register chip UN5 and the path between it and Bus Interrupter chip UF4.

250 GP Status Register: xxxx xxxx xxxx xxxx
PP tried to generate a VME rupt with GP rupts disabled. That should have set GP Status Reg bit 15, Interrupt Pending, but it didn't happen.

Suspect the PP Destination hardware, GP Status Register chip UP1, VME Bus Interrupter chip UF4 and the paths among them. Note whether you also get the following lines of message:

Interestingly, the PP's VME Status Register Rupt Pending bit, 15, shows a rupt pending, and that's supposed to be a copy of GP Status Reg

bit 15, Interrupt Pending, so check on that.
VME Status Reg: xxxx xxxx xxxx xxxx
(Note that the VME Status Reg uses reverse logic, so leftmost bit 15 should be 0.

This narrows down the fault to GP Status Register chip UP1 and the path between it and Bus Interrupter chip UF4.

The PP's VME Status Register Rupt Pending bit, 15, also does not show an interrupt pending.
VME Status Reg: xxxx xxxx xxxx xxxx
(Note that the VME Status Reg uses reverse logic, so leftmost bit 15 should be 0.

These lines tell you to concentrate on the PP Destination hardware and its connection to Bus Interrupter chip UF4.

251 The PP's VME Status Register Rupt Pending bit, 15, fails to show an interrupt pending. It is supposed to be a copy of GP Status Reg bit 15, Interrupt Pending, so check on that.
VME Status Reg: xxxx xxxx xxxx xxxx

This is a test of bit 15 in the PP's VME Status Register. That bit indicates whether the PP microcode tried to generate a VME bus interrupt when they were disabled by the GP Control Register. The test did try to generate an interrupt, but this bit did not go to zero, as it should. Suspect the PP Destination hardware, VME Status Register chip UN5, VME Bus Interrupter chip UF4 and the path among them.

252 Tried to clear the GP Status Reg bit 15, Interrupt Pending, by setting GP Control Register bit 15 Clear Interrupt. It didn't clear.
GP Status Reg: xxxx xxxx xxxx xxxx

Suspect GP Status Register chip UP1, GP Control Register chip UN3, VME Bus Interrupter chip UF4 and the paths among them.

255 Exec couldn't install handler for this interrupt vector: %x (hex).

In order for the diagnostic to test the PP VME Bus Interrupter, it needed an interrupt handler for the expected vector. This message says that it got the wrong response from the diagnostic executive. Try the test again, reload and retry the test, reboot the Exec and try the test. If you still get this error code, contact Sun Customer Support.

256 VME Interrupt test tried to generate an interrupt with this vector id %d (decimal), but got no interrupt at all.
GP Status Reg: xxxx xxxx xxxx xxxx

The test tried to get the GP1 to generate an interrupt over the VME bus, but that did not happen. Suspect the VME Bus Interrupter hardware, the Interrupt ID Register and their connections to the PP bus and to the VME bus.

Notice that GP Status Register bit 15, Interrupt Pending, was set. Since Grappler ENABLED interrupts, why didn't we get one?

If these lines appeared, suspect GP Control Register chip UN3.

257 Exec couldn't remove the interrupt handler.

In order for the diagnostic to test the PP VME Bus Interrupter, it needed an interrupt handler for the expected vector. This message says that it got the wrong response from the diagnostic executive when it tried to remove the interrupt handler. Try the test again, reload and retry the test, reboot the exec and try the test. If you still get this error code, contact Sun Customer Support.

260 Did a VME word write to shared memory on a byte boundary. That should have set VME Status Register bit 4, Illegal Access, but it didn't. VME Status Reg: xxxx xxxx xxxx xxxx

Suspect the VME Control Register hardware, the VME Master Address Register low byte hardware (specifically the lowest bit), the Miscellaneous VME Master Logic hardware, the VME Status Register and the paths among them.

261 Did a VME access to an undefined address modifier. Expected the VME Status Register to show a time-out error, but that didn't happen. This indicates a problem with the VME Timeout Counter.

Suspect the VME Status Register and the path between it and the VME Bus Timeout Counter hardware ("Timeout").

262 Did a VME access to an undefined address modifier. Expected the VME Status Register to show a time-out error, but it didn't. VME Status Reg: xxxx xxxx xxxx xxxx

Suspect the VME Bus Timeout Counter hardware, the VME Status Register and the path between them ("Timeout").

265 Generated vp 29116 negative condition code, but cc select logic didn't pass that signal to 2910A.

Suspect the VP 2910A and the VP Condition Code Select hardware. Also suspect the paths between the VP Instruction Register and the PP Condition Code Select hardware.

267 Generated carry, fifo not full, fifo not empty cc's. The VP. incorrectly detected a 29116 NEGATIVE cc. Check cc select logic.

Suspect the VP 2910A and the VP Condition Code Select hardware.

268 System didn't detect high bit of dreg field (microcode bit 39), programmable negation bit.

Suspect chip UE32 and the path between it and the VP Instruction Register.

269 Generated vp 29116 carry condition code, but cc select logic didn't pass that signal to 2910A.

Suspect the VP 2910A and the VP Condition Code Select hardware. Also suspect the paths between the VP Instruction Register and the PP Condition Code Select

hardware.

270 Generated negative, fifo not full, fifo not empty cc's. The VP incorrectly detected a 29116 CARRY cc. Check cc select logic.

Suspect the VP 2910A and the VP Condition Code Select hardware.

271 Generated vp 29116 zero condition code, but cc select logic didn't pass that signal to 2910A.

Suspect the VP 2910A and the VP Condition Code Select hardware. Also suspect the paths between the VP Instruction Register and the PP Condition Code Select hardware.

272 Generated carry, fifo not full, fifo not empty cc's. The VP incorrectly detected a 29116 ZERO cc. Check cc select logic.

Suspect the VP 2910A and the VP Condition Code Select hardware.

273 Generated vp 29116 overflow condition code, but cc select logic didn't pass that signal to 2910A.

Suspect the VP 2910A and the VP Condition Code Select hardware. Also suspect the paths between the VP Instruction Register and the PP Condition Code Select hardware.

274 Generated carry, fifo not full, fifo not empty cc's. The VP incorrectly detected a 29116 OVERFLOW cc. Check cc select logic.

Suspect the VP 2910A and the VP Condition Code Select hardware.

275 Moved a zero in 29116 with STATUS ENABLED. CC logic behaved as if zero cc weren't set.

Suspect the paths between the VP Instruction Register and the VP Condition Code Select hardware and between the 29116 and the VP Condition Code Select hardware.

276 Moved a zero in 29116 with status enabled; then moved a 1 with STATUS NOT ENABLED. CC logic said zero cc wasn't still set. Check cc select logic.

Suspect the paths between the VP Instruction Register and the VP Condition Code Select hardware and between the 29116 and the VP Condition Code Select hardware.

277 Grappler did a reset, which should have emptied the fifo. The VP 2010A did not detect fifo1 not full. Check cc select logic.

Suspect the reset circuitry and its path to the FIFO Synchronizer hardware and the VP 2910A and the VP Condition Code Select hardware.

278 Grappler filled the FIFO, but 2910A detected 'fifo1 not full' cc. Check cc select logic.

Suspect the FIFO Control hardware and its connections to the cc select logic.

279 Grappler cleared FIFO but the VP 2910A still detected 'fifo2 not empty' cc. Check cc select logic.

Suspect the FIFO Control hardware and its connections to the cc select logic.

280 Grappler wrote a word to FIFO, but 2910A failed to detect 'fifo2 not empty' condition code. Check cc select logic.

Suspect the FIFO Control hardware and its connections to the cc select logic.

281 Grappler forced a negative number in the floating point hardware, but the VP 2910A failed to detect 'fp negative' cc. Check cc select logic.

Suspect the Weitek ALU chip, FP status chip UA33, the cc select logic and the paths among them.

282 Grappler assured positive number in floating point hardware, but the VP 2910A detected the 'fp negative' condition code. Check cc select logic.

Suspect the Weitek ALU chip, FP status chip UA33, the cc select logic and the paths among them.

302 GB DRAM constant pattern test error found.
LOC(high) = 0x%x LOC(low) = 0x%x OBS = 0x%x EXP = 0x%x

The test wrote a constant pattern to GB DRAM and the result did not compare. Suspect the GB DRAM, the PP Bus Extension Transceivers and Decoders, the GP/GB Interface Signal Decoders on the GB and the backplane of the workstation.

304 GB DRAM address unique test error found:
LOC(high) = 0x%x LOC(low) = 0xhhhh OBS = 0xhhhh EXP = 0xhhhh

Suspect the Z-Buffer Address Pointer, Row/Column Address Multiplexer, Address Buffers and the paths among them.

306 GB DRAM refresh logic test error found
LOC(high) = 0x%x LOC(low) = 0xhhhh OBS = 0xhhhh EXP = 0xhhhh

Suspect the GB Row/Column Address Multiplexer and Refresh Address Counter.

308 GB DRAM address test error found.
LOC(high) = 0x%x LOC(low) = 0xhhhh OBS = 0xhhhh EXP = 0xhhhh

Suspect the Z-Buffer Address Pointer, Row/Column Address Multiplexer, Address Buffers and the paths among them.

310 GB DRAM Surround Disturb test error found.
LOC(high) = 0x%x LOC(low) = 0x%x ACT = 0x%x EXP = 0x%x

Suspect the GB DRAM.

312 GB DRAM fill mode test error found.
LOC(high) = 0x0 LOC(low) = 0xhhhh OBS = 0xhhhh EXP = 0xhhhh

Suspect the DRAM and the Fill Mode hardware.

314 GB DRAM RMW mode test error found.

Suspect the DRAM and the Fill Mode hardware.

317 GB Integer Multiplier test error found.
Mode Reg = 0x1e X = 0x%x Y = 0x%x ACT result = 0x%x EXP result = 0x%x

Suspect the Multiplier hardware of the GB.

9.5. Abortion Message Interpretation.

Because the diagnostics use microcode primitives heavily to stimulate the GP1 and GB hardware and carry the results to the workstation, they must abort if something is wrong with the microprocessing mechanism. These messages explain why the test aborted.

Couldn't stop the VP in order to load microcode. Try the slave reg tests.

Couldn't stop the PP in order to load microcode. Try the slave reg tests.

The microstore will not accept microcode from the workstation unless both microprocessors are halted. The test failed to detect the halted condition for the named processor, as shown in the GP Status Register. Suspect the GP Status Register, the GP Control Register and the Run/Halt hardware.

Couldn't start the VP after loading microcode. Try the slave reg tests.

Couldn't start the PP after loading microcode. Try the slave reg tests.

The test could not detect the started condition of the named processor, as shown in the GP Status register. It starts a processor using a bit in the GP Control Register. Suspect the GP Status Register, the GP Control Register and the Run/Halt hardware.

Couldn't initialize the VP status reg prior to loading microcode.

Couldn't initialize the PP status reg prior to loading microcode.

The test always loads and starts a primitive which sets the processor's status register to a known pattern before it loads the requested primitive. This message says that the initializing primitive failed to do its job. Suspect the GP Status Register, the GP Control Register, the Run/Halt hardware and microstore.

Couldn't verify microcode after loading. Try the slave microstore tests.

The test wrote the microcode primitive to microstore, then read it back for verification and found an error. Suspect the microstore RAM and its connection to VME.

Couldn't modify microcode. Try running the slave microstore tests.

The test needed to modify some part of microcode after loading and verifying it. This message said that the modification process failed. Suspect the Microstore Address Register and Data Register.

Couldn't enable interrupts via the GP Control Register, according to the GP Status Register. Try the slave reg tests.

The message is self-explanatory. Suspect the GP Status and Control Registers.

Couldn't disable interrupts via the GP Control Register, according to the GP Status Register. Try the slave reg tests.

The message is self-explanatory. Suspect the GP Status and Control Registers.

Couldn't clear the VME interrupt bit via the GP Control Register, according to the GP Status Register. Try the slave reg tests.

Suspect GP Status Register chip UP1, GP Control Register chip UN3, VME Bus Interrupter chip UF4 and the paths among them.

Wanted the PP test to pass some data via the FIFO, but microcode found the fifo to be full. Try running FIFO test.

The test always tries to clear the FIFO before using it for data transfer by doing a control register reset. Suspect the reset hardware and the FIFO hardware.

VP tried to write data to the FIFO, but unexpectedly got FIFO full signal.

Suspect the FIFO hardware.

CPU tried to write data to the color board as a test setup, but when it read it back, that data wasn't there. Try running the test again; but in any case, don't blame the GP.

In order to test the highest bits of the VME Address Counter, the test needs the color frame buffer board. If the test believes that that board is not installed, it puts out a message and continues. This message says that the test thinks the color board is installed, but it couldn't reliably write to it as a bus slave. The test did not yet involve the GP in this process. Suspect the workstation CPU, the color board or the backplane.

10

Sun-2 and Sun-3 Keyboard Diagnostic

10

Sun-2 and Sun-3 Keyboard Diagnostic

The keyboard test program runs under the SunDiagnostic Executive. It tests the functionality of the keyboard on Sun-2 and Sun-3 systems.

10.1. Requirements

The diagnostic requires a standard Sun monitor or a 1024 x 1024 monitor, and a standard keyboard.

The test takes about 3 minutes, and requires the operator to participate.

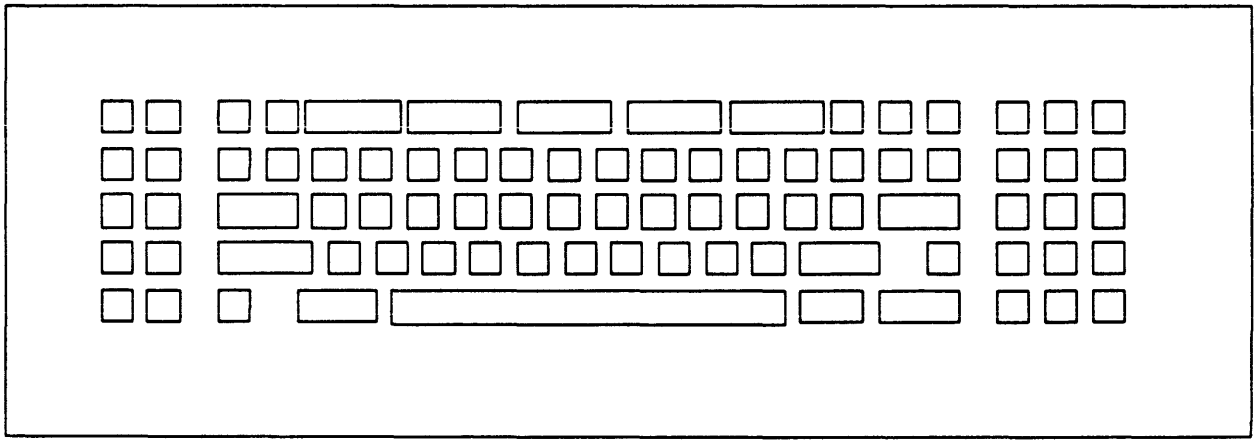
10.2. Description

To use the diagnostic:

Enter **k** from the diagnostics menu.

When the diagnostic starts, it clears the screen, then displays a drawing of a keyboard. The example below depicts the Sun-3 keyboard layout.

Figure 10-1 *Sun-3 Keyboard Display*



Starting at the upper left-hand corner of the keyboard, press every key in sequence, going from left to right. Start with `L1` on the left, then type across to `R3` on the right.

As you press each key, the image of that key on the screen should change from solid to "hash-marked", while the idle indicator (the square in the upper left corner) should disappear. When you release the key, the image of that key should turn white and the idle indicator should reappear.

Continue this sequence from left to right on each row, working from the top row down to the bottom.

If you press a key out of sequence, the beeper sounds, the idle indicator disappears, and the image of the key changes from dark to striped. When you release it, the image returns to dark and the idle indicator reappears. Press the correct key (the next dark one in the display) to continue the test.

If you want to discontinue key testing, press the Control key and the **C** key simultaneously (Control-C). The keyboard test will abort and you may then perform the click test (if applicable).

NOTE *Key click is a feature of Sun-3 workstations only.*

After you press the last key, this message (for Sun-2's) appears:

```
Audio Annunciator Test: BEEP! BEEP! BEEP!
Keyboard Test Complete
```

Or, for Sun-3's, this display appears, and the bell should sound three times:

```
Audio Annunciator Test: BEEP! BEEP! BEEP!
Key Click Test: Type keys and check for click, <ESC> to continue to next test
(type keys and listen for click)
Key No-Click Test: Type keys and check for NO click, <ESC> to continue to next test
(type keys and listen for click)
Keyboard Test Complete
```

NOTE *Keyboard click can be disabled through an EEPROM parameter entry (refer to the PROM User's Manual for more information.)*

The following signs indicate a damaged keyboard:

- The diagnostic displays an error message such as

  ```
  KB DETECTED ERROR
  ```

  or

  ```
  UNKNOWN KEYCODE ERROR
  ```

- The idle indicator does not appear in the upper left corner of the display.
- Any key on the display fails to change from dark to striped to clear when you press it in the correct sequence.
- The bell does not sound.

If the test fails to acknowledge a key, and you are unable to continue the test, abort it by entering a Control-C (press the Control key and hold it down while pressing the c key) from an alternate Exec console, or by cycling the power on the test system.

If the keyboard proves defective, replace it.

CAUTION **To avoid damage to components, power-down the system before installing or removing a keyboard.**

11

MCP/ALM2 Diagnostic

11

MCP/ALM2 Diagnostic

11.1. Introduction

The Sun Multiprotocol Communications Processor has circuitry that provides up to 16 serial ports and one parallel port on a Sun workstation. Twelve of the serial ports are asynchronous only and four of them are either asynchronous or synchronous; two of those four support baud rates up to 230K, full duplex. Two configurations of this board are available: one with four synchronous serial ports and no printer port, called MCP, and the other with sixteen asynchronous serial ports and one parallel printer port, called the ALM-2.

11.2. Hardware Requirements

To run this diagnostic, you need a Sun workstation with at least one Sun ALM-2/MCP installed. Certain of the tests need loopback connectors fitted to the ports.

You don't need (or want) any external device hooked up to the ALM-2/MCP board while running these tests.

Loopback Connectors

A diagnostic program must test as many of a board's components as possible, which means, in the case of this Multi-Port Communication board, testing all of the port line drivers, receivers and handshaking circuits. One way to do this is to have some programmed processors attached to those ports, behaving in an expected way when the diagnostic tries to communicate. A simpler and less expensive way is to connect the ports so that the data going out a port comes right back into it, or to connect one port to another one, which is what loopback connectors do.

For manufacturing Final Test, an entire workstation is configured with the parts that the customer ordered. In that case, the loopback configuration will be pairs of RS-232/449 connectors or single connectors in the sockets provided with the board.

All of the following figures show the pins as they are on the external DB25 or DB50 sockets, which are the ones that the customer uses. The ALM2 and MCP boards are functionally similar, but have different connectors, as indicated in this text.

The ribbon connector on the ALM-2 board that provides RS232 signals is called J6. Note that, due to the fact that the MCP board has synchronous ports only, these signals are not available on that board. The figure below shows the RS232 signals on the asynchronous-only ports, with their DB25 pin numbers. *From Pin* and *To Pin* indicates which output signals the diagnostic expects to be looped

back to which inputs, in either a single loopback or loopback pair fixture.

Figure 11-1 *RS-232 Loopback Signals, Asynchronous-only Ports*

Signal	*From Pin*	*To Pin*	*Signal*
Chassis Ground	1		
Transmitted Data	2	3	Received Data
Data Terminal Ready	20	8	Data Carrier Detect
Signal Ground	7		

The figure below shows the RS232 signals on the synchronous-only ports with their DB25 pin numbers. The ribbon connector that provides these signals is J5 on the ALM2 board.

Figure 11-2 *RS-232 Loopback Signals, Synchronous Ports*

Signal	*From Pin*	*To Pin*	*Signal*
Chassis Ground	1		
Transmitted Data	2	3	Received Data
Received Data Clock	17	24	Transmitted Data Clock Out
Data Terminal Ready	20	8	Data Carrier Detect
Data Carrier Detect	8	6	Data Set Ready
Request To Send	4	5	Clear To Send
Signal Ground	7		

The following figure shows the RS232 signals on synchronous/asynchronous port 0 with the DB50 pin numbers. Ports 0 through 3 are synchronous/asynchronous ports on the MCP board. Please note that although this illustration seems to indicate a single port looping back to itself, you may also use these signals in paired cables. On the MCP version of the board, synchronous RS232 signals are available on ribbon cable connectors J9 and J11.

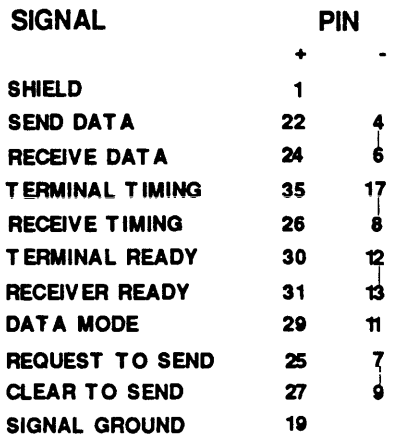
Figure 11-3 *RS232 Loopback Signals, Synchronous/Asynchronous Ports*

Signal	*From Pin*	*To Pin*	*Signal*
Transmitted Data	26	2	Received Data
Received Data Clock	18	1 (on J4)	Transmitted Data Clock Out
Data Terminal Ready	3	27	Data Carrier Detect
Data Carrier Detect	27	17	Data Set Ready
Request to Send	16	40	Clear to Send

Note that, on the MCP board, channels 0 and 1 may be configured as either RS232 or RS449 ports. Either jumper J1501 or J1502 is installed, depending on which type of signal is selected. (They are labeled accordingly.) This diagnostic requires the MCP board to be jumpered as RS449 and the ALM2 board to be jumpered for RS232 operation. The figure below shows the RS449 signals on the two possible MCP RS449 synchronous ports, with their DB37 differential paired pin numbers. On the ALM2 board, these signals are provided through ribbon

cable connector J4. On the MCP board, RS449 signals are available on J8 and J10.

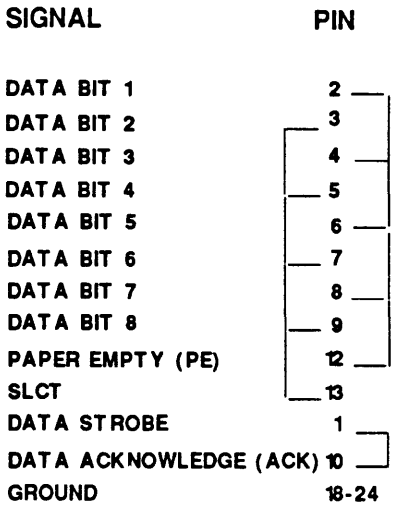
Figure 11-4 *RS449 Loopback Signals, The Two RS449 Synchronous Ports*



The next figure shows the parallel printer port signals on the DB25 plug, with their loopback connections for the Manufacturing, non-printer environment. It connects all of the even-numbered data outputs to the PE status input and all the odd-numbered data outputs to the SLCT status input. This will cover a case where two adjacent pins are shorted together.

Note that the printer port is available only on the ALM2 board. The printer connector on the ALM2 board is J7.

Figure 11-5 *Parallel Printer Port Signals, DB25 Plug With Loopback*



11.3. Limitations

This diagnostic does not name failing parts of the board. It displays actual and expected values during each test. Refer to the section showing error messages for tips on how to interpret them and locate hardware faults.

11.4. Operating Instructions

Read Chapter 2 for information on how to start up the SunDiagnostic Executive.

Loading And Starting

From the Diagnostics menu of the Executive, select the MCP/ALM2 diagnostic. The Exec will load it and display its first menu.

11.5. The User Interface

The user interface consists mostly of menus, and sometimes the program asks you some simple questions. Use the command line language outlined in Chapter 2 to interact with the MCP/ALM2 diagnostic.

Recommended Test Procedure

If you are testing a fresh ALM-2/MCP that was never used or tested before, load this diagnostic and select each test from each of the three menus. Watch for error messages.

If you are confident that the board is good and just want to verify that fact, install the board into a workstation, install the loopback fixtures, load the MCP/ALM2 diagnostic and select these tests:

Common RAM.
Line Drivers and Receivers.
DTR and DCD.
RTS and CTS.
Printer Port Loopback.
Baud Rate Accuracy.
X-Off Function.
SCC Interrupt Vectors.
CIO EOP Interrupt Vectors.
CIO DSR Interrupt Vectors.
Multiport.

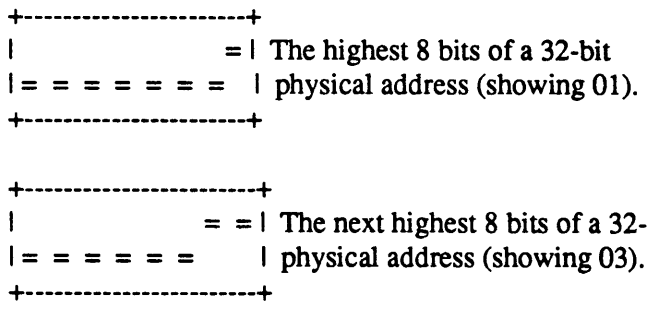
To run every test repeatedly, select the Exec's Options menu, then choose a `Pass=` *number* from that menu. Then select the Diagnostic menu and the MCP/ALM2 diagnostic. Select the ALM-2 or MCP board and use `DLF` to describe the loopback fixtures. Now select `DEF` from the basic test menu. That will run every MCP/ALM2 test for as many times as you selected in the Exec `Pass=` option.

The Main Menu

The first menu you see after choosing the MCP/ALM2 diagnostic from the Executive presents a choice of four possible boards to test, showing their physical addresses as expected by the diagnostic. This menu also lets you choose whether or not the test should halt when it finds an error. An example of the main menu is shown below:

```
Hoe     Halt On Error.              [current]
GOe     Go On Error (don't halt).
A       Test Board 0.  Phys address hex 01000000
B       Test Board 1.  Phys address hex 01010000
C       Test Board 2.  Phys address hex 01020000
D       Test Board 3.  Phys address hex 01030000
Va      Show Virtual Addrsses.
BA=     Baud Rate (300, 1200, 2400, 9600, 19200, 38400). [currently 38400]
```

Please note that you must set DIP switches on the ALM-2/MCP board(s) you are testing to match the physical address(es) in this menu (refer to the following figure).



The previous example shows the DIP switches set for physical address 0x01030000.

NOTE *Note that you are looking at the ALM-2/MCP board "upside-down," with the three VMEbus connectors on your left and the ports on your right.*

The `Show Virtual Addresses` selection is intended for use in debugging. It tells you what virtual address the diagnostic executive has given to certain addressable parts of the ALM-2/MCP.

The `Baud Rate Selector` enables you to force the diagnostic to use a baud rate different than the pre-selected default in the program, which is 38,400 characters per second.

The Basic Test Menu

After you have selected one of the physical boards to test, the diagnostic will try to open it for testing. It it succeeded, the test will display a menu such as this:

```
                Testing Board 0 at 38400 baud.

        M           Manufacturing Board Test Defaults.
        DLF         Describe Loopback Fixtures.
        DEF         Default Tests.  Enter "?" for the list.
        Dev         DEVCTL Register Test.
        Int         Interrupt Vector Register Test.
        Ram         Common RAM Test.
        Xof         X-Off File Test.
        Fif         FIFO Test.
        DMa         DMA Chip Addressing, Data Test.
        Scc         SCC Chip Addressing, Data Test.
        SL          Scope Loops.
        MT          More Tests.
```

Test Menu SL Option

Choosing the `SL` option allows you to run scope loops. The tests under this option are designed so that the software performs the requested test within an infinite loop to facilitate hardware debugging. The only way to exit the infinite loop is to press Control-C.

Test Menu DLF Option

Because the diagnostic must be flexible enough to run under variable circumstances, it allows you to tell it how the ports are set up. Choose `DLF` from the basic test menu to give the diagnostic the details on how you have the board under test configured. Selecting `DLF` display the menu shown below:

```
Please choose a loopback description for the board.

M           Manufacturing Defaults (MCP + ALM-2, printer plug).
Pairs       8 ALM pair cables.
All         16 ALM single plugs.
MDf         MCP Pairs with RS449 types on ports 0-1.
MDT         MCP Pairs with RS232 types on ports 0-1.
MPf         MCP Plugs with RS449 types on ports 0,1.
MPT         MCP Plugs with RS232 types on ports 0,1.
None        only use internal loopback tests.
MIxed       prompt me for details.
RP          A Real Printer is plugged in (default, except under M).
PL          A Printer Loopback plug is in.
S           Show loopback Configuration.
```

M **M** selects a default configuration to be used in manufacturing to test the ALM-2/MCP. It has all 16 RS232 ports paired, with port 0 paired with port 1, port 2 paired with port 3, and continuing that way for all ports. It also simultaneously pairs the two ports 0 and 1 as RS449 ports. The RS449 signal drivers and receivers are separate on the board from the ones for RS232, making this possible.

The Manufacturing default also tells the diagnostic that the parallel printer port has its special loopback connector installed in place of a real printer.

`Pairs` and `All`

`Pairs` and `All` are two ways to configure the board with asynchronous-only loopbacks. That is why they are called ALM configurations, rather than MCP.

The four MCP choices allow you to tell the diagnostic that the board under test is configured as an MCP; i.e. it has only four ports, which are intended to be synchronous, but which can also be asynchronous. The diagnostic knows that only those four ports can have any kind of loopback fixture, and your selection from this menu tells it which kind.

The Middle Test Menu

If you select item **`MT`** from the Basic Menu, the diagnostic will display the Middle Menu, shown below.

```
                Testing Board 0 at 38400 baud.

A       Async Data Flow Test. (Uses internal loopback.)
S       Sync Data Flow Test. (Uses internal loopback.)
D       DMA Addressing Test. (Uses internal loopback.)
L       Line Drivers, Receivers Test. (Needs loopback.)
DT      DTR And DCD Control Test. (Needs loopback.)
R       RTS And CTS Control Test. (Needs loopback.)
RP      Real Printer Test.  (Needs a real printer.)
PL      Printer port Loopback Test.  (Needs printer port loopback plug.)
MT      More Tests.
```

The Advanced Test Menu

If you select item **`MT`** from the Middle Menu, the diagnostic displays the Advanced Menu, shown below.

```
              Testing Board 0 at 38400 baud.

B       Baud Rate Accuracy Test. (Uses internal loopback.)
X       X-Off Function Test. (Uses internal loopback.)
S       SCC Interrupt Vectors Test. (Uses internal loopback.)
F       CIO FIFO Interrupt Vectors Test. (Uses internal loopback.)
E       CIO EOP Interrupt Vectors Test. (Uses internal loopback.)
D       CIO DSR Interrupt Vectors Test. (Needs MCP loopback fixtures.)
V       VME Interrupts from SCC, CIO.  (Uses internal loopback.)
M       Multiport Test. (Uses internal loopback.)
```

11.6. Error Handling

The main menu of the ALM-2/MCP diagnostic gives you a chance to tell it whether you want it to stop testing when it finds a hardware fault. The default is to keep going. If you do not halt on error, you will probably see more error messages. These might help you determine where the problem lies, but it could also add confusion. Later tests expect the hardware tested earlier to be good, so their messages could mislead you to the wrong area of the board.

11.7. Message Interpretation And Failure Analysis.

Messages from the ALM-2/MCP diagnostic have a number at the beginning, which helps you find their descriptions in this text.

A percent (%) sign in the examples below indicates how the value will be presented under running conditions.

```
%d  a decimal value (0's - 9's).
%x  a hexadecimal value (0's - f's).
%b  a binary value (0's and 1's).
```

Following is a list of possible test messages and their meaning:

2 Tried to set all bits of port p DEVCTL reg. Obs: %b Exp: %b

The test tried to set every bit in the given port's DEVCTL register to a 1, but one or more showed a 0. (The DEVCTL hardware is on sheet 5 of the ALM-2/MCP Schematics. It is not organized as you might expect, with one register for each port. Instead, it is organized by the functions of the individual bits.)

3 DEVCTL reg port p Observed %b Expected %b

A walking 1's test tried to write a single 1 to the DEVCTL register and a zero to the others. The test showed each bit of the DEVCTL register for the given port, where one or more of the bits was wrong.

4 DEVCTL reg port p Observed %b Expected %b

Same as above, only the test was a walking zeros test.

9 Interrupt Vector Register Observed %b Expected %b

The Interrupt Vector Register test tried a bit pattern that mis-compared. The Ivec Reg is on sheet 5 of the schematics.

10 Common RAM byte address test. Address %x Observed %x Expected %x

Software can address the ALM-2/MCP's common RAM as bytes, 16-bit words or 32-bit words. This message says that a byte-addressing test, using address-unique patterns, found a mis-compare. An addressing failure could be a failing chip or broken or shorted address lines to a common RAM chip.

11 Common RAM 16-bit word address test.
Address %x Observed %x Expected %x

A 16-bit word-addressing test, using address-unique patterns, found a mis-compare. An addressing failure could be a failing chip or broken or shorted address lines to a common RAM chip.

12 Common RAM 32-bit word address test.
Address %x Observed %x Expected %x

A 32-bit word-addressing test, using address-unique patterns, found a miscompare. An addressing failure could be a failing chip or broken or shorted address lines to a common RAM chip.

13 Common RAM pattern Read-After-Write.
Address %x Observed %x Expected %x

This message came from a test that wrote a pattern to a common RAM address and immediately read it back before going on to the next address. This test will catch a RAM chip that is slow to store a pattern for correct readback.

15 X-Off File Addressing. Port %d Observed %x Expected %x

The X-Off File holds one byte per ALM-2/MCP port. The message above came from an address-unique pattern test. X-Off hardware is on sheet 7 of the schematics.

16 X-Off File Zero Data. Port %d Observed %x Expected 0

The test tried to write zeros to every address of the X-Off file, but the location for the named port showed a wrong pattern.

17 X-Off File Walk-1. Port %d Observed %b Expected %b

The diagnostic found a problem with a location in the X-Off file during a walking-1's test.

18 X-Off File Walk-0. Port %d Observed %b Expected %b

The diagnostic found a problem with a location in the X-Off file during a walking-0's test.

19 Board reset failed to clear FIFO. FIFO word 0x%x Obs %x Exp ffff

After signaling a software reset to the ALM-2/MCP, a reading of the FIFO address should result in sixteen 1's (ffff). The test received some other result, as shown in the message. Suspect the FIFO controller, shown on schematic sheet 7.

20 FIFO Write/Read test. Observed %x Expected %x

The test tried writing a pattern to the FIFO then immediately reading it back. Suspect either the FIFO controller, shown on schematic schematic sheet 7, or the FIFO itself, which is in some of the common RAM chips on sheet 9.

21 FIFO Addressing Test. FIFO word %x Obs %x Exp %x

The test wrote unique patterns into every location of the FIFO, then began reading the FIFO. The miscompare could be due to a bad common RAM chip, the FIFO controller, the common RAM decoder on sheet 4, or, perhaps, a bad address line.

22 FIFO Pattern Test. FIFO word %x Obs %x Exp %x

This test tried to find pattern-sensitive faults of the FIFO memory (common RAM).

24 FIFO Random Test. FIFO word %x Obs %x Exp %x

This test used random patterns to detect FIFO faults.

25 DMA Chip Addressing. Chip %d Channel %d Obs %x Exp %x

The test addressed the DMA chips, writing a unique pattern to each one. When it tried to read them back, it got a wrong value. Suspect the VME addressing hardware on schematic sheet 2, the device decoding hardware on sheet 4 or the chip itself.

27 DMA Chip Data. Chip %d Observed %x Expected %x

The test went to each DMA chip and tested it for holding all possible patterns. Suspect the chip or the data lines.

29 SCC Chip Address. Port %d Observed %x Expected %x

The test addressed the SCC chips, writing a unique pattern to each one. When it tried to read them back, it got a wrong value. Suspect the VME addressing hardware on sheet 2, the device decoding hardware on sheet 4 or the chip itself. Each chip is responsible for two ports, with chip channel A being the odd-numbered port and chip channel B being the even-numbered port. So, for example, chip 0's channel B is for port 0 and channel A is port 1.

30 SCC Chip Data. Port %d Observed %x Expected %x

The test went to each SCC chip and tested it for holding all possible patterns. Suspect the chip or the data lines. Each chip is responsible for two ports, with chip channel A being the odd-numbered port and chip channel B being the even-numbered port. So, for example, chip 0's channel B is for port 0 and channel A is port 1.

50 SCC Async Data Test Setup. FIFO Data: %x Expected ffff.

The test tried to empty the FIFO by reading its address more times than its size. Then, after another FIFO read, it did not get the expected 16 1's result.

**51 SCC Async Data. Read %d (dec) bytes, then FIFO showed empty.
Internal loopback on port %d.**

**52 SCC Async Data. Observed port in FIFO: %d Port actually used: %d
Count of FIFO reads (hex): %x**

When software reads the 16 bits at the FIFO address, the low byte is a byte received by one of the serial ports. The high byte is supposed to show the number of the port that received that byte. Error message 52 says that the port number in the high byte is wrong. Suspect the async receive controllers on schematic sheet 6 or the FIFO port address buffers on sheet 7.

53 SCC Async Data. Observed %x Expected %x; FIFO read count (hex) %x.

This messages says that the received data in the FIFO does not match the transmitted data. Suspect the appropriate SCC chip, the FIFO (common) RAM or the data path between them.

55 Sync Data Port %d. Timeout before EOP from DMA controller.

The test waits for a certain amount of time for a block of data to move to common RAM. This says that the synchronous receiver's DMA controller did not get to its final byte count in the time allowed. Suspect the SCC, the transmitter DMA controller, the receiver DMA controller and the paths among them. Of course if the test is set to continue on error, you will also get the next message showing a data miscomparison.

56 SCC Sync Data Port %d. Rcv CRAM Addr %x Observed %x Expected %x

The Synchronous Data test moved data from the receiving SCC into common RAM, then compared with expected. Suspect the appropriate SCC chip the common RAM and the data path between them.

60 DMA Addressing Test Setup. FIFO Data: %x Expected ffff.

The test tried to empty the FIFO by reading its address more times than its size. Then, after another FIFO read, it did not get the expected 16 1's result.

61 DMA Addressing. Read %d (dec) bytes, then FIFO showed empty. Internal loopback on port %d

The test was reading the FIFO in order to check for correct data, when it unexpectedly got the FIFO Empty signal from the FIFO controller. Suspect the FIFO controller, but it could be bad clocking in the SCC's, too.

62 DMA Addressing. Observed port in FIFO: %d Port actually used: %d

When software reads the 16 bits at the FIFO address, the low byte is a byte received by one of the serial ports. The high byte is supposed to show the number of the port that received that byte. The message says that the port number in the high byte is wrong. Suspect the async receive controllers on schematic sheet 6 or the FIFO port address buffers on sheet 7.

63 DMA Addressing. Port %d byte %x Observed %x Expected %x

The test used different data in each address of common RAM, to verify that the port's DMA controller can correctly address it. Suspect the port's DMA controller chip, common RAM and the addressing lines between them.

64 DMA Addressing, port %d. Timeout before EOP from DMA controller.

The test waits for a certain amount of time for a block of data to move to common RAM. This says that the synchronous receiver's DMA controller did not get to its final byte count in the time allowed. Suspect the SCC, the transmitter DMA controller, the receiver DMA controller and the paths among them. Of course, if the test is set to continue on error, you will also get the next message showing a data miscomparison.

**65 DMA Addressing, sync receive data, Port %d.
Address %x Observed %x Expected %x**

This message shows the address in common RAM to which the receiving SCC, via DMA, wrote the observed byte. Suspect the receiving DMA controller, if the transmitting DMA controller for the same port worked in the async test, which already ran.

70 Line Driver & Receiver Test Setup. FIFO Data: %x Expected ffff.

The test tried to empty the FIFO by reading its address more times than its size. Then, after another FIFO read, it did not get the expected 16 1's result.

71 Line Driver/Receiver. Unexpected FIFO Empty pattern. Transmitting port %d

The test was reading the FIFO for received data, expecting it to come from the named port. Instead it got the FIFO empty reading, sixteen 1's. Suspect the port's SCC, its transmit DMA controller, and the FIFO port address buffers.

72 Line Driver/Receiver. Observed port in FIFO: %d Port actually used: %d

When software reads the 16 bits at the FIFO address, the low byte is a byte received by one of the serial ports. The high byte is supposed to show the number of the port that received that byte. This says that the port number in the high byte is wrong. Suspect the async receive controllers on sheet 6 or the FIFO port address buffers on sheet 7.

73 Line Driver/Receiver. Xmit port %x Rcv port %x Observed %x Expected %x

If the same port passed the DMA Addressing test, this message indicates either a bad line driver of the transmitting port or a bad line receiver of the receiving port (depending on your loopback configuration, these could be the same port).

74 Line Drive & Receive, port %d. Timeout before EOP from DMA controller. (Using the port's RSXXX hardware.)

The test waits for a certain amount of time for a block of data to move to common RAM. This message says that the synchronous receiver's DMA controller did not get to its final byte count in the time allowed. Suspect the SCC, the transmitter DMA controller, the receiver DMA controller and the paths among them. Of course, if the test is set to continue on error, you will also get the next message showing a data miscomparison.

The line about the RSXXX hardware only shows up for messages concerning port 0 or 1, which could have either RS232 or RS449. All of the other ports have RS232 hardware.

75 Line Drive & Receiver Sync data, sending port %d receiving port %d. Address %x Observed %x Expected %x
(Using the port's RSXXX hardware.)

If the same port passed the DMA Addressing test in sync mode, this message indicates either a bad line driver of the transmitting port or a bad line receiver of the receiving port (depending on your loopback configuration, these could be the same port).

The line about the RSXXX hardware only shows up for messages concerning port 0 or 1, which could have either RS232 or RS449. All of the other ports have RS232 hardware.

76 The port(s) not tested (no loopback):

You get this message if any port did not have a loopback fixture on it.

80 DTR/DCD Control Test. Port %x sent DTR but port %x didn't see DCD.

This handshake signal test indicates either the transmitting port's DTR line driver is bad or the receiving port's DCD receiver is bad. Depending on your loopback fixture, the two ports could be the same.

81 DTR/DCD Control Test. Port %x turned off DTR but port %x still saw DCD.

The DEVCTL hardware that controls DTR could be bad, the DTR line driver or DCD receiver could be bad, or the receiving port's SCC could be bad. Bit first, suspect the path between the DCD input line and the receiving port's SCC input.

90 RTS/CTS Control Test. Port %x sent RTS but port %x didn't see CTS. Port was tested with RSXXX hardware.

Another handshake signal test. The ALM-2/MCP design only implements these signals on the synchronous ports, 0-3. So make sure you have the suitable loopback plug or pair on those ports. An async-only loopback plug or pair cable will not carry this signal. The transmitting port's SCC sends RTS, the receiving port's SCC receives CTS. Check the SCC's, the RTS driver and the CTS receiver.

The line about RSXXX hardware only shows up for messages concerning port 0 or 1, which could have either RS232 or RS449. Ports 2 and 3 have RS232 hardware.

91 RTS/CTS Control Test. Port %x turned off RTS but port %x still saw CTS. Port was tested with RSXXX hardware.

See the hint about a suitable loopback fixture under 90, above. Maybe a path between the CTS receiver and the receiving SCC is open or shorted to something. The RTS output path could be shorted, too.

The line about RSXXX hardware only shows up for messages concerning port 0 or 1, which could have either RS232 or RS449. Ports 2 and 3 have RS232 hardware.

92 The synchronous port(s) not tested (no loopback):

If you had told the diagnostic, through the DLF option on the first test menu, that some port(s) did not have any loopback fixture, this message reminds you that the test did not cover some hardware.

100 Real Printer test can't work unless you have a real printer. Use the DLF option to notify me that you have one installed.

The Real Printer test gave this message because it believes that you do not have a real printer plugged into the parallel printer port. Go back to the basic menu and select DLF. Then choose the RP option, so that the software will understand that you really do have a real printer plugged in.

101 Real Printer Test detects Paper Empty signal in the CIO. Aborted.

The test detected the Paper Empty signal on the parallel printer port. Either the printer is missing paper, the paper was installed badly, the printer is not turned on, the printer cable is broken or the ALM-2/MCP hardware is bad. Among the ALM-2/MCP hardware, suspect chip u709 or the CIO or the path between them and to the printer port.

102 Real Printer Test doesn't see SLCT signal in the CIO. Aborted.

The SLCT signal comes from the real printer when you "select" the printer. Typically, the button is labeled "On Line." See the discussion for the Paper Empty signal, above.

110 Printer Port Loopback test can't work unless you have a loopback plug on that port. Use the DLF option to notify me that you have one installed.

You have selected the Printer Loopback test, but the diagnostic believes that you do not have a looopback plugged into the parallel printer port.

111 Printer Port Loopback test. Asserted all 1's on data lines but failed to get expected PE status signal.

The printer loopback test relies on the special loopback plug doing things with the data lines and the status lines. See the section on loopback fixtures at the beginning of this chapter. This message says that, with all 1's on the data lines, the CIO should have detected the PE signal.

112 Printer Port Loopback test. Asserted all 1's on data lines but failed to get expected SLCT status signal.
The printer loopback test relies on the special loopback plug doing things with the data lines and the status lines. See the section on loopback fixtures later in this document. This messages says that, with all 1's on the data lines, the CIO should have detected the SLCT signal.

113 Printer Port Loopback test. Sent pattern %x; got unexpected PE.

Because of the nature of the printer loopback plug and the status detection hardware, the given pattern should not have caused the CIO to detect the PE signal, but that did happen. Suspect one of the printer data lines, the printer data buffer u711, the printer data line drivers u712 or u713, u709 or the CIO chip.

114 Printer Port Loopback test. Sent pattern %x; failed to detect SLCT.

Because of the nature of the printer loopback plug and the status detection hardware, the given pattern should not have caused the CIO to detect the SLCT signal, but that did happen. Suspect one of the printer data lines, the printer data buffer u711, the printer data line drivers u712 or u713, u709 or the CIO chip.

115 Printer Port Loopback test. Sent pattern %x; got unexpected SLCT.

Because of the nature of the printer loopback plug and the status detection hardware, the given pattern should not have caused the CIO to detect the SLCT signal, but that did happen. Suspect one of the printer data lines, the printer data buffer u711, the printer data line drivers u712 or u713, u709 or the CIO chip.

116 Printer Port Loopback test. Sent pattern %x; failed to get PE.

Because of the nature of the printer loopback plug and the status detection hardware, the given pattern should not have caused the CIO to detect the PE signal, but that did happen. Suspect one of the printer data lines, the printer data buffer u711, the printer data line drivers u712 or u713, u709 or the CIO chip.

200 SCC Interrupt Vector Test. After mcp reset, DEVVEC = %x; expected ff.

After a reset of the ALM-2/MCP board, a reading of the address of DEVVEC should get eight 1's (hex ff). This test found something else, as reported. DEVVEC comes form the CIO and every SCC. One of those chips failed to reset properly or software cannot properly read the DEVVEC location. Suspect the SCC and CIO chips and u405, the device decode2 PAL, as well as the paths among them.

201 SCC Interrupt Vector Test. Exp vector %x from chip %d; obs %x on DEVVEC

The test tried to cause the named SCC chip to generate a vector for DEVVEC, but it read a different vector at DEVVEC's address. If the observed vector is ff, you will see one of these follow-up lines:

Check the IEI/IEO connection between chip %d and chip %d.
or
Check that the IEI input of this chip is pulled up.

Priority is determined by the daisy chained Interrupt Enable signal going from the IEO output of one chip to the IEI input of the next one. Chip 0 has the highest priority, going down to chip 7 the lowest of the SCC chips, and the CIO chip lower still. If that daisy chain is broken, every chip after the break in the daisy chain would not be able to put its vector out to DEVVEC, so the observed vector would be ff, the sign of no vector.

The vector is supposed to equal the SCC chip number. So if the observed number is higher than the expected number, that means the named chip did not generate its vector at all. Suspect that SCC chip. If the observed vector is aa, bb or cc, the vector came from the CIO, whose vectors were supposed to be disabled. You should never see such a vector, but if you do the named chip should be the highest SCC chip. Suspect it and the CIO both.

210 FIFO Signal Test. After a board reset, CIO fails to show FIFO empty.

The test expected a FIFO empty signal from the CIO after an ALM-2/MCP reset. So suspect either the FIFO controller, the CIO or the path between them.

211 FIFO Signal Test. Incorrect vector after FIFO not empty condition. Observed vector %x Expected vector %x.
The test installed the expected vector in the CIO port A, but DEVVEC showed a different value. Suspect the FIFO controller, the CIO chip and the paths between them.

212 FIFO Signal Test. Moved %d bytes into the FIFO but CIO still says empty.

Suspect the FIFO controller, the CIO chip and the paths between them.

213 FIFO Signal Test moved %d words into the FIFO, which should not generate an interrupt, but DEVVEC showed this vector: %x

The test moved some words into the FIFO, but not enough to cause it to be half-full. Nevertheless, the CIO did try to generate some kind of interrupt. See the section titled *Status in DEVVEC Patterns* if you want to interpret the vector. Suspect the FIFO controller, the CIO chip and the paths between them.

214 FIFO Signals. Move bytes to FIFO till half full. Obs %d bytes, Exp %d

The test thought it knew how many bytes it would take to make the FIFO half full, but the CIO generated a DEVVEC interrupt vector unexpectedly. If the vector is the one expected for the half-full condition, the test puts out this follow-up line:

Vector was correct for half-full condition.

Suspect the FIFO controller for the incorrect number of bytes causing the FIFO-Half-Full condition. See the section titled *Status in DEVVEC Patterns* if you want to interpret the vector.

If the vector was not the one for the half-full condition, the test puts out this follow-up line:

Observed vector %x Expected FIFO-half-full vector %x.

A different vector means something besides the FIFO-Half-Full condition caused the CIO or an SCC chip to put out the vector. The CIO has three "ports," A, B and C. If the vector was from the CIO, the high byte of the vector will be one of those letters. If the vector is from an SCC, it will be the SCC chip number. Suspect that chip. See the section called *Status in DEVVEC Patterns* if you want to interpret the vector.

215 FIFO Signal Test. Incorrect vector after FIFO half full condition. Observed vector %x Expected vector %x.

This message means that the test moved the number of bytes into the FIFO necessary to make it half full, but the resulting vector in DEVVEC was not the expected one. The CIO has three "ports," A, B and C. If the vector was from the CIO, the high byte of the vector will be one of those letters. If the vector is from an SCC, it will be the SCC chip number. Suspect that chip. See the section called *Status in DEVVEC Patterns* if you want to interpret the vector with more detail.

216 FIFO Signals. Moved bytes to FIFO till full. Obs %d bytes, Exp %d.

The test moved 16-bit words into the FIFO, checking the CIO for the FIFO Full signal. This message said that the CIO indicated that the FIFO showed full before the test finished moving the expected number of bytes into it. Suspect the FIFO controller first, then the CIO.

220 EOP Signal Test. After a board reset, CIO shows a Port B EOP signal. Expected 0 Observed %b.

The test did a reset of the ALM-2/MCP board, then examined the CIO for any EOP indication. The message shows port B of the CIO, with one or more bit unexpectedly set to 1. Suspect the corresponding DMA controller chip, the path between it and the CIO, and the CIO itself.

221 EOP Signal Test. Incorrect vector after dma chip %d EOP condition. Observed vector %x Expected vector %x.

The test caused an EOP condition in the named DMA controller chips, and knew what vector to expect from the CIO. Another vector showed up in DEVVEC. If the high byte of the vector is a B, the port was correct, but the EOP came from the wrong dma controller chip. Suspect the wrong controller chip, the expected dma controller chip, the CIO and the paths among them.

If the high byte of the vector is an A, suspect the CIO. If the vector was something else, it is either stray data from an SCC chip (which should have been reset into a quiet state), or garbage due to improper decoding of the DEVVEC address. The decoding hardware of the ALM-2/MCP is on schematic sheet 4. See the section called *Status in DEVVEC Patterns* if you want to interpret the vector with more detail.

222 EOP Signal Test. Incorrect vector after SDLC receive. Observed vector %x Expected vector %x.

This message came from the test that expected an EOP from the DMA controller, which moves synchronous received data from an SCC chip to common RAM. Instead of the EOP from that controller chip, it got the one shown. Suspect the DMA chip that generated the EOP shown, the receiving DMA chip that failed to send the expected EOP, the CIO and the paths among them. See the section called "Status in DEVVEC Patterns" if you want to interpret the vector with more detail.

226 CIO DSR Signals Test, port %x. Obs vector %x Exp vector %x.

This test tried to generate a unique vector from the CIO. The vector would be caused by receiving a DSR signal on the port connected by your loopback fixture to the named port. The port named is the one that sent out a DTR, which your loopback fixture connects to the receiving DSR, on one of the synchronous ports. The vector was incorrect. Maybe you got the types of loopback fixture mixed up, so that the test expected the DSR from the wrong port. For example, the port really has a single loopback plug but you told the DLF option on the basic menu that the port has a loopback pair.

If that was not the case, suspect the data path which carried the erroneous DSR signal to the CIO, the CIO itself and the path between them.

227 The port(s) not tested (no loopback):

If you had told the diagnostic, through the DLF option on the first test menu, that some port(s) did not have any loopback fixture, this message reminds you that the test did not cover some hardware.

230 VME Interrupt Test IVVEC = %x, expected %x. Aborted.

The VME Interrupt Test depends on good hardware holding the board's interrupt vector. The interrupt vector from the board, during an interrupt, has to correspond to the interrupt handler in the workstation memory. The test was unable to install the correct vector in the ALM-2/MCP IVVEC hardware, so it halted its attempt to test VME interrupting.

231 Failed to install mcp interrupt handler, so skipped VME rupt test.

The test invoked a feature of the diagnostic executive to install an interrupt handler program for the VME Interrupt test, but the exec software gave an indication that it failed to do that properly. Try the test again; if still not successful, reboot the executive and see if that helps.

232 VME Interrupt test failed to get port %x interrupt using 0x%x as vector.

The test tried to cause an interrupt from one of the SCC chips, but it did not receive one before its waiting time ran out. Suspect the SCC chip for the named port or the CIO chip, because it controls whether ALM-2/MCP interrupts are enabled.

233 VME Interrupt test; expected vector: %x observed vector %x.

The test was successful at generating an ALM-2/MCP interrupt, but the vector that it read at the DEVVEC location was not the expected one. Vectors from the SCC chips are their chip numbers. Vectors from the CIO are ax, bx or cx where the x represents a status value and the letter represents the port of the CIO that made the vector. See the section called *Status in DEVVEC Patterns*.

234 Exec couldn't remove the interrupt handler.

The test invoked a feature of the diagnostic executive to remove an interrupt handler program for the VME Interrupt test, but the exec software gave an indication that it failed to do that properly. Try the test again; if still not successful,

reboot the executive and see if that helps, or seek someone who knows about the diagnostic executive or this diagnostic.

240 X-Off Function test port %x. X-Off was disabled but DEVCTL shows transmit not enabled. DEVCTL: %b. Pattern under test: %x.

The test disabled the X-Off function for the named port, which should mean that the hardware will not disable transmitting for that port; nevertheless, transmission was disabled. If you run this test continuing on errors and you get this failure with lots of different patterns, suspect the X-Off File, u705 and the X-Off Comparator, u706 and the Transmit Enable PALs u707 and u708. If it only happens with one pattern, suspect the X-Off Comparator.

241 X-Off Function test port %x. X-Off was enabled with pattern %x but DEVCTL shows transmit still enabled. DEVCTL: %b.

This test enabled the X-Off function for the named port, and sent the pattern shown. That should have disabled transmitting for the port, but the test found transmitting still enabled. Suspect the X-Off File, u705 and the X-Off Comparator, u706 and the Transmit Enable PALs u707 and u708.

242 X-Off Function test port %x. Unexpected no. of characters moved. Observed %d, Expected %d + 1 or 2. X-Off pattern %x.

The design spec of the ALM-2/MCP board states that when the transmitting port receives the X-Off character, it will stop transmitting within 2 characters. This message says that too many or too few characters were moved before X-Off did its job. If the observed number is smaller than the expected number, this is a puzzle that was either caused by a weakness in the diagnostic program or a weakness in the hardware that should have been revealed by an earlier diagnostic test.

If the observed number is greater than the expected number, suspect the X-Off Comparator, u706 and the Transmit Enable PALs u707 and u708.

250 Failed to install mcp interrupt handler, so skipped Multiport test.

The test invoked a feature of the diagnostic executive to install an interrupt handler program for the Multiport test, but the exec software gave an indication that it failed to do that properly. Try the test again; if still not successful, reboot the executive and see if that helps.

251 Multiport Test. Invalid port number in FIFO: %x.

The Multiport test caused all of the ports to move bytes into the FIFO as they received them (from themselves, using internal loopback). When workstation software reads the FIFO, it reads 16-bit words instead of bytes, with the high byte indicating which port received the low byte. When the test did that, it found a number in that high byte that could not be one of the port numbers. Suspect the Async Receive Control hardware shown on sheet 6 of the ALM-2/MCP schematics, the FIFO RAM (really the Common RAM) and the paths among them (especially the SCCID lines).

252 Multiport Test no. of chars received by port %x: %x (hex) exp %x.

The Multiport test caused all of the ports to move a certain number of bytes into the FIFO as they received them (from themselves, using internal loopback). It read 16-bit words instead of bytes, with the high byte indicating which port received the low byte. Using a FIFO-Not-Empty interrupt handler, the diagnostic moved the received characters from the FIFO into workstation memory, separating them according to which port received them. This message said that too many characters went to the save area for the named port. This happened because the high byte of the FIFO word incorrectly named this port as the receiver of characters that it did not receive.

Suspect the Async Receive Control hardware shown on sheet 6 of the ALM-2/MCP schematics, the FIFO RAM (really the Common RAM) and the paths among them (especially the SCCID lines).

253 Multiport Test. Port %x Observed Pattern %x Expected %x. Common RAM virtual address of expected pattern: %x.

The multiport test caused all of the ports to move a certain number of bytes into the FIFO as they received them (from themselves, using internal loopback). Then, using a FIFO-Not-Empty interrupt handler, it moved the received characters from the FIFO into workstation memory, separating them according to which port received them. After all of that movement ended (which the test noticed by checking for the Terminal Count indications of every channel of the transmit DMA controllers), the test compared the patterns in saved memory with their expected values still in Common RAM, from which it transmitted them. The message showed a pattern in Common RAM which didn't get properly received into the FIFO.

Suspect the path between Common RAM and the transmitting port, the port's SCC, the path between the SCC and the FIFO and the FIFO RAM (really part of Common RAM) itself.

254 Exec couldn't remove the interrupt handler.

The Multiport Test received the wrong return indicator from the Exec when it tried to remove its interrupt handler. Try the test again or reboot the Exec.

260 Failed to install mcp interrupt handler, so skipped baudrate test.

The Baud Rate Accuracy Test needs to generate an interrupt after moving a block of data, but the Exec call to install the interrupt handler failed. Try the test again or reboot the Exec.

261 Exec couldn't remove the interrupt handler.

The Baud Rate Accuracy Test received the wrong return indicator from the exec when it tried to remove its interrupt handler. Try the test again or reboot the Exec.

262 Baudrate Test unable to get a time-of-day reading via exec call, so aborted. Return value from call: %d.

The Baud Rate Accuracy Test needed a function from the diagnostic executive, but that function failed. Try the test again, reboot the exec or get in touch with an expert in this diagnostic or the diagnostic executive.

263 Baudrate test didn't get expected device vector. Obs %x exp %x

The Baud Rate Accuracy Test needs to generate an interrupt after moving a block of data, but something unexpected caused the interrupt to happen. See the section called *Status in DEVVEC Patterns* if you want to interpret the vector with more detail.

**264 Baudrate test found inaccurate MCP clock.
Observed data block transfer time %d seconds, %d microseconds.
Expected data block transfer time %d seconds, %d microseconds
(plus or minus %d microseconds).**

The Baud Rate Accuracy Test measured the time it took to transfer a block of data through a port, using internal loopback. This message said that the elapsed time was off by over a second. Because this test uses the workstation time-of-day clock and its interrupt handling, it is unreliable if you run it while running another program using the exec's multitasking capabilities. Try the test a few more times to see if the results are consistent. If they are, suspect the clock crystal on the ALM-2/MCP board or the one on the workstation CPU board. If not, suspect that the CPU is being kept busy by another program.

**265 Baudrate test found MCP clock too fast.
Observed data block transfer time %d seconds, %d microseconds.
Expected data block transfer time %d seconds, %d microseconds
(plus or minus %d microseconds).**

The Baud Rate Accuracy Test measured the time it took to transfer a block of data through a port, using internal loopback. This message says that the elapsed time was too short, indicating that the ALM-2/MCP will be using data transfer rates faster than expected. This could cause incompatibility problems with attached serial devices. Suspect the clock crystal on the ALM-2/MCP board or the one on the workstation CPU board.

**266 Baudrate test found MCP clock too slow.
Observed data block transfer time %d seconds, %d microseconds.
Expected data block transfer time %d seconds, %d microseconds
(plus or minus %d microseconds).**

The Baud Rate Accuracy Test measured the time it took to transfer a block of data through a port, using internal loopback. This message says that the elapsed time was too long, indicating that the ALM-2/MCP will be using data transfer rates slower than expected. This could cause incompatibility problems with attached serial devices. Because this test uses the workstation time-of-day clock and its interrupt handling, it is unreliable if you run it while running another program using the exec's multitasking capabilities. Try the test a few more times to see if the results are consistent. If they are, suspect the clock crystal on the

ALM-2/MCP board or the one on the workstation CPU board. If not, suspect that the CPU is being kept busy by another program.

11.8. Status In DEVVEC Patterns

The following information is to help you interpret the status information that comes from some ALM-2/MCP chips when they try to generate an interrupt. This "vector" information shows up in some of the diagnostic error messages, so this is an attempt to describe their meaning. It is not easy to understand, at first, but understanding this is not necessary to troubleshoot the board. Read more about these status signals in literature on the SCC or CIO chips.

The CPU software (the diagnostic, in this case) acknowledges an interrupt from a chip by reading the address of something called DEVVEC. The "Vector" at that address came from the chip that was generating the interrupt signal.

The chips that can generate an interrupt, and that produce an 8-bit "vector" at the DEVVEC location are the eight SCC chips and the CIO chip. The diagnostic has installed a vector in each of those chips, so that it is unique to that chip. In addition, the diagnostic enables an event called, by the chip maker, `Vector Includes Status` (VIS). When VIS is enabled, the lowest four bits of the chip's vector show a special pattern. The VIS pattern is determined by the type of condition that caused the chip to signal an interrupt and so write its vector to DEVVEC. The diagnostic usually takes advantage of VIS, because it provides information about what has been happening.

Each SCC chip has one vector. That means there is one vector per pair of serial ports, since one SCC supports two serial ports. The CIO chip has three vectors, one for its "A Port," one for its "B Port" and one for its "C Port." (Remember these CIO ports are not ALM-2/MCP serial ports, just channels of the CIO chip.) The diagnostic sets the vectors for the chips as follows:

SCC Chip 0: 00
SCC Chip 1: 10
SCC Chip 2: 20
SCC Chip 3: 30
SCC Chip 4: 40
SCC Chip 5: 50
SCC Chip 6: 60
SCC Chip 7: 70
CIO Port A: a0
CIO Port B: b0
CIO Port C: c0

The SCC chips will modify the lower four bits of their vectors with status information as follows:

x0 Channel B Transmit Buffer Empty.
x2 Channel B External/Status Change.
x4 Channel B Receiver Character Available.
x6 Channel B Special Receive Condition.
x8 Channel A Transmit Buffer Empty.
xa Channel A External/Status Change.
xc Channel A Receiver Character Available.
xe Channel A Special Receive Condition.

The "x" refers to the chip number, 0 through 7. Channel B goes to the even-numbered port, Channel A goes to the Odd numbered port. Double the chip number to get its even-numbered port, add 1 for the odd port. So, if x is 2 and the status comes from channel A, the port number is 5.

The CIO chip will modify the lower four bits of its vectors with status information as follows:

a0 DSR, port 0.
a2 DSR, port 1.
a4 DSR, port 2.
a6 DSR, port 3.
a8 FIFO Empty.
aa FIFO Half Full.
ac FIFO Full.

b0 EOP, DMA chip 0.
b2 EOP, DMA chip 1.
b4 EOP, DMA chip 2.
b6 EOP, DMA chip 3.
b8 EOP, DMA chip 4.
ba EOP, DMA chip 5.
bc PE, Printer Port.
be SLCT, Printer Port.

Remember, this status information is only important if you wish to interpret certain error messages that show it.

11.9. Glossary

ALM-2 The device that this software diagnoses is produced in two different forms. One of them is the ALM-2 board, a 16-port asynchronous device, which also has one parallel printer port. The other is the MCP board, described below.

MCP The MCP version of this product has four synchronous ports and no printer port.

Asynchronous This term implies that data communication goes on between two pieces of equipment that do not share a common clock. The ALM-2/MCP has twelve asynchronous-only ports and four more ports that can be either synchronous or asynchronous.

Synchronous This term implies that data communication goes on between two pieces of equipment, one of which is providing a clock for both of them to use.

Port A place for data to flow into or out of a device. The Sun ALM-2/MCP has fourteen RS232/RS423 ports and two more that can be either RS232/RS423 ports or RS449 ports.

RS232/RS423 RS232 is a usage convention that names a certain kind of socket and plug combination and certain pins in them for carrying data, handshaking and clocking signals for serial communication.

RS423 is a compatible revision of RS232. It specifies improved signal driving and receiving characteristics.

RS449 This specifies differential pairing of the same signals used in RS232. It uses twice as many wires to provide the same signals, allowing higher speeds over greater distances.

Handshaking Before two devices can pass data efficiently, they must exchange certain signals. These signals assure each one that the other is ready to send or receive.

Loopback This term implies connecting a port so that the data going out of it goes right back into it.

Serial A type of data communication in which data flows from one device to another one bit at a time.

Parallel A type of data communication in which data flows from one device to another more than one bit at a time, typically 8 bits.

DMA Controller

The 8237 direct memory access controller is a special-purpose LSI chip that permits the movement of data to and from memory. It has four "channels," which means it can control DMA data movement for four different devices. In the case of the ALM-2/MCP, one DMA controller chip can handle the transmission of data from Common RAM for four ports. In the case of the four ports that are able to handle synchronous I/O, receipt of data into Common RAM is handled by one DMA controller.

EOP The 8237 DMA Controller chips have an output pin called EOP, for End Of Process. When that signal is asserted, one of its channels has finished its requested data movement.

Terminal Count Terminal Count, or TC, is an event in one of the DMA controller channels. It means that the channel has moved all of the bytes it was expected to move. If one (or more) of the four channels of a DMA controller has reached TC, then that chip signals an EOP.

CIO The ALM-2/MCP employs the Zilog 8536 Counter/timer and I/O chip, called CIO, for generating interrupts to the workstation and for a few other functions. Signals from the DMA controllers as well as certain synchronous communication signals, if enabled in the CIO, can cause the CIO to generate a VME bus interrupt.

SCC The ALM-2/MCP uses eight Zilog 8530 Serial Communication Controller chips, called SCC's. These devices do most of the work needed to translate between bytes of data and bits of serial information with communication protocols. Another name for such a device is USART, for universal synchronous, asynchronous receiver transmitter.

12

Sun Memory Diagnostic

12

Sun Memory Diagnostic

12.1. General Description

The Sun Memory Diagnostic is a menu-driven diagnostic designed to detect errors in Sun-2 or Sun-3 main memory. It is a group of tests which can be run in a number of modes, giving the diagnostic flexibility while allowing it to thoroughly exercise the CPU board's main memory. It provides a default test sequence for ease of use.

NOTE *This diagnostic does not test cache on any system CPU board. It is a main memory diagnostic; if the CPU board has cache memory, it will be disabled during testing.*

12.2. Hardware Requirements

- The system must be either a Sun-2 or a Sun-3.
- The system must have at least 2 Megabytes of main memory.
- The system must have a current Boot PROM.
- The MMU must be functional, because it is tested by the Boot PROM.

12.3. The Main Menu

The Main Menu is shown below:

```
Sun-3/NNN Memory Diagnostic   Rev R.RR  MM/DD/YY   Menu

All          Execute ALL Memory tests (10), Long, Word then Byte
Default      Execute ALL Memory tests ( 5), Long, Word then Byte
Quick        Execute ALL Memory tests ( 1), Long

ADdress      Address test (unique pattern)
Checker      Checker test
Mats         MATS+ test
Nta          NTA test
Pattern      Pattern test (constant pattern)
Random       Random pattern test
Option       Set Options

Command ==>
```

Memory Diagnostic Command Descriptions

All

The *All* command executes the following command sequence:

```
opt long incr verbose obs ; set Pass=10 ; ad-r ;
opt word ; ad-r ; opt byte ; ad-r
```

Refer to the *Options* description at the end of this chapter for an explanation of the `opt` part of this command string. This sequence is stored in the variable `A_Main`, and can be modified from the Exec.

Default

The *Default* command executes the following command sequence:

```
opt long incr verbose obs ; set Pass=5 ; ad-r ;
opt word ; ad-r ; opt byte ; ad-r
```

This sequence is stored in the variable `D_Main` and can be modified from the Exec.

Quick

The *Quick* command executes the following command sequence:

```
opt long incr verbose obs ; set Pass=1 ; ad-r
```

This sequence is stored in the variable `Q_Enable` and can be modified from the Exec.

ADdress

The *Address Test* tests a specified block of memory using a data pattern that is incremented after each cycle. For example, in longword mode, entering:

```
ad address=100000 size=80000 pass=3 inc=2 pattern=0
```

writes 0x0 into location 100000 and 0x2 into location 100004. The last location tested is 17fffc. The test is executed for 3 passes. You need only enter **ad** to call up the address test, followed by these parameters:

`address=:` *base address for this test only*
`size=:` *size of memory to be tested for this test only*
`pass=:` *number of passes for this test only*
`inc=:` *increment*
`pattern=:` *base data pattern for this test only*

The default values for these parameters are:

```
address=0
size=environment size
pass=1
inc=1
pattern=0
```

This test writes a unique pattern into each address location, then reads it back, trying to quickly spot coupling faults in the memory chip decoder.

`Checker`

The *Checker Test* tests a specified block of memory using a pattern that is the logical negation of every other address location. For example, in long-word mode:

```
checker address=100000 size=80000 pattern=0x5555aaaa
```

writes `0x5555aaaa` into locations

100000,100008,100010...17fff8

and writes `0xaaaa5555` into locations

100004,10000c,100014...17fffc.

This test catches some types of coupling faults. After entering **`checker`**, these parameters may be entered:

`address=:` *base address for this test only*
`size=:` *size of memory to be tested for this test only*
`pass=:` *number of passes for this test only*
`pattern=:` *data pattern for this test only*

The default values for the parameters are:

```
address=0
size=environment size
pass=1
pattern=0x55555555
```

Mats

The *MATS+ Test* tests a specified block of memory for "stuck-at" faults (faults in a data bit is stuck at either the logic 1 or logic 0 state.)

The command syntax is:

mats *address= size= pass=*

The parameters are:

`address=:` *base address for this test only*
`size=:` *size of memory to be tested for this test only*
`pass:` *number of passes for this test only*

The default values of the parameters are:

```
address=0
size=environment size
pass=1
```

Nta

The *NTA Test* tests a specified block of memory for "stuck-at" faults and coupling faults. The test is quite thorough in its detection of both. To execute this test, use the following syntax:

nta *address= size= pass=*

The parameters are:

`address=:` *base address for this test only*
`size=:` *size of memory to be tested for this test only*
`pass=:` *number of passes for this test only*

The default values for the parameters are:

```
address=0
size=environment size
pass=1
```

Pattern

The *Pattern Test* tests a specified block of memory using the same pattern for every address location. This test is good for quickly spotting "stuck-at" faults. In order to execute the `pattern` test, use this syntax:

pattern *address= size= pass= pattern=*

The parameters are:

`address=:` *base address for this test only*
`size=:` *size of memory to be tested for this test only*
`pass=:` *number of passes for this test only*
`pattern=:` *base data pattern for this test only*

The default values for those parameters are:

```
address=0
size=environment size
pass=1
pattern=0xa55aa55a
```

Random

The *Random Test* tests specified block of memory using a randomly generated pattern for each address location.

This test writes a unique pattern into each address location to quickly spot coupling faults in the memory chip decoder. To execute this test, use this syntax:

. **random** *address= size= pass=*

address=: *base address for this test only*
size=: *size of memory to be tested for this test only*
pass=: *number of passes for this test only*

The default values for these parameters are:

```
address=0
size=environment size
pass=1
```

Option

The *Option Command* sets the default modes for the other tests. Each test can run in byte, word, or longword mode. Turning on one of these modes automatically turns the other two off. Each test can be run in *observe, verbose, increment* or *decrement* mode. These modes can be turned off by using *noobserve, noverbose, noincrement* or *nodecrement* .

The *increment* and *decrement* modes increment or decrement the specified pattern for the next pass of a test. The increment mode should not be confused with the increment parameter of the address test. For example,

```
opt increment ; ad pass=3 inc=2 pattern=11111111
```

will on its first pass use 11111111 as its base pattern. On the second pass it uses 11111112 and on its third pass use 11111113. For the random test, the seed used for the random pattern generation is incremented, resulting in a different series of random patterns being written on each successive pass of the test.

Options are set as follows:

Option *option(s)*

options may be any of the following, and you need only enter the letters shown below in upper case to invoke the option.

```
Byte Word Long OBserve Verbose
INCrement DECrement NOOBserve NOVerbose NOINCrement NODECrement
```

If you enter **Option** with no arguments, current options are displayed. The default options are:

```
long observe verbose noincrement nodecrement
```

12.4. Glossary

Coupling Fault
An error in which the change of value in one data bit on a memory chip changes the value of another bit on that chip. Such a pair of bits are said to be coupled.

DRAM
Dynamic RAM. A form of semiconductor memory requiring periodic refreshing to maintain its data. Sun-2 and Sun-3 main memory is implemented in DRAM.

Stuck-at Fault
An error in which a data bit is stuck at either the logic 1 or logic 0 state.

13

Sun-3 Mouse Diagnostic

13

Sun-3 Mouse Diagnostic

13.1. General Description

The Mouse Diagnostic checks the transmission of data from the mouse to the host system. This data contains information about mouse movements and the state of its switches (buttons).

The Mouse Diagnostic consists of the `button` test and `motion` test. Both tests prompt you for input and return to the main menu when the test completes or when you exit the test.

The user interface of the Mouse Diagnostic contains a single Main Menu. The individual tests prompt for necessary input or actions.

The mouse diagnostic requires no special hardware other than the mouse and "mouse pad" (for optical mice), along with a functioning Sun-2 or Sun-3 system.

13.2. Command Line Description

The minimum characters that must be entered to invoke each test are indicated by the upper case letters. Exceptions to this are the esc character and the ? character (not listed in the menu).

13.3. Mouse Data

The mouse transmits data when it experiences a change of state. The state changes when the mouse is moved on the pad, or one of its buttons is pressed or released. The mouse sends data in five-byte blocks. The start of each block is indicated by a "sync" byte. The "sync" byte contains the information about the state of the buttons. The other four bytes contain information about movement of the mouse on its pad.

13.4. The Main Menu

The Main Menu has four options: Typing **B** tests the mouse buttons. Typing **M** starts the mouse movement test. Pressing the escape key Esc returns the user to the previous menu. Pressing ? displays the Mouse Diagnostic Help Menu.

```
Sun-2/Sun-3 Mouse Diagnostic REV X.X  mm/dd/87 Main Menu

Button          Button Test

Motion          Motion Test

<esc>           menu exit (return to previous menu)

Command==>
```

B

The *Button*Test verifies that the mouse is transmitting the correct data about the state of its buttons.

The Button test prompts you to press the Left, Middle, and Right buttons. As a button is pressed, the data received from the mouse is compared with the expected value. The test displays an error message if the data is incorrect. The test then prompts for the next button.

The correct hexadecimal value for each "state" of the mouse buttons follows:

Values	*Left*	*Middle*	*Right*
0	down	down	down
1	down	down	up
2	down	up	down
3	down	up	up
4	up	down	down
5	up	down	up
6	up	up	down
7	up	up	up

If incorrect data values are received from the mouse, the error message indicates what value was expected and what value was received.

If the test receives no data for 10 seconds, it will ask if the appropriate button has been pressed. If you type **y** (yes), the Button test indicates that the button failed and prompts you to press the next button. If you type **n** (no), the Button test will wait ten seconds for you to press the button, then repeat the question if no data is received.

The Button test returns to the Main Menu when all buttons have been tested, or when you type **q**.

M

The *Motion*Test verifies that the mouse transmits data when it is moved on the mouse pad.

The Motion Test prompts you to move the mouse on the mouse pad. As you move the mouse on the pad, the cursor moves on the screen. If the test does not see any mouse data (indicating motion), the test waits 10 seconds, then asks if the mouse has been moved. If you type **y** (yes), the test prints an

error message and exits the test. If you type **n** (no), the test waits another 10 seconds, then repeats the question if it still receives no indication that the mouse has moved.

You may exit the Motion test by typing **q**. If the mouse has been idle for ten seconds and the motion test has verified movement of the mouse on the mouse pad, the test prints a message showing that the mouse passed, then exits to the Main Menu.

13.5. Error Messages

1 - <function> - Unable to get time of day

2 - <function> - data: expected <value> received <value>

3 - <function> - Unable to install exception handler

4 - <function> - SCC map failed; physical address <value>

5 - <function> - Unable to remove exception handler

14

MTI/ALM Board Diagnostic

14

MTI/ALM Board Diagnostic

14.1. Introduction

The MTI board is the serial communications subsystem for Multibus products. It provides an interface between the Multibus and either eight or fourteen RS-232-C ports. The ALM board is the serial communications subsystem for VME products. It provides an interface between the VMEbus and sixteen RS-232-C ports. The MTI/ALM Diagnostic is designed to test either the MTI or ALM boards. It consists of baud rate, stop bit, parity, modem, block, character and word length tests. Each one of the ports in the MTI or ALM board are tested under various configurations.

14.2. Hardware Requirements

The first requirement for this diagnostic is a fully configured Sun workstation, able to boot the Exec as described in Chapter 2 of this document. One megabyte of usable, functional memory is required. The required test fixtures are a minimum of 4 single loopback plugs and 2 loopback cables. However, having 16 loopback plugs and 8 loopback cables would be optimum. These plugs and cables are to be fitted to the appropriate RS232 sockets for each of the various configurations.

14.3. Operating Instructions

Follow the procedures found in *Chapter 2* for loading the SunDiagnostic Executive.

Recommended Test Procedure

Select the MTI/ALM diagnostic from the `diagnostic` menu after bringing up the Exec. From the main menu, select `Char`. If that test passes, select `All`. If any test fails, verify that you have the test fixtures on correctly and that you have told the MTI diagnostic about them, using `Group`. (You don't need to use `Group` if you are using the default test fixture configuration of all single loopback connectors.)

Each menu and the options associated with it are discussed next.

The Main Menu

The main menu looks something like this:

```
Sun Multibus/VME MTI/ALM Diagnostic x.x MM/DD/YY MTI/ALM

All        All Test Sequence
Default    Default Test Sequence
Character  single character data test
BLock      block data Test
BAud       baud rate test
Stop       stop bit test
Word       word length test
Parity     parity test
Modem      modem lines test
Group      user selects ports

Select one, or press <esc> to test next board or exit
```

After you press one of the letters shown in upper case, the MTI/ALM diagnostic performs the operation associated with that item, then it returns the menu again. The Exec is not case sensitive; you may use upper- or lower-case letters when interacting with it.

Each MTI board is tested individually. Up to four boards can be tested, beginning with board 1. You may run any number of tests on each individual board.

The following paragraphs describe the MTI/ALM menu choices.

All
Selecting **a** runs all the MTI/ALM tests.

Default
Selecting **d** executes the `Character`, `BLock`, and `Modem` options.

Character
Selecting **c** executes the read/write test. This module tests the MTI board character read and write capabilities.

BLock
Entering **bl** executes the block read/write test. This module tests the block data read and write capabilities.

BAud
Selecting **ba** tests a list of baud rates for the MTI board.

Modem
Selecting **m** turns modem line signals on and off and checks for continuity.

Group
Selecting **g** provides a sub-menu of selections that allow you to change the port test configuration.

<esc>
Pressing Esc and Return prompts you to test the next board, if one is installed, or exit the MTI board diagnostic.

Help

The following entry displays a single-line help message that explains the selected command:

```
help command

(replace "command" with one of the menu choices)
```

Configuration Selection Sub-Menu

Selecting the `Group` option from the main MTI/ALM menu brings up a sub-menu that looks something like this:

```
Select one of the following port configurations

.          Source Ports        Receiving Ports

single4      0-3                    0-3
single8      0-7                    0-7
single14     0-9,a-d                0-9,a-d
all          0-9,a-f                0-9,a-f
double8      0,1,3,5                7,2,5,6
double14     0,1,3,5,8,9,b          7,2,4,6,d,a,c
pairs        0,1,3,5,8,9,b,d        7,2,4,6,f,a,c,e
```

When you select `single4` from this menu, for example, the ports shown under `Source Ports` are paired with the ports shown under `Receiving Ports`. Therefore, for any configuration option that begins with `single`, each of the ports under test should be fitted with a single loopback connector. When the selection begins with `double`, loopback cables should be used. For the `pairs` option, use loopback cables, and for the `all` option, use loopback connectors.

You may also enter a series of ports, such as

```
0,1,2,3
```

which is the equivalent of entering

```
0-3 .
```

No spaces are required when you enter the port numbers. You may also enter combinations such as:

```
0-3,5-7,a,c
```

The examples given above represent single lists of ports, meaning that each port has a single loopback connector installed in it. You may also enter two lists, separated by a space, as follows:

```
0,2,4 1,3,5
```

In this example, ports 0, 2 and 4 are the source ports and 1, 3 and 5 are the receiving ports.

14.4. Test Descriptions

The following text describes each test in the MTI/ALM Board Diagnostic. Each of the tests writes the test data to the source port and then reads it from the receiving port for verification. the test is executed once when a single port is tested, twice when a pair of ports is tested. The tests first execute with the ports in their original configuration, then with the configuration reversed. That is, the source port becomes the receiving port and the receiving port becomes the source port.

Character Data Test

This module is designed to test the character read/write capabilities of the MTI and ALM boards. There are 8 different bit patterns that are tested. The data is written to the source port and then read from the receiving port for verification.

Block Data Test

This module tests the block data read/write capabilities of the MTI and ALM boards. The block of data is written to the source port and then read from the receiving port and compared byte for byte.

Baud Rate Test

This module tests the data at different baud rates. Again the data is written to the source port and read from the receiving port for verification. This is repeated for 16 different baud rates. The baud rates are listed below.

Baud Rate	
50	0.8 kHz
75	1.2 kHz
110	1.76 kHz
134.5	2.1523 kHz
150	2.4 kHz
300	4.8 kHz
600	9.6 kHz
1200	19.2 kHz
1800	28.8 kHz
2000	32.081 kHz
2400	38.4 kHz
3600	57.6 kHz
4800	76.8 kHz
7200	115.2 kHz
9600	153.6 kHz
19200	316.8 kHz

Stop Bit Test

This module tests 8 different stop bit patterns. Like all the modules previously described, this test writes the data pattern to the source port and reads it from the receiving port for verification. The test is repeated for three stop bit types. The stop bit values are 1.0, 1.5 and 2.0.

Word Length Test

This module tests word length sensitive data for 4 different word length types. The possible word lengths are 5, 6, 7 and 8 bits. Again the data is written to the source port and read from the receiving port for verification.

Parity Test

This module tests parity-sensitive data for three possible parity types. The parity types are "no parity", "odd parity" and "even parity". The test data is written to the source port and read from the receiving port for verification.

Modem Lines Test

This module tests the functionality of the modem lines. The line signals are turned on and off in checking for continuity. The on or off signal is detected through the 2661 chip.

14.5. Error Handling

Each test first identifies the port that caused the error and then displays the appropriate error message. In addition, each error message is recorded in the Executive message log (through the `execlog` function).

The following section explains various messages that are displayed by each of the tests.

`Data Compare error. read x, expected y, xor z.`
This is the most common error message for the MTI/ALM diagnostic. Any test may generate this message. It means that the data written to the source port and read from the receiving port does not match the original test data.

`Serial data, framing error. read x expected y.`
This message is generated from the `character read/write` test. It means that there was a framing error detected in one of the USART status register bits.

`Serial data, overrun error, read x expected y.`
This message originates from the `character read/write` test. In this case it means that an overrun error has occurred in one of the bits of the USART status register.

`Serial data, parity error, read x expected y.`
This message may be displayed when running the `character read/write` test or the parity test. In this case the parity error is detected in one of the bits of the USART status register.

`Data Set Ready (DSR) line, read x, expected y`
This message may occur during the `modem lines` test. It means that the DSR modem line is not functioning properly.

`Clear to Send (CTS) line, read x, expected y`
This message may also occur during the `modem lines` test. In this case it means that the CTS line is not functioning properly; that is, the transmitter buffer is empty.

`Tested OK.`
The test has passed. The only difference between this message and those ones previously described is that this message is not recorded in the Executive log.

`Check loopback connector`
If this message is encountered, check to see if the loopback connectors or cables are installed correctly. Also reconfigure the ports to match the installation of the loopback plugs or cables by using the `Group` option. This message is not recorded in the Executive log.

`Initialization Failed`
If the board is not properly initialized, this message will appear before the main menu is seen. If this message appears, check to see if the board is inserted correctly into the system and that the bus is configured for the board.

`Board not responding`
This message means the board is not sending information to the diagnostic. If this message is encountered, check to see if the board is inserted correctly into the system and that the bus is configured for the board.

14.6. Glossary

ALM
Asynchronous Line Multiplexer. A serial communications subsystem assembly that provides an interface between the VME bus and 16 RS-232-C ports.

Asynchronous
When communications equipment operates asynchronously, it means that different parts do not share a common clock. So to keep the receiving part in line with the transmitting part, the transmitting part must send some kind of signal that indicates it is starting or stopping a character. The MTI board design allows the sending of stop bits after each character.

Channel
A channel is part of a port, providing a path for data to move in only one direction. So on the MTI, each port has two channels: one for input, one for output.

MTI
Multiple Terminal Interface. A serial communications subsystem that provides an interface between the Multibus and either 8 or 14 RS-232-C ports.

Port
An electrical connection for data transfer.

Serial
A serial port moves data through a channel one bit at a time. Another type of port is called parallel, which moves data more than one bit at a time, usually eight. While a parallel port might seem to be more efficient, it limits the length of a connecting cable to just a few feet. A serial cable may be hundreds of feet long.

USART
Universal Synchronous/Asynchronous Receiver/Transmitter. A data communications controller chip.

14.7. References

For more information on either the MTI or ALM board, refer to these documents:

Installation and Service Manual for the Multiple Terminal Interface Board , Part Number 813-1007

VME Asynchronous Line Multiplexer Configuration Procedures, Part Number 813-2003

15

SCSI Subsystem Diagnostic

15

SCSI Subsystem Diagnostic

This diagnostic is associated with the SCSI Host Adaptor, Subsystem Disk Drive, Disk Controller and Tape Drive.

15.1. Introduction

Sun Microsystems supports several different SCSI Host Adaptors, Subsystem Disk Drives, disk controllers, and tape drives. There are three different SCSI Host Adaptor boards that control data transferred to/from outside world: SCSI2, SCSI3, and on-board SCSI3. The hardware board that communicates directly with the SCSI Subsystem disk drive is the disk controller board. Sun supports two (2) such boards, the Emulex MD21 and the Adaptec 4000. The controller takes care of many details of error checking, data transfer, and arrangement of the data on the disk. The board that interfaces with the tape drive is the tape controller. Sun uses two such boards, the Emulex Mt02 and Sysgen.

NOTE *Sun does not support system configurations in which both a SCSI3 board and a Sysgen disk controller board are installed.*

This diagnostic was designed to be as friendly as possible. During loading time, it will determine system configuration, such as the type of SCSI host adaptor, disk controller, and disk drive. This determination is used to set up certain tests for current configuration. If there is a problem, you need only answer some non-technical questions before the test is started.

At present, the following disk drives are supported for the ST506/SCSI environment (their respective capacity is also shown):

1. Micropolis 1304 - 40 Mbytes
2. Micropolis 1325 - 71 Mbytes
3. Fujitsu 2243 - 71 Mbytes

The following disk drives are supported for the ESDI/SCSI environment (their respective capacity is also shown):

1. Micropolis 1355 - 141 Mbytes
2. Toshiba MK 156F - 141 Mbytes

The following disk controllers are supported by Sun for the SCSI Subsystem (also shown are their respective interface types):

1. Adaptec 4000 ST506->SCSI
2. Emulex MD21 ESDI-> SCSI

These SCSI Host Adaptors are supported across Sun architectures:

- On-board SCSI3 — used on Sun-3/50, 3/60 systems.
- SCSI2, SCSI3 — used on the other Sun-2 and Sun-3 systems.

The following tape drive and controller configurations are supported by Sun SCSI Host Adaptors:

1. Wangtek Tape drive (Model 5099EG11) + Sysgen Controller.
2. Wangtek Tape drive (Model 5099EG11) + Emulex Controller.
3. Archive Tape drive (Model 5945 C) + Sysgen Controller.
4. Archive Tape drive (Model 5945 C) + Emulex Controller.
5. Archive Tape drive (Model 9050 B) + Sysgen Controller.
6. Archive Tape drive (Model 9020 B) + Sysgen Controller.

15.2. Problem Specification

The objective of the SCSI Subsystem Diagnostic is to ensure that the SCSI Host Adaptors, disk drives, disk controllers, and tape drives work correctly. Though the SCSI Subsystem diagnostic is designed to provide the testing capability for different SCSI Host Adaptors, Subsystem controllers and different SCSI Subsystem disk drives, the examples in this chapter reflect a configuration consisting of the Emulex MD21 Disk Controller along with a Micropolis 1355 disk drive.

15.3. Requirements

In the process of designing the SCSI Host Adaptor, Subsystem Disk Drive, disk Controller, and tape drive Diagnostic, the following performance, functional, hardware, and environment requirements were assumed.

Performance Requirements

The confidence level for the disk drive tests depends on the commands that are allowed by the disk controller to test the drive. The confidence level for the disk controller will depend on the accessibility and flexibility to test the hardware of the controller. This also applies to the tape drive as well. The maximum time for completion of a disk drive test should be less than or equal to 20 minutes. The maximum time for completion of a SCSI Host Adaptor or controller test should be less than or equal to 5 minutes and less than 30 minutes for tape drive tests.

Functional Requirements

You will be able to interrupt the execution of the SCSI Host Adaptor, Subsystem Disk Drive, Disk Controller, and Tape Drive Diagnostic after each test is executed and in some cases, while the test is being executed. Adequate on-line help is available. The SCSI Subsystem Diagnostic also generates and stores meaningful error messages for later retrieval. In addition to making default tests available, the parameters of each test are automatically given a default value.

Hardware Requirements

1. A working Sun-2 or Sun-3 CPU board
2. A working keyboard and mouse
3. A working monitor
4. SCSI Subsystem Disk Controllers (Adaptec 4000, Emulex MD21)
5. Formatted SCSI Subsystem Disk Drive (device under test - See Introduction Section for disks)
6. A boot device (i.e. local disk, local tape or remote disk over Ethernet)
7. SCSI2 or SCSI3 or on-board SCSI3.
8. Tape drives and tape controllers listed at the beginning of this chapter listed at the beginning of this chapter.

15.4. General Information

The standard power-up Boot PROM is needed to run SCSI Subsystem Diagnostic. The boot PROM is used to load and begin execution of the SCSI Subsystem Diagnostic. The boot path (Ethernet, disk, tape) is assumed to be checked out so that the SCSI Subsystem diagnostic can be loaded.

15.5. Operating Instructions

Read *Chapter 2* to load the SunDiagnostic Executive. After the Exec Main Menu is displayed on the screen, type `d; sub` to load scsisub.exec file and to start the test, and refer to *Chapter 2* for further information on how to set up and run the test under Exec.

NOTE *While running the SCSI Subsystem Test on a system that does not have a SCSI disk controller installed, execute only commands that refer to the tape drive. This includes commands from SCSI Host Adapter and Tape Drive sub-menus that are accessed from the SCCI Subsystem Diagnostic main menu. Executing commands that require the disk controller may fail or appear to "hang" the system. Under the SCSI Host Adapter menu, run only the "Test SCSIX W/O Disk and Controller" test (where X is 2 or 3); this will test all the host adapter-only functions that are available.*

On a system without a SCSI disk controller, the SCSI Subsystem Diagnostic may take between one and two minutes to load.

15.6. Overview of the Diagnostic

The SCSI Subsystem Diagnostic allows you the flexibility to test the SCSI Host Adaptor, disk controller, and disk drive separately. In addition, commands are provided to do the tests in a continuous sequence.

The user-interface of the SCSI Subsystem Diagnostic consists of a Main Menu with several sub-menus. A help option on each menu makes more detailed user command syntax available to the user. Esc and `MAin` options are also available on each test menu to provide the ability to (1) return to the current sub-menu and (2) return to the main menu.

15.7. The User Interface

The SCSI Subsystem Diagnostic attempts to create the most user- friendly environment possible. One of these features is that the you can choose to test either the SCSI Host Adaptor, disk drive, disk controller, tape drive or all of them. The user interactive command and response sequences are intended to be simple and straightforward.

The features of the user interface are covered in the following sections. The user interface consists of a menu with options associated with each diagnostic test.

The Main Menu

The Main menu contains the sub-menus and commands as follows: As in all the SunDiagnostic Executive tests, you need only enter the letters shown in upper case when making a menu selection.

```
SCSI SUB-SYSTEM DIAGNOSTICS. Rev xx, Date:xx/xx/xx  Main Menu

All                 Execute Scsi, disk and controller test.
Scsi Host Adaptor   SCSI Host Adaptor Menu.
Drive               Disk Drive test Menu.
Controller          Disk Controller Test Menu.
Tape Drive          Tape Drive test Menu.
SYSconfiguration    Subsystem configuration.
STop on error       Enable STop on error option.
?                   Help for Main Menu.
Exec                Return to Exec Menu

Command ===>
```

Main Menu items are described in the following paragraphs:

All
This selection executes the disk and controller test. The program starts executing all SCSI Host Adaptor, disk drive, disk controller, and tape drive tests.

Scsi
When you type this command, the program points to the SCSI Host Adaptor sub-menu.

Drive
Entering `D` brings up the Disk Drive Tests Menu, discussed later.

Controller
Typing `C` brings up the Controller Tests Menu.

Tape Drive
Typing **T** brings up the Tape Drive Test Menu

SYSconfiguration
Typing `SYS` invokes the Subsystem Configuration command, which displays the type of machine, type of scsi board, type of disk controller, type of disk drive, and a partition table found in the system under test.

ST or **SOE=**
Entering `ST` enables the stop-on-error option. This allows the test either to

keep running or to halt when an error occurs. The default is not to stop-on-error and the syntax is :

ST or **ST SOE=y** stops the test when an error occurs.

ST SOE=n lets the test run when an error occurs.

? The question mark brings up Help for the Main Menu, showing you how to select and run the test.

Exec
Entering E quits and returns to the Exec Main Menu.

SCSI Tests Menu

Each SCSI Host Adaptor has its own test menu. There are three SCSI menus: SCSI2 Menu, SCSI3 menu, and SCSI3(ob) menu.

The SCSI2 Menu

The following are the tests for SCSI2 Host Adapter board:

```
SCSI2 Adapter Diagnostics  Rev. xx, Date:xx/xx/xx  SCSI2 Test Menu

All                Execute all SCSI2 tests.
SCSI               Test Scsi2 w/o drive and controller.
ICR                Test Interface Control Register.
DMA                Test DMA Counter Register.
AD                 Test DMA Address Register.
TI                 Test Target Initialization.
DR                 Test Device Ready.
BT                 Test Data Transfer via Bus.
DT                 Test Data Transfer via DMA.
SI                 Test Status Interrupt.
DO                 Test DMA Overrun.
DI                 Test Data Integrity.
SCC                Test SCC (Multibus only).
?                  Help for SCSI2 test Menu.
ESC                Return to previous Menu.
MAin               Return to Main Menu.

Command ===>
```

The following are descriptions of the commands in this menu :

All
By executing this command, the program will execute all the tests given in this menu.

SCSI
This command tests SCSI registers without involving the disk drive and disk controller.

ICR
This command tests writable/readable bits in this 16 bits register.

DMA
This command tests the DMA counter register with the data pattern 0-FFFFH.

AD This command tests the DMA address register with the pattern 0-FFFFH.

TI This command initializes the target and waits for a proper response.

DR This command tests the device that interfaces to the SCSI host adaptor for ready communication.

BT This command tests the data transfer over the SCSI bus.

DT This command tests the data transfer over the DMA controller.

SI This command tests the Interrupt Register with the pattern 0x0-FFH.

DO This command transfers more data bytes than the DMA actually did and waits for a DMA overrun condition to occur.

DI This command does DMA transfers for 20 blocks with a random data pattern.

SCC
This command tests the Serial Communication Controller.

? This command will displays the help menu for SCSI2 menu.

Esc
Pressing the Esc key brings back the previous menu.

MAin
This command displays Main Menu.

SCSI3 Menu

The following are the tests for SCSI3 Host Adaptor board:

```
Scsi3 Adaptor Diagnostics  Rev.xx, Date:xx/xx/xx Scsi3 Test Menu.

All                 Execute All Scsi3 tests.
ICR                 Test Initiator Command Register.
MR                  Test Mode Register.
TCR                 Test Target Command Register.
NCR                 Test NCR 5380 Register.
CSR                 SCSI3 Control/Status Register.
DAR                 DMA Address Register
DCR                 DMA Counter Register.
FC                  Test FIFO Counter Register.
FD                  FIFO Data Register.
IV                  Test Interrupt Vector Register.
FID                 Test Interactive FIFO/DMA registers.
DF                  Test Interactive DMA/FIFO registers.
FR                  Test FIFO RAM.
BPR                 Test Byte Pack Register.
II                  Test Interactive Interrupt.
BDT                 Test Bus/DMA transfer.
SCSI                Test SCSI3 w/o disk and controller.
?                   Help for SCSI3 test menu.
MAin                Return to Main Menu.

Command ===>
```

Following are descriptions of the commands in the SCSI3 Test Menu:

All
This command executes all tests on the SCSI3 test menu.

ICR
This command tests the Initiator Command Register of NCR 5380.

MR This command tests the Mode Register of NCR 5380.

TCR
This command tests the Target Command Register of NCR 5380.

NCR
This command tests writable/readable registers of NCR 5380.

CSR
This command tests writable/readable bits of this register.

DAR
This command tests the 32-bit DMA address Register.

DCR
This command tests the 24-bit DMA counter register.

FC This command tests the 24-bit FIFO counter registers.

FD This command tests the 16-bit FIFO data register.

IV This command tests the writable/readable bits of the Interrupt Vector register.

FID
This command writes to the FIFO register and reads back from the DMA counter Register, then compares the data.

DF This command writes to the DMA counter register and reads back from FIFO, and then compares data.

FR This command tests FIFO RAM with a "walking 1" data pattern.

BPR
This command uses DMA t transfer 16- and 32-bit data in order to test 16-bit and 32-bit byte packing registers.

II This command tests the SCSI interrupts by setting up certain interrupt condition and check for the return.

BDT .
This command performs both bus and DMA transfer to/from the disk.

SCSI
This command tests the SCSI3 host adaptor without disk and controller involvement.

? This command will display the SCSI3 help menu.

MA This command brings you back to main Menu.

SCSI3(OB) Menu

The following are the tests for SCSI3(OB) Host Adaptor board:

```
Scsi3(OB) Adaptor Diagnostics  Rev. xx, Date: Scsi3(OB) Test Menu.

All         Execute All SCSI3 tests.
ICR         Test Initiator Command Register.
MR          Test Mode Register.
TCR         Target Command Register.
NCR         Test NCR 5380 Register.
CSR         Test Control/Status Register.
SC          Test Scsi Counter.
FR          Test FIFO RAM.
UDC         Test UDC Am9516.
UM          UDC Master Mode Register.
UAR         UDC Current Address Register.
UCNR        UDC Counter Register.
UPM         UDC Pattern/Mask Registers.
UIM         UDC Interrupt/Channel Mode Registers.
UCR         UDC Chain Address Registers.
BDT         Test Bus/Dma Transfer.
SCSI        Test SCSI3 w/o controller and drive.
?           Help for SCSI3 test menu.
MAin        Return to main Menu.

Command ===>
```

The following are descriptions of the commands in this menu :

All
This command executes all the tests in this menu.

ICR
This command tests the ICR of NCR 5380.

MR This command tests the MR of NCR 5380.

TCR
This command tests the Target Command Register of NCR 5380.

NCR
This command tests the writable/readable registers of NCR 5380.

CSR
This command tests all writable/readable bits of this register.

SC This command tests the 16-bit SCSI counter register.

FR This command test FIFO RAM on the SCSI3 board.

UCC
This command tests all writable/readable registers of the UDC Am 9516.

UM This command tests the Master mode register of UDC.

UAR
This command tests the UDC Current Address Register.

UCNR
This command tests the UDC Counter Register.

UPM
This command tests the UDC Pattern/Mask Registers.

UIM
This command tests the UDC Interrupt/Channel Mode Register.

UCR
This command tests the UDC Chain Address Register.

BDT
This command performs both Bus and DMA data transfer to/from the disk.

SCSI
This command tests all SCSI3 registers without disk and controller involvement.

? This command displays the SCSI3 help menu.

MA This command brings you back to the Main menu.

Controller Tests Menu

The following are the tests for a SCSI Disk Controller:

```
Disk Controller Diagnostics  Rev. xx, Date:  Controller Tests Menu

All             Execute All Controller Tests
Misc            Miscellaneous Tests Menu
Diagnostic      Diagnostic Command Test Menu
Read            Read Command Test Menu
Write           Write Command Test Menu
?               Help for Controller Menu
MAin            Return to Main

Command ===>
```

The following are descriptions of the commands in this menu:

All

When you execute this command, the program executes all the tests given in this menu.

M *repeat=*

This command causes the program to jump into a sub-menu where you can test the commands that do not relate to reads, writes or self-maintenance.

By using the Exec's `repeat=` command, you can select number of times the tests are to be executed.

D This command brings up the Controller Diagnostic sub-menu. From this sub-menu you can run individual tests that are explained in the menu.

R The program jumps to the sub-menu of tests for read-related commands.

W The program jumps into a sub-menu that contains tests for write-related commands.

? When you type this command, the program displays the syntax for each command.

MA This command returns you to the SCSI Subsystem Main Menu.

Diagnostic Command Menu The following are the tests that execute diagnostic related tests.

```
Disk Controller Diagnostics  Rev.xx, Date:xx/xx/xx Diag. Command Menu

All                 Execute All Controller Diagnostic Tests
Buffer              Read and Write Buffer
Send                Send and Receive Diagnostic
Disconnect & Rec.   Disconnect and reconnect.
?                   Help Diag. Command Menu.
ESC                 Return to Previous Menu
MAin                Return to Main Menu

Command ===>
```

The following are descriptions of the commands in this menu:

All
This command executes all the tests in the controller/diagnostic menu.

B This command tests the controller's data buffer memory and the SCSI bus integrity.

S This test requests that the controller perform diagnostic tests on itself, on the attached disk controller, or on both.

D This command tests the controller's disconnect/reconnect capability.

? When you type this command, the program displays the command syntax for each command in this menu.

MA This command returns you to the SCSI Subsystem Main Menu.

Controller Write Command Menu

The following commands are all write-related commands:

```
Disk Controller Diagnostics  Rev. xx, Date:xx/xx/xx  Write Command Menu

All             Execute All Controller Write commands.
Write           Write
Extended        Write Extended.
Long            Write Long (Emulex only)
?               Help for Write command Menu.
ESC             Return to Previous Menu
MAin            Return to Main Menu

Command ===>
```

The following are descriptions of the commands in this menu:

All
This command executes all the tests in the controller/write menu.

W This command instructs the controller to write data that was transferred by the host adapter to the disk drive.

E This command is basically the same as the Write command above except that it allows for two extra bytes in the command block for the logical block address and one extra byte for means of transfer length specifications.

L This command requests that the controller perform a write operation of one data block and the six bytes of ECC information. The data and the six ECC bytes for the specified logical block are supplied by the host adapter during the Data Out phase. Only the Emulex controller supports this command.

? This command displays the help menu that provides information on how to enter the commands in this menu.

MA This command returns you to the SCSI Subsystem Main Menu.

Controller Read Command Menu

The following are the read-related commands.

```
Disk Controller Diagnostic  Rev. xx, Date:xx/xx/xx Read Command Menu
All          Execute All Controller Read Commands
Read         Read
Defect       Read Defect List ( Emulex only )
Extended     Read Extended
Long         Read Long ( Emulex only )
?            Help for Read Command Menu
ESC          Return to Previous Menu
MAin         Return to Main Menu

Command ===>
```

The following are descriptions of the commands in this menu:

All
This command executes all the tests in the Controller/Read Menu.

R This command transfers a block (512 bytes) of data to the host. The logical starting block address is specified by the program.

D This command requests that the controller transfer the defect list maintained by the controller to the host adapter.

E This command will request that the controller transfer data to the host adapter from the disk drive. This command transfers four bytes of the block address instead of two as in the read command (described above). It also allows for specification of more logical data blocks of data to be transferred than the read command.

L This command requests that the controller perform a read operation of one data block and the six ECC bytes associated with that block. The data from the block and the ECC bytes are transferred to the host adapter during the Data In phase.

? This command displays the syntax for the commands in this menu.

MA This command returns you to the SCSI Subsystem Main Menu.

Miscellaneous Command Menu

The following are commands not considered to be read or write commands, but comprise all the other commands supported by the controllers.

```
Disk Controller Diagnostic  Rev. xx, Date:xx/xx/xx  Miscellaneous Menu

All                 Execute All Miscellaneous Tests
Inquiry             Inquiry (Emulex only)
SENse               Mode Sense
REServe             Reserve Unit (Emulex only)
RELease             Release Unit (Emulex only)
REZero              Rezero Unit
Seek                Seek
XSeek extd          Seek Extended (Emulex only)
STart stop unit     Start-Stop unit.
REQuest             Request Sense
Test Unit Ready     Test Unit Ready
ROM Version         Display ROM version. (Emulex only)
Performance         Performance Test
?                   Help for Misc. menu
ESC                 Return to Previous Menu
MAin                Return to main menu

Command ===>
```

The following are the descriptions of the commands in this menu:

All
This command executes all the tests in the Controller/Miscellaneous menu.

I This command causes the host adapter to request information regarding the controller and its attached disk drive(s). Only Emulex supports this command.

SEN
This command provides a means by which the host adapter may receive the medium, logical unit and peripheral device parameters from the controller.

RES
This command will reserve a specified LUN for exclusive use by the host adapter. Only Emulex supports this command.

REL
This command causes the LUN (connected to the controller and previously reserved by the Reserve Unit command) to be released. Only Emulex supports this command.

REZ
This command requests that the controller set the logical unit to the logical block address zero.

S This command causes the selected LUN to seek to the logical block addresses which are specified by the program.

XS This command is basically the same as the Seek command above except that this commands allows for two extra bytes in the command block for the logical block address. Only Emulex supports this command.

ST This command requests that the controller enable or disable the logical unit for further operation. For the Adaptec controller, the Stop command moves W/R head to shipping zone.

REQ
This command provides a means for the host adapter to obtain more detailed information after execution of a command. Typically, a Request Sense command is issued after the previous command had completed and a Check Condition status has been returned to the host adapter.

T This command causes the host adapter to check to see if the logical unit is ready.

ROM
This command displays the controller firmware revision level. Only Emulex supports this command.

P This command tests the performance level of the Adaptec controller board to see if it is able to perform up to specifications.

? This command displays the command syntax for each command in this menu.

MA This command brings you to the SCSI Subsystem Main Menu.

Disk Drive Tests Menu

The following menu shows the tests for a SCSI Disk drive :

```
Disk Drive Diagnostic Rev.x, Date:xx/xx/xx  Disk Drive Test Menu

All                 Execute All Disk Tests (W/R/S/CST)
Write               Write Test Menu (data destroyed)
Read                Read Test Menu
Seek                Seek Test Menu
CSTime              Check Seek Time
Drive Infor         Drive Information.
Format              Format disk drive.
SELect              Select disk drive.
User                User Select Test Block
QUick               Quick Test
?                   Help for drive test menu
MAin                Return to Main Menu

Command ===>
```

The following are descriptions of the commands in this menu:

All

This command executes the sequential W/R test, the seek test, and the check seek time test in the subsystem/disk menu. The test will default to one pass through each test.

W This command selects a sub-menu that contains tests that write data to the disk drive. Any previous data that was on the disk will be destroyed except the disk drive label and defect list data.

R This command selects a sub-menu that contains tests that read the disk drive and compute the soft and/or hard error rates per error.

S This command selects a sub-menu of tests that perform different types of seek test patterns and compute seek error rates per error.

CST

This command executes the tests that calculate the seek times (Average, track-to-track, 1/3 stroke, and maximum) on the disk drive under test. Due to critical timing, it must be run as single tasking to guarantee the time accuracy.

D This command displays the geometry of the drive under test.

F This command formats the drive with default drive parameters obtained from controller. The confirm message will be displayed before the test is started.

SEL

This command selects the disk drive to be tested by the remainder of the commands in the Disk Drive Test Menu. It prompts for the number (0 or 1) of the disk drive to be tested.

U *b=block address*

This command executes write/read commands to the disk drive. Up to 128 blocks, beginning at the block address you specify, is written and then read back. The test checks to see if the specified block address is within the limits of the drive or the block address 0 is selected. If it is not, an error message is displayed. The confirm message will be displayed and wait for your the test is executed.

QU This command executes a quick test by writing and then reading back a block of data on the inner, the outer, and the middle cylinder. You need to respond to a warning message before the test is started.

? This command displays the menu of each of the tests and gives brief descriptions of what each command will do.

MA This command will quit the current menu and return to the Main Menu.

Disk Write Test Menu

The following are the tests for an SCSI Disk write test :

```
Disk Drive Diagnostic  Rev. xx, Date:xx/xx/xx  Write Test Menu

All          Execute All Disk Write Tests
6DB          Write with worst case data pattern 0x6DB
B6D          Write with worst case data pattern 0xB6D
DB6          Write with worst case data pattern 0xDB6
7A6E         Write with pattern 7A6E (for RLL drives)
Random       Random Write Test
?            Help for Write test menu
ESC          Return to Previous Menu
MAin         Return to Main Menu

Command ===>
```

This menu consists of commands that will write data patterns to the drive. You must respond to the warning message before writing to the the disk drive is allowed. The label block (block 0) and defect list block will be preserved and skipped during all the write tests.

All
This command executes all tests in the disk/write menu.

6DB
This command writes the entire disk except the disk drive label and defect list block with the pattern 0x6DB.

B6D
This command writes the entire disk except the disk drive label and defect list block with the pattern 0xB6D.

DB6
This command writes the entire disk except the disk drive label and defect list block with the pattern 0xB6D.

7A6E
This command writes the entire disk except the disk drive label and defect list block with the pattern 0x7A6E. This pattern is used to test the drives that are using RLL (Run Length Encode).

R This command randomly selects a data pattern for a random block address before the write command is executed. It will repeat for 300 random blocks which are generated by a random number generator.

MA This command exits from this menu and returns to the Main menu.

Disk Read Test Menu

The following are the tests for the Read Menu of the SCSI disk:

```
Disk Drive Diagnostic  Rev. xx, Date:xx/xx/xx  Read Test Menu

Soft                Read until 1x10^10 bits transferred is met
Hard                Read until 1x10^12 bits transferred is met
User                User select number of times to loop
Random              Random Read Test
?                   Help for Read test Menu
ESC                 Return to Previous Menu
MAin                Return to Main Menu

Command ===>
```

The following are descriptions of the commands in this menu:

S This command reads the drive sequentially until at least 10^{10} bits have been transferred. In this way soft error rate of the drive is computed. The number of loops needed to complete this test is automatically computed by the program. You may abort the test any time by typing the ! key.

H This command reads the drive sequentially until at least 10^{12} bits have been transferred. This process computes the hard error rate of the drive. The number of loops needed to complete this test is automatically determined by the program. You may abort the test any time by typing the ! key.

U *PASs=*

With this command you may select the number of times to loop on a test. At the end of the test, the number of bits that have been transferred is computed.

The value you enter after *PASs=* is the number of times you want to execute the test. If you do not enter a value, the test will run only once (the default). Note that the *PASs* argument differs from the `repeat=` command, in that it keeps track number of loops that have been completed for later use of error rate calculation, while the `repeat=` argument is controlled by the Exec, and doesn't keep track of the number of loops.

R This command will perform 300 random reads on the drive under test.

? This command displays information regarding usage of the commands in this menu.

MA This command exits the Read Test Menu and returns to the Main menu.

Disk Seek Test Menu The following are tests for the Seek Test Menu:

```
Disk Drive Diagnostic  Rev. xx, Date:xx/xx/xx  Seek Test Menu

All             Execute All Disk Seek Tests
Ping-pong       Seek with ping-pong pattern until 1x10^6 seeks
Center          Seek with center-in pattern until 1x10^6 seeks is met
UPPong          Seek ping-pong per user select loop
UCenter         Seek center in per user select loop
?               Help for Seek test Menu
ESC             Return to Previous Menu
MAin            Return to Main Menu

Command ===>
```

NOTE *Note that the PASs argument differs from the* `repeat=` *command, in that it keeps track number of loops that have been completed for later use in error rate calculation, while the* `repeat=` *argument is controlled by the Exec, and doesn't keep track of the number of loops.*

The following are descriptions of the commands in this menu:

All
This command executes all the tests in the Disk/Seek menu.

P This command performs a seek test with a ping-pong pattern that moves the W/R head from cylinder 0 to last cylinder, back to cylinder 0 and then to last cylinder *n*, where *n* is incremented by one for each complete W/R head move. When *n* = last cylinder, it will be reset to 0 and the test will be repeated until at least 10^6 seeks are met. The number of loops are determined by the program for each type of drive.

C This command performs a seek test with a center-in pattern that moves the W/R head from the cylinder 0 + *n* to the last cylinder - *n*, where *n* is incremented by one for each complete W/R head move. When *n* = (last cylinder + 2), it will be reset to 0 and the test will be repeated until at least 10^6 seeks are met.

UPP *PASs=*
This command does a seek test with the ping-pong pattern after you select the number of loops to run.

PASs is the number of times you want to execute the test. You must enter a number greater than 0 for the test to be executed. The default is one pass.

UC *PASs=*
This command does a seek test with the center-in pattern after you select the number of loops to run.

PASs is the number of times you want to execute the test. You must enter a number greater than 0 for the test to be executed.

? This command displays information regarding usage of the commands in this menu.

MA This command exits the Seek Test menu and returns to the Main Menu.

Tape Drive Tests Menu

The following are the tests for the Tape drive.

```
Tape Drive Diagnostics Rev. xx, Date:xx/xx/xx Tape Drive Test Menu.

All          Execute All Tape Tests.
RWT          Rewind the Tape.
ERT          Erase the Tape.
RTT          Retension the Tape.
RDT          Read the Tape.
WRT          Write/Read the Tape.
FST          File Skipping Test.
WET          Write until End of Tape.
RAT          Read Alignment Tape.
TPI          Tape Information.
Q11          Set up QIC-11 Format.
Q24          Set up QIC-24 Format.
?            Help for Tape test Menu.
ESC          Return to previous Menu.
MAin         Return to Main Menu.

Command ===>
```

The following are descriptions of the commands in this menu:

All
This command execute all tests in the Tape Drive test menu.

RWT
This command rewind the tape to physical BOT (Beginning Of Tape).

ERT
This command erases the whole tape if the tape is not write protected.

RTT
This command retensions the tape for better tape drive performance.

RDT
This command does read operations on a certain number of blocks on the tape.

WRT
This command performs write/read tests on ten (10) files, 27 blocks each. Write/read data is compared after each file is created and written with a random data pattern. QIC_11 and QIC_24 format is also be tested with this command.

FST
This command writes seven (7) files to the tape with data pattern 2929H - 2931H and then randomly reads back one (1) file by doing file-skipping with random numbers (1-7). The data will be compared after the read operation to validate it.

WET
This command writes from BOT to EOT with a random data pattern. This test assures that the w/r head mechanism steps and moves up and down properly. If it does, the BOT and EOT signal is trapped and the test is completed. For a SCSI-3/SYSGEN combination, this test takes about 20 times longer to run as compared to the other combinations. This increase in runtime is because the Sysgen tape controller takes longer to return status information to the SCSI-3 host adapter. There is a warning message concerning this problem before the test is executed; you must decide whether to proceed or to abort the test.

RAT
This command reads the alignment tape to assure that the w/r head is aligned properly. Currently, there are two alignment tape formats (QIC11 and QIÇ24) that are supported. After each file is read, the data is checked and compared to guarantee it is valid.

TPI
This command displays the tape configuration information, such as the type of SCSI board, type of tape controller, QIC format being set up, and so son.

Q11
This command sets the Tape controller to handle QIC-11 format. All data written on the tape with QIC-24 format will not be read by QIC-11 format, and vice versa; an error message will inform you if this condition exists.

Q24
This command sets the tape controller to handle QIC-24 format. The restriction described for the `Q11` command also applies here.

? This command displays the help menu.

The Escape Key
Pressing Esc returns you to the previous menu.

MA This command returns you to the Main Menu.

15.8. Error Handling

There are two types of error messages. One (code 1H- 50H) is the message interpreting the error code that was returned by disk controller using the request sense command. The other (code 80H-90H) type is comprised of error messages that come from the test program. The diagnostic test reports the failure of the selected test in as much detail as possible and saves it in a log file. The stop-on-error option, which you can set up from the Main Menu allows the test either to keep running or to halt when an error occurs. Please refer to *Chapter 2* for information on displaying and saving the error log file.

15.9. Message Interpretation

The following are the error code descriptions, according to vendor or OEM technical manuals (SYSGEN, Adaptec, and Emulex). Please refer to these manuals for more detail.

```
Code 1   - No index signal.
Code 2   - No seek complete.
Code 3   - Write fault.
Code 4   - Drive is not ready.
Code 5   - Drive is not selected.
Code 6   - No track 0 found.
Code 8   - Target is busy or Command queue is full.
Code 9   - Tape media is not installed.
Code AH  - Not enough space on tape for transfer.
Code BH  - Tape drive time-out.
Code 10H - I.D CRC error.
Code 11H - Uncorrectable data error.
Code 12H - I.D address mark not. found.
Code 13H - Data address mark not found.
Code 14H - Block or sector not found.
Code 15H - Seek error.
Code 16H - DMA time-out while serving tape drive.
Code 17H - Tape write protected/Read error recovered with retries.
Code 18H - Ecc recovered read error.
Code 19H - Ecc error during verify/Defect list error/Tape bad block found.
Code 1AH - Interleave error or Parameter overrun.
Code 1CH - Unformatted/bad format on drive/Tape file mark detected.
Code 1DH - Self test failed or compare error.
Code 1EH - Defective track ( media error).
Code 20H - Invalid command.
Code 21H - Invalid/illegal block address.
Code 22H - Illegal function for device.
Code 23H - Volume overflow.
Code 24H - Bad argument/illegal field in CDB.
Code 25H - Invalid logical unit number.
Code 26H - Invalid field in parameter list.
Code 27H - Write protected.
Code 28H - Medium change.
Code 29H - Power up or reset occurred.
Code 2AH - Mode select just changed.
Code 30H - Tape Unit Attention.
Code 31H - Format Failed/Tape command time-out.
Code 32H - No alternate track on LUN.
Code 33H - Tape Append error.
Code 34H - Read EOT ( End Of Tape).
Code 40H - RAM error detected.
Code 43H - Message reject error.
Code 44H - SCSI hardware/firmware error.
Code 45H - Select/reselect failed.
Code 47H - Parity error.
Code 48H - Initiator detected error.
Code 49H - Inappropriate/illegal message.
```

The following error messages are interpreted for the SYSGEN tape controller:

```
QIC2-0  81H - File Mark Detected.
QIC2-0  82H - Bad Block not Allocated.
QIC2-0  84H - Unrecoverable data error.
QIC2-0  88H - End of Media.
QIC2-0  90H - Write protected Cartridge.
QIC2-0  A0H - Unselected Tape Drive.
QIC2-0  C0H - Tape not in place or just inseted.
QIC2-1  81H - Power on/Reset occurred.
QIC2-1  82H - Reserved for end of recorded media.
QIC2-1  84H - Reserved for bus parity error.
QIC2-1  88H - Beginning of Media.
QIC2-1  90H - Marginal Block detected.
QIC2-1  A0H - No data detected.
QIC2-1  C0H - Illegal Command.
```

The following are the program error code messages, which tell you in which routine the error occurred and for which reason the error was generated.

```
Code 80H - DMA TRANSFER : Error returned from CHECK_PHASE.
Code 81H - DMA TRANSFER : Error returned from SEND_COMMAND_TO_CDB.
Code 82H - DMA TRANSFER : Expect cnt=0, Observed cnt=xxxx.
Code 83H - SEND_COMMAND_TO_CDB : Error returned from CHECK_PHASE.
Code 84H - BUS_TRANSFER : Error returned from SEND_COMMAND_TO_CDB.
Code 85H - CHECK_STATUS: Error returned from CHECK_PHASE.
Code 86H - PHASE_MATCH : Phase was mismatched.
Code 87H - REQUEST_SENSE : Error returned from SEND_COMMAND_TO_CDB.
Code 88H - REQUEST_SENSE : Expect no error, observed check condition.
Code 89H - TARGET_INITIALIZATION : Busy bit never asserted.
Code 8AH - TEST UNIT READY : Error returned from SEND_COMMAND_TO_CDB.
Code 8BH - Error while doing Exec Map.
Code 8CH - HANDSHAKING : Request bit never toggle.
Code 8DH - DMA TRANSFER(W/R): Bus_err/Scsi_bcnfl bit is set.
Code 8EH - REQUEST_SENSE: Error from w/r_via_bus.
Code 8FH - SEND COMMAND TO CDB: Error from target_initial.
Code 90H - DMA TRANSFER: Sbc_int bit was not set after Dma.
DEFAULT  - Unknown error returned from the controller.
```

15.10. Failure Analysis

The purpose of failure analysis is to help narrow down to a certain area what might cause the error, although those components may not necessarily be bad. It is up to the service person to perform further tests, investigation and troubleshooting to resolve the problem. The error codes listed below are divided into groups to help you determine the area in which the error occurred.

The following error codes are most likely to be disk drive related:

ERROR CODES (in Hex): 1, 2, 3, 4, 5, 6, 10, 11, 12, 13, 14, 15, 16, 17,18, 19.

ERROR CODES (in Hex): 1A, 1C, 1E, 23, 27, 28,

The rest of the controller error codes are disk controller related:

ERROR CODES (in Hex): 8, 1D, 20, 21, 22, 24, 25, 26, 29, 2A, 31, 32.

ERROR CODES (in Hex): 40, 43, 44, 45, 47, 48, 49.

All of the program error codes (80H - 90H) are most likely referring to SCSI board problems, such as bad connections, faulty switch settings, no power on, a problem with the target device, bad cable, and so on. It is recommended that you power down both the system and storage device to do a check. If the problem still exists then replacing the SCSI board is the next step.

The `all` option in the Main Menu of the SCSI Subsystem Diagnostics performs an exhaustive test on the SCSI host adapter, disk drive, disk controller board, and tape drive. It provides a more confident statement regarding SCSI subsystem device functionality.

15.11. Glossary

ACB	Adaptec 4000 Disk Controller Board
CPU	Central Processing Unit
Cylinder	Most common disk devices consist of a number of platters mounted on a spindle spinning at a high speed. The cylinder is geometrically located at a radius (from the center of the spindle) on all platters. Disks store information on a thin magnetic coating.
ESDI	Enhanced Small Device Interface
Head	To read and write the information on the surface of the disk, a number of heads are mounted on a common arm so they travel together. Usually, there are two heads for each platter (one for the top and one for the bottom surfaces).
Mbytes	Megabytes
MD21	Emulex MD21 Disk Controller Board
ST506	Seagate Technology Disk Interface
Sector	Each track is further divided into segments called sectors. The sector is a basic unit of storage on the disk. On Sun systems, each sector contains 512 bytes of data and is surrounded by a header and trailer containing addressing and error correction information.
SASI	Shugart Associates System Interface
SCSI	Small Computer System Interface
Track	A track is the portion of a disk passing under a single stationary head during disk rotation. The cylinder, tracks and sectors each inhabit a distinct dimension that separates and locates them; cylinders are located by distance from the spindle; tracks are separated from each other by surface on platters; and sectors from each other by time.

16

Sun-2 Sky FFP Diagnostic

16

Sun-2 Sky FFP Diagnostic

16.1. Sky Board General Description

The Sun-2/120 Sky Fast Floating Point board is a floating point accelerator used in Multi-bus or VMEbus systems. This board has no I/O (input/output) capabilities; it uses the bus for its data transfers. The Sky board is primarily used to perform math operations rapidly on 32-bit integers.

16.2. Sky Board Functional Overview

The Sky FFP board must have its RAM loaded with microcode before use. At this point the board is initialized and is ready to execute commands. Command codes are sent from the CPU to the Sky board over the bus, followed by the data (usually two 16-bit words) that requires computation. The command code describes the operation the Sky board is to perform on the Data and what form returned data should take. If a command code arrives while an operation is in process, the Sky board performs Context Restore/Save, which saves the current task on the stack, loads the new task, completes it, then restores and completes the previous assignment. In this way the Sky board can be considered a multitasking device.

The Sky FFP is a single board that plugs directly into the backplane of a Multibus or VMEbus cardcage. The board requires +5 VDC, at 4.0 Amps and can operate in an environment of 0 to 55 degrees Celsius. The board uses the 2901 bit-slice ALU and 16 x 16 Multiplier, running at 8 MHz internal cycle time (125 ns). The Program Memory on board is equal to 4K x 32 Bits of RAM, with seven levels of interrupt available.

The Microcode is loaded to the Sky board by means of the Microcode Register (16 Bits), while commands are received through the Status/Control Register (16 Bits.) The data comes into the board through two Data Registers (16 Bits.)

16.3. Sky Diagnostic Test Overview

The Sky Fast Floating Point Diagnostic exercises the Sky FFP board with the following tests:

Addition	Logarithm	Microcode Load
Subtraction	Division	Microcode Restore
Multiplication	Configuration	Microcode Save

The Sky FFP Diagnostic works on both the Sun 2/120 (Multibus) and the Sun 2/160/130 (VMEbus) systems. Although the boards physically differ, they are the same electrically for both systems. No external (I/O) cables are required to operate the Sky FFP board.

16.4. Hardware Requirements

To operate the Sky Floating Point Diagnostic, a Sun system must contain:

- A Sun-2 CPU board with:
  1. Two Mbytes of RAM
  2. A Sun-2 Video board
  3. A Sun-2 Ethernet board or SCSI Tape drive (for loading the diagnostic)
  4. A Sky Fast Floating Point board.

16.5. The Main Menu

The Main Menu of the Sky FFP Diagnostic is shown below. A total of nine different options are listed in the menu.

```
Sky Fast Floating Point Board Diagnostic  REV X.X  mm/dd/yy Main Menu

Configure       Configuration Test Sequence
Microcram       Microcode Load Test Sequence
Restore         Context Restore/Save Test Sequence
ARithmetic      Arithmetic Selection Menu
All             All Test Sequence
Default         Default Test Sequence
Quick           Quick Version Test Sequence

Command ==>
```

Configure

The *initialization test* reads the system's ID PROM to determine whether the board is connected to a Multibus or VMEbus backplane. The test then sets the appropriate Physical Address and read the board's address back ten (10) times. If an error should occur, the diagnostic stops with an error message such as:

```
Sky board not installed in system.
Check system configuration.
```

At the same time, the error is logged to disk.

Microcram

The *Microcode Load test* writes approximately 4080 bytes of Microcode to the Command Register of the Sky board. Each byte written is then read back to insure its accuracy and determine if the correct amount of code has been loaded. The Sky diagnostic displays a message showing the total number of bytes loaded, along with the Sky board's status. If an error occurs during the Microcode load, the diagnostic is stopped and the error is logged to disk. An error at this point is serious enough to terminate operation of the diagnostic, since the Sky board must have its microcode loaded to accept commands from the host CPU. Once loaded, the Microcode resides in the Sky board's RAM.

Restore

The *Restore/Save test* tests the Sky board's ability to operate in a multitasking environment. Eight bytes of data are saved, then restored, simulating an interrupt for a Context Restore/Save. The bytes are displayed during

the Restore portion of the test. If an error occurs during this test, the error message

```
Error while running Restore test-Check log
```

is displayed, the test is stopped, and the error is logged to disk. The diagnostic can still be used after this type of error.

ARithmetic

This menu item selects the *Arithmetic Sub-Menu.* For an explanation of the available tests, and a description of each, see the *The Arithmetic Selection Menu* section. The Microcode load test must be executed prior to running these tests, or the error message

```
Sky board NOT configured, running configuration routine
```

is displayed.

All

The *All test sequence* executes the following tests automatically in the order they appear. Each displays information about the Sky board and log errors to the disk, just as if they were initialized manually.

1. Configuration Test: Quick Configuration Check, and set Physical Address.
2. Microcram Test: Load the 4080 Bytes of Code into the Sky board RAM.
3. Restore/Save Test: Context Restore/Save, 8 - 32 bit words are Saved and Restored.
4. Logarithm Test: Take the Log of a 1.0
5. Addition Test: Add two Random Floating Point Numbers 25 times.
6. Subtraction Test: Subtract two Random Floating Point Numbers 25 times.
7. Multiplication Test: Multiply two Random Floating Point Numbers 25 times.
8. Division Test: Divide two Random Floating Point Numbers 25 times.

The `All test Sequence` command can have a `Loop=*` appended added to the command line for infinite execution, as shown below.

```
Command ==>All , Loop=*
```

The Exec displays a message containing the Pass Loop number each time the test is executed. This way you know how many passes the test sequence has executed.

Default

The *Default Test Sequence* executes the tests listed below:

1. Initialization Test: Quick Configuration Check, and set Physical Address.

2. Microcram Test: Load the 4080 Bytes of Code into the Sky boards RAM.
3. Restore/Save Test: Context Restore/Save, 8 - 32 bit words are Saved and Restored.

The `Default Test Sequence` is intended to be a default for the Main Menu. If the you know nothing about the Diagnostic and want to run the tests displayed, select the default option and invoke the tests listed above.

`Quick`

The *Quick Test Sequence* executes the `Configuration Test Sequence` once. This test can be used to determine if the Sky board is configured correctly.

16.6. Main Help Menu

The main Help Menu is shown below. It can only be selected from the Main Menu, by typing **?**. The Help Menu contains an explanation of the test options in the Main Menu. It contains information only; tests can't be selected from this menu. To leave this menu, enter Esc on the command line.

```
Sky Fast Floating Point Board Diagnostic  REV x.x  mm/dd/yy  Help1  Menu

Configuration       Will Execute a Test checking the SKY Boards Configuration.
Microcram           Will Execute a Micro Code Loading test on the SKY Board.
Restore             Will Execute a Multiple User Test (Context Switch).
Arithmetic          Will Display a Menu of the Arithmetic Tests.
All                 Will Execute ALL the Sky FFP board tests in sequence.
Default             Will Execute a Default series of Sky FFP board tests
Quick               Will Execute a Quick test on the Sky board.


Command ==>
```

This is the only level of help for the Main Menu. It explains what each test option does. If **?** is typed at this point, the message

```
Already in the help Menu
```

is displayed.

16.7. The Arithmetic Selection Menu

The `Arithmetic Selection Menu` is shown below. Nine options are included in the this menu. Each option corresponds to a character or set of characters for ease of test selection.

The math function tests all use Single Precision Floating Point Real numbers, (SPR), generated by a random number routine. The results of a test are calculated with software, then used to verify the hardware value. If the hardware answer doesn't match the software answer, an error message is generated.

If an error is found during the execution of a Math Function, the test that is currently running exits. The diagnostic continues to the next test after logging the error.

```
Sky Fast Floating Point Board Diagnostic     REV X.X   mm/dd/87  Arithmetic Menu

LOG                Logarithmic Test Sequence

ADdition           Addition Test Sequence

SUbtraction        Subtraction Test Sequence

MUltiplication     Multiplication Test Sequence

DIvision           Division Test Sequence

All                All Math Test Sequence

Command ==> LOG
```

LOG

The *logarithm test* takes the logarithm of a known value (1.0), computes its value, checks it against the known answer, then displays the result. The test repeats 256 times. If an error occurs during execution, the message `Logarithm Failed` is displayed, and the error is logged to the disk.

ADdition

The *Addition test* uses single precision floating point numbers to add two random numbers and generate an answer. The addition is repeated 256 times , each time using a different random number. The last iteration uses a hard coded value of 1.0 as the operand. If an error occurs, `FPP Addition Test done- Errors:xx.` is displayed, and the error is logged to disk.

SUbtraction

The *Subtraction test* uses single precision floating point numbers to subtract two random numbers and generate an answer. The subtraction is repeated 256 times , each time using different random number. After each pass, the answer and the two random operands are displayed. The last iteration uses a hard coded value of 1.0 for the operand. If an Error occurs, `Subtraction Failed` is displayed and the error is logged to disk.

MUltiplication

The *Multiplication test* uses single precision floating point numbers to multiply two random numbers and generate an answer. The operation is repeated

256 times , each time using different random number. The last iteration uses a hard coded value of 1.0 for the operand. If an error occurs, `Multiplication Failed` is displayed and the error is logged to disk.

`Division`

The *Division test* uses single precision floating point numbers to multiply two random numbers and generate an answer. The operation is repeated 256 times , each time using different random number. The last iteration uses a hard coded value of 1.0 for the operand. If an error occurs, `Division Failed`, is displayed and the error is logged to disk.

16.8. Arithmetic Help Menu

The Arithmetic Help Menu is shown below. Type **?** to bring it up. The Help Menu contains an explanation of the Arithmetic Menu options. It contains information only; tests can't be selected from this menu. To leave this menu, enter Esc on the command line.

```
Sky Fast Floating Point Board Diagnostic   REV 0.1   01/14/86  Arithmetic Help2  Menu

LOG            Will Execute the LOG Function on 1.0.

ADdition       Will Execute a Single Precision Floating Point Addition.

Subtraction    Will Execute a Single Precision Floating Point Subtraction.

Multiplication Will Execute a Single Precision Floating Point Multiplication.

DIvision       Will Execute a Single Precision Floating Point Division.

All            Will Execute ALL the Math Test Choices Listed Above.

Command ==> ESC
```

16.9. Error Handling

Error Handling for the Sky Diagnostic takes two steps: the error message is displayed, then it is logged to the disk. In this way a history of the diagnostic is preserved, and can be reviewed at a later time. The error message warns you that an error has occurred.

16.10. Recommended Test Procedure

The `All Test Sequence` option on the Main Menu provides a thorough test of the board's functionality. Upon successful completion of this series of tests, the board should have been exercised as though it were running under normal conditions.

The `Default Test Sequence` tests only the board's ability to load Microcode. It does not test the Math functions. The default option provides a standard set of main menu tests, and should not be used as an overall test of the Sky board.

The `Quick Test Sequence` only checks that the Sky board has been initialized properly and that the correct Physical address has been set. This test should be used as a first test to make sure the CPU can communicate with the Sky board. It is a good check of the board's configuration.

16.11. Glossary

Executive
The diagnostic operating system, under which this diagnostic runs.

Sky FFP
Sky Fast Floating Point Board.

17

Sun SMD Diagnostic

17

Sun SMD Diagnostic

17.1. General Description

Sun Microsystems supports a number of SMD Drives and controllers. The SMD controller board communicates directly with the SMD Drive. The Controller takes care of the details of error checking, data transfer, and arrangement of the data on the disk.

The following Fujitsu, CDC, NEC, and Hitachi SMD Disk Drives are currently supported by this diagnostic:

Table 17-1 *Supported Disk Drives*

Fujitsu	2312	8"	84	Mbytes
"	2284	14"	169	Mbytes
"	2322	8"	168	Mbytes
"	2333	8"	337	Mbytes
"	2351	Eagle	474	Mbytes
"	2297	14"	600+	Mbytes
"	2361	Eagle xp	689	Mbytes
CDC	9720	8"	337	Mbytes
NEC	2363	9"	9xx	Mbytes
Hitachi	815-10	9"	9xx	Mbytes

The following SMD Disk Controllers are currently supported by this diagnostic:

Table 17-2 *Disk Controller Boards*

Xylogics	450	Multibus
Xylogics	451	Multibus
Xylogics	7053	VME

17.2. Hardware Requirements

- A working Sun CPU Board.
- A working Keyboard.
- A working Monitor.
- An SMD Controller.
- An SMD Drive.
- A boot device (i.e. local disk, local tape or remote disk over Ethernet).

CAUTION **You must run this diagnostic from a system other than the one under test.**

17.3. Set-Up Procedures

Before you execute this diagnostic, you should check to see what drive and controller type the program is set up to test, and make changes if necessary. These paragraphs provide information on how to change the test parameters and how to enter a command line that deals with such things as multiple disk testing.

First of all, this diagnostic is set up with these default parameters:

c0d0=fuj2322
ct0=xy450
cylinder=*the disk diagnostic cylinder — change at your own risk*
track=0
retry=1
pass=1

These values may be changed any time during the testing process. To check on the disk types that Sun supports at this time, enter

p c*number***d***number***=?** Return

number may be any numeric value

To check on the present test parameters, simply enter **p** from the main menu.

Controller Selection

The following examples show you how you can specify a certain controller number and type. Any of these commands may be mixed together on the same command line or used separately.

To select the controller under test, you first enter a command from one of the SMD diagnostic menus, followed with the controller number and type, as shown below:

command **ct***number***=xy7053**

command may be any command except **?**.

number is the controller number to be tested (0 - 3). If you issue multiple **ct***number***=** commands, you may test more than one controller at once.

Disk Selection

To select the disk under test, you first enter a command from any of the menus (excluding the ? command), followed by the controller and disk number and the disk type, as shown below:

command c*number*d*number*=*disk type*

As described for controller selection, the c*number* and d*number* entries may be any number from 0 - 3. The c represents the controller under test and the d represents the disk. The *disk type* parameter may be any disk that Sun supports. To view possible choices, use the command:

p c*any number*d*any number*= **?** Return

Option Selection

To select SMD diagnostic test options, use this command:

command **option test= cylinder= track= pass= retry=**

command may be any command except ?, from any of the SMD menus. The test parameter selects a special test in a given sequences of tests. For example, if *command* was s for the seek tests, the *test* argument could be:

```
test = 1   sequential seek test
test = 2   Long seek test
test = 3   Oscillating seek test (hourglass or butterfly)
test = 4   Random seek test
test = 5   Seek timing test
```

For descriptions of these tests, refer to the *Drive Tests Menu* section in *Chapter 17.*

Another example of test numbers that might be entered when using the Option command is when calling up the ECC test, using e in place of *command*, where *test* is the test number and is as follows :

```
test = 0   ecc pattern test 1 - takes about 6 minutes
test = 1   ecc pattern test 2 - takes about 40 minutes
test = 2   ecc pattern test 3 - takes about 6 HOURS
```

These tests are described in the this chapter under the *Controller Tests Menu* section.

The cylinder and track parameters to the Option command specify which area of the disk the test will write to and read from. pass specifies the number of passes the program or test will execute, and retry determines the number of times the I/O routine (driver) will try to complete an unsuccessful operation.

CAUTION **When changing the cylinder to any cylinder other than the default, diagnostic cylinder, you risk destroying data that may be stored on that cylinder.**

17.4. Main Menu

The main menu provides access to the sub-menus. It also contains a fast and default test command. The main menu is shown below:

```
SMD Subsystem Exerciser    REV 1 xx/xx/xx  Main Menu

Controller          Controller tests menu
Drive               Drive tests menu
All                 All test sequence
Burnin              Burn-in Controller tests
Quick               Quick test sequence
Utilities           Utilities Menu
PARameters          Enter Configuration Parameters

?                   Display the command syntax of this menu

Command:
```

C*ontroller*

The *Controller tests menu* command brings up the controller sub-menu. Enter **c** to bring up this menu.

D*rive*

The *Drive tests menu* command brings up the drive sub-menu. Enter **d** to bring up this menu.

B*urnin PASs=*

The *Burn-in Controller tests* command automatically executes a set controller tests that don't require an SMD disk. This test is useful for burning in the controller (with no disk connected) in a burn-in oven. Enter **b** and *pass=* followed with the number of passes you want the tests to make. No `pass=` entry means that the tests run once.

This command executes the following sub-tests:

1. Controller Self Test.
2. DMA Test.
3. Buffer Load and Dump Test.

A*ll pass=*

The *All tests* command executes all the SMD tests in their proper sequence. You can control the number of times this command is executed. The syntax is:

a *PASs=some number*

pass= is the number of times this command should execute. If no count is given, the program runs only once.

The `all tests` command executes the following tests:

- Registers Test
- NOP Test
- Controller Diagnostic Test
- Controller Maintenance Test
- Interface Test
- Label Test
- Addressing Test
- Seek Test
- Switch Test
- Pattern Test
- ECC Test

Quick

The *Quick test* command executes a subset of the SMD tests. This subset takes less time, yet still provides good test coverage. You can control the number of times the Quick test is executed. Enter

q *PASs=some_number*

pass is the number of times the command executes. If no value is given, the program runs only once.

The `quick tests` command executes the following tests:

- Registers Test
- NOP Test
- Controller Diagnostic Test
- Controller Maintenance Test
- Interface Test
- Label Test (Primary Label is tested)
- Addressing Test
- Seek Test (Sequential Seek test only)
- Switch Test
- Pattern Test

PAR*ameters*

This command allows you to enter controller parameters. Enter **par** and the desired change, such as `CTO=xy451` or `C0D0=Fuj2333`.

?

The *Help Command* displays the syntax of all the commands. Enter **?** to invoke Help.

17.5. Controller Tests Menu

The tests below check the SMD Controller. You should run these tests first, to find out if the controller is working.

```
SMD Subsystem Exerciser    REV 1 xx/xx/xx  Controller Tests Menu

Register          Registers test
Nop               Nop test
Diagnostic        Controller Diagnostic tests
Controller        Controller Maintenance tests
Ecc               ECC test
All               All Controller Tests Sequence
PARameters        Enter Configuration Parameters
? - Display the command syntax for this menu

Command :
```

n *PASs=*

The *NOP test* commands the controller to read the IOPB and mark it complete. Then the test sends a NOP command to the controller and checks the return status.

pass is the number of times the given test is executed. If no value is given, the test runs only once. The disk must be connected to the controller for this test.

r *PASs=*

The *Registers test* reads and writes different patterns to all writable/readable registers then verifies them.

The command syntax is :

r *PASs=some_number*

pass is the number of times the test is executed. If no value is given, the test runs only once. The SMD disk does not need to be connected to the controller for this test.

c *PASs=*

The *Controller maintenance tests* command executes the DMA test, the load and dump test, the IOPB addressing test, and the interrupt test. The SMD disk needs to be connected to the controller for this test.

The DMA Test uses the DMA command. The patterns used for the DMA test are: 0x0, 0x55, 0x77, 0xAA, 0xCC, and 0xFF.

The Load and Dump test uses the Buffer Load and Buffer Dump commands. It runs the following patterns: 0x0, 0x55, 0x77, 0xAA, 0xCC, and 0xFF.

The IOPB Addressing test uses the NOP command. It tests all the address bits of IOPB address registers.

The Interrupt test sends a command to the controller after setting the interrupt bit. The program finds out if the interrupt occurred through the interrupt handler.

pass is the number of times the test is executed. If no value is given, the test runs only once.

- d *PASs=*
  The *Controller Diagnostic tests* executes the controller's self test.

  pass is the number of times the given test should be executed. If no value is given, the test runs only once.

- e *TEst= PAS=*
  The *Ecc test* command runs the ECC (Error Checking and Correction) test. The disk should be attached for this test because it writes and reads from diagnostic disk cylinder. This test is not part of any default test sequence. There is a mode test in which all four possible ECC correction modes are checked. The test checks ECC circuitry three ways. It alternates, using 1's as a background and 0's for the ECC correction, then 0's are the background and 1's as the ECC correction.

Test "0" steps are:

1. Write all 1's to a sector using the `write` command (to the diag cylinder).
2. Read the entire sector using the `read all` command.
3. Write 11 bits of 0's, starting from the first bit of the sector.
4. Read the same sector and check the status bits.
5. Read the status bits to see if the ECC circuit worked properly.
6. Starting from the 12th bit, repeat steps 2 through 5.
7. Do steps 1 through 6 for all 512 bytes so that all the bits are covered.
8. Do steps 1 through 7 with 0's as background and 1's as ECC pattern.

Test 1 is similar to Test 0 except it does an 11-bit ECC bit by bit. Test "1" steps are:

1. Writes all 1's to a sector using the `write` command (to the diag cylinder).
2. Read the whole sector using the `read all` command.
3. Write 11 bits of 0's from the first bit of the sector.
4. Read the same sector and check the status bits.
5. Check the status bits to see if the ECC circuit worked properly.
6. Starting from the 2nd bit repeat steps 2 to 5 (do all bits - 8 times).
7. Repeat steps 1 through 6 for all 512 bytes; now all the bits are covered.
8. Repeat steps 1 through 7 using 0's as the background and 1's as the ECC pattern.

Test "2" is more extensive than the previous two. It starts with a 1-bit ECC pattern and works up to an 11-bit pattern. The ECC patterns from 1 bit to 11 bits are done on each bit of the sector. The steps are similar to Tests 0 and 1.

test is the test number, as follows:

Table 17-3 *Test Parameter Values*

Value	*Test*
0	ECC pattern test 1 - takes about 6 minutes
1	ECC pattern test 2 - takes about 40 minutes
2	ECC pattern test 3 - takes about 6 HOURS

The default test is "0". The argument `pass=` is the number of times the given test is executed. If no value is given, the test runs only once.

a

The *All tests* command executes all the tests given in this menu, in the sequence given below:

- Registers Test
- NOP Tests
- Controller Diagnostic Tests
- Controller Maintenance Tests

You can set the number of times the tests are executed. The default is one. The syntax is :

a *PASs=some_number*

pass is the number of times you want to execute the test. If no value is supplied, the test runs only once.

PAR*ameters*

This command allows you to enter controller parameters. Enter **par** and the desired change, such as `CTO=xy451` or `C0D0=Fuj2333`.

?

The *Help* command displays the syntax of each command in the menu. Enter **?** to get help with the commands.

17.6. Drive Tests Menu

These tests check the SMD drive: It tests drive interface and drive operations. There are eight commands in this menu. they are :

```
SMD Subsystem Exerciser   REV 1 xx/xx/xx Drive Tests Menu

Interface        Interface test
Label            Label test
Addressing       Addressing test
Seek             Seek test
SWitch           SWitch test
Pattern          Pattern test
ALl              ALl tests
Quick            Quick tests
PARameters       Enter Configuration Parameters
? - Display the command syntax for this menu

Command:
```

i *PASs=*

The *Interface test* makes sure the drive is connected and the cables between the drive and controller are good. *pass* is the number of times the test is executed. If no parameters are given, the test runs only once.

l *Num= PASs=*

The *Label Test* checks label of the Disk to see if it is good or not. It checks the primary label (at sector 0 and track 0) and alternate labels (at sectors 1, 3, 5, 7, and 9 of the second alternate). This test accepts two arguments. If no *Num* values are given, the test checks all the labels once.
The argument *num* is for individual labels as shown below:

Table 17-4 *Num Parameter Values*

Value	*Meaning*
0	all the labels
1	primary label (sector 0, track 0)
2	alternate label (sector 1, second alternate)
3	alternate label (sector 3, second alternate)
4	alternate label (sector 5, second alternate)
5	alternate label (sector 7, second alternate)
6	alternate label (sector 9, second alternate)

The second argument, *pass*, is the number of times the test will execute. If no values are given, the test runs only once.

a *CYLinder= TRAck= OPTion=Sector Start= End= PASs=*

or

a *OPTion= Cylinder Start= End= PASs=*

The *Addressing test* checks the addressing of cylinders and sectors by addressing even sectors (and cylinders) from first to last, then addressing odd sectors (and cylinders) from last to first. The test accepts four or six arguments.

- *cylinder* is the cylinder number.
- *track* is the track number.
- *option* specifies whether the test is for a cylinder or sector.
- *start* is the starting sector/cylinder number to test.
- *end* is the last sector/cylinder number to test.

  If no values are given, the test checks the sectors and cylinders, going from minimum to maximum number.

  pass is the number of times the test executes. The default value is one.

s

The *Seek test* performs a number of different seek tests on the SMD drive. There are five different tests. They are explained below :

Sequential Test
: This test seeks to the starting cylinder number, then does seeks in increments of one to last cylinder number. It then starts seeking from the last cylinder number to the first cylinder number in decrements of one.

Long Seek Test
: This test seeks to the starting cylinder number, then seeks to the last cylinder number. Then it does the reverse, seeking to the last cylinder number, then seeking to the first cylinder number.

Oscillating Seek Test
: This test seeks from the starting cylinder to the next cylinder, and back again. Then it seeks two cylinders up, then back, repeating the cycle to the top cylinder. Next it starts doing the reverse; starting from the top cylinder, it seeks down one then back up, down two then up, until it reaches the starting cylinder.

Random Seek Test
: This test does random cylinder seeks between the starting and ending cylinder numbers. The program gets the cylinder number from a random number generator.

Timing Seek Test
: This test consists of four timing measurements:

  The Average Minimum Seek (forward) test seeks the head sequentially (forward) over the center 100 cylinders and the average is displayed.

  The Average Minimum Seek (reverse) test is identical to the previous test, except that it seeks in the reverse direction.

  The Average Seek test does 100 average seeks and the average is displayed.

  The maximum seek test does 100 maximum seeks and then displays the average.

NOTE *Software overhead has NOT been removed for these timing tests.*

sw

The *SWitch test* checks the switching of heads and cylinders. The syntax is:

sw *OPTion=Cylinder Start= End= PASs=*

or

sw *OPTion=Head Start= End= PASs=*

This test accepts four arguments. *option* specifies whether cylinders or heads are tested. If no value for *option* is supplied, the test is executed for heads and cylinders.

start is the starting cylinder or head number to test.

end is the last cylinder or head number to test. *pass* is the number of times to execute the test (default is one).

p

The *Pattern test* writes and reads different patterns to the SMD drive. The test uses the patterns in the table below:

Table 17-5 *Pattern Test Values*

Pattern Values
0xAAAAAAAA
0xFFFFFFFF
0xEBD6EBD6
0xD7ADD7AD
0xAF5BAF5B
0x5EB75EB7
0x6DB66DB6

The *Pattern* test uses the diagnostic cylinder to do a write/read operation in the default mode. Each pattern is written to the disk, read back and compared. All errors are reported. If any mode other than the default mode of operation is selected, it is *VERY probable* that data on the disk will be destroyed. "Other than default mode" means that you select which cylinder or track is to be used. All cylinders other than the diagnostic cylinder MAY have data stored on them, and as a result this data would be lost.

This command accepts five or seven arguments. The command syntax is:

p *CYLinder= TRAck= OPTion=Sector Start= End= PATtern= PASs=*

or

p *OPTion=Cylinder Start= End= PATtern= PASs=*

- *cylinder* is the cylinder number.
- *track* is the track number.
- *option* specifies whether a cylinder or a sector is tested.

- *start* is the starting sector or cylinder number you want to test.
- *end* number is the end sector or cylinder number you want to test.

  If no sector or cylinder number is provided, the test runs the pattern test for all available sectors with default patterns one time.
- *pattern* is the pattern used to test on the disk.
- *pass* is the number of times the test runs (default is one).

The default test is a non-destructive test on every cylinder using its six default patterns.

al *PASs=*

The *ALl tests* command executes all the tests in this menu in the sequence below:

1. Interface Test
2. Label Test
3. Addressing Test
4. Seek Test
5. Switch Test
6. Pattern Test

You can set the number of times the tests are executed. The default is one. The syntax is:

al *PASs=xxxx*

pass= is the number of times the test executes. If no arguments are supplied, the test runs once.

q *PASs=*

The *Quick test* completes faster than the *All tests* but performs only limited testing. It runs the following tests:

1. Interface Test
2. Label Test (Primary label is tested)
3. Addressing Test
4. Seek Test (only Sequential part)
5. Switch Test
6. Pattern Test

pass= is the number of times the test executes. If no arguments are supplied, the test runs once.

PAR*ameters*

This command allows you to enter controller parameters. Enter **par** and the desired change, such as `CTO=xy451` or `C0D0=Fuj2333`.

?

The *Help* command displays the syntax of each command. Enter **?** to get help with command syntax.

17.7. Utilities Menu

The Utilities Menu for an SMD drive will look something like this:

```
SMD Subsystem Exerciser    REV 1 xx/xx/xx Utilities Menu

Dump              Dump
RHeader           Read Headers
Read              Read
Write             Write
Parameters        Enter Configuration Parameters
?                 Display the command syntax for this menu

Command:
```

This menu provides limited support for the SMD subsystem through the use of four utilities. Some of these utilities are rough and will not always display or handle data in the manner desired because they were originally designed to be used in debugging this diagnostic. Due to the length of some of the displays, all utilities may be aborted with the Esc key.

The options on the Utilities Menu are used as follows:

D *cylinder= track=*

The dump utility "dumps", or displays, the contents of the disk referenced when you entered the P command, or of the disk specified in a command line. The complete sector is displayed as it is transferred from the disk, which includes header, data, and ECC. As a result, the header may not make sense at first, because the cylinder, track and sector are not separated in this release. This will be corrected at a later date.

The *cylinder* and *track* parameters are optional.

CAUTION **The cylinder and track information used by this utility is the same information used by the diagnostic. Therefore, if reference is made to an area other than the diagnostic cylinder, it should be changed before returning to the test menus. This procedure will prevent the destruction of disk data.**

RH *cylinder= track=*

The Read headers utility displays only the header information referenced by the values displayed when using the P command, or those included as part of a command line. The complete header is displayed as transferred from the disk, and as a result it may not make sense at first, because the cylinder, track, and sector information are not separated for easy viewing. This will be corrected at a later date.

The *cylinder* and *track* parameters are optional. Refer to the CAUTION message above.

R *cylinder= track=*

This utility displays only the disk data information referenced by the values

displayed when using the `P` command, or those included as part of a command line. The sector data for a complete track is displayed as transferred from the disk.

The *cylinder* and *track* parameters are optional. Refer to the **CAUTION** message above.

W *cylinder= track=*

This utility writes data to the track referenced by the values displayed when using the `P` command, or those included as part of a command line. The complete track is written.

The *cylinder* and *track* parameters are optional. Refer to the **CAUTION** message above.

P *ct#= c#d#= cylinder= track= pattern= pass=*

The `Enter Configuration Parameters` utility prints the values of the existing parameters on the screen, or allows you to enter new ones.

The type of controller, controller and drive configuration, cylinder, track, pattern, and pass count are all optionally entered parameters. The program has a default value for all these parameters. The defaults are:

ct1=xy450
c0d0=fuj2322
cylinder= *(this value is controlled by the disk being used)*
track= *(this value is controlled by the disk being used)*
pattern= *(if none is issued all are used — refer to Table 17-5)*
pass=1

When a disk parameter is entered, the cylinder and track information is extracted from a table containing that information for all the disk units that the diagnostic presently supports.

Entering `p` with no arguments displays the present parameters.

CAUTION **The cylinder and track information used by this utility is the same information used by the diagnostic. Therefore, if reference is made to an area other than the diagnostic cylinder, data on that cylinder could be destroyed during testing.**

17.8. Controller Errors and Their Interpretation

The table below lists Xylogics' 450/451 error numbers in hexadecimal. The numbers are more fully explained in the Xylogics user's manual. These numbers are printed when the program fails at the driver level, causing the status of the IOPB to be printed.

Table 17-6 *Xylogics 450/451 Error Numbers (in Hex)*

Number	*Type*	*Meaning*
00	N/A	successful completion
01	hard	interrupt pending
03	hard	busy conflict
04	soft	operation timeout
05	hard	header not found
06	hard	hard ECC error
07	hard	illegal cylinder address error
09	soft	sector slip command error
0A	hard	illegal sector address
0D	hard	last sector too small
0E	hard	slave ACK error (non-existent memory)
12	hard	cylinder & head header error
13	soft	seek retry required
14	hard	write protect error
15	hard	unimplemented command
16	hard	drive not ready
17	hard	sector count zero
18	hard	drive faulted
19	hard	illegal sector size
1A	hard	self test A
1B	hard	self test B
1C	hard	self test C
1E	hard	soft ECC error
1F	soft	soft ECC error recovered
20	hard	illegal head error
21	hard	disk sequencer error
25	hard	seek error

The table that follows lists the error conditions, and their decimal numbers, detected by the program.

17.9. Program Reported Errors

```
000     DRIVE NOT READY.
001     DISK CONFIGURED FOR ONLY [dd] SECTORS.
002     IT MUST BE ABLE TO HANDLE AT LEAST [dd] DATA SECTORS!
003     iopb address test failed at physical address = [xxxxxx], virtual address = [xxxxxx].
004     No action, status only, error status = [xx].
005     Non-retryable programming error, error status = [xx].
006     Non-existing error code, error status = [xx].
007     Successful recovered soft error, error status = [xx].
008     Hard error / retry, error status = [xx].
009     Non-existing error code, error status = [xx].
010     Hard error / Reset and retry, error status = [xx].
011     Fatal Hardware error, error status = [xx].
012     Miscellaneous error, error status = [xx].
013     Requires Manual intervention, error status = [xx].
014     Driver failed, error = [xx], retry = [d].
015     Bus error (reset or read), controller bad or not installed.
016     Operator aborted.
017     Controller reset failed, CSRT failed to clear.
018     Controller failed to set RIO or clr BUSY, timeout. CSR = [xx], Cmd = [xx], subfun = [xx].
019     Fatal controller error  fatal error reg = [xx].
020     Controller failed,  DONE not set in IOPB.
021     BUFFER LOAD failed in command chaining.
022     Driver, Controller is busy, Timeout.
023     driver failed, error = [xx], cyl = [dddd], head = [dd], sector = [dd], retry = [d].
024     DMA Could not be done for pattern = [xx], error = [xx].
025     DMA Test failed at iopb byte no. [dd], expected / observed = [xx] / [xx].
026     failed, error = [xx], iopb chaining, retry = [d].
027     Minimal seek test forward failed (Timing), cyl = [ddd].
027     Minimal seek test reverse failed (Timing), cyl = [ddd].
028     Acceptable strings for this test for 'option' are 'cylinder' or 'sector'.
029     cylinder number given is out of range.
030     start is out of range.
031     end is out of range.
032     Controller/disc initialization failed.
033     Cylinder address test failed (reset, Read track header, or Read all).
034     Cylinder address test failed :expected / observed = [dd] / [dd].
035     Sector address test failed (reset, Read track header, or Read all).
```

Program Errors, Continued

```
036     sector address test failed : cylinder expected / observed = [dd] / [dd].
037     sector address test failed : track expected / observed = [dd] / [dd].
038     sector address test failed : sector expected / observed = [dd] / [dd].
039     Controller diagnostic test failed (Self Test).
040     NOP Test Failed.
041     (BUFLOAD or BUFDUMP) Test failed.
042     BUFFER DUMP_LOAD FAILED, at addr = [xxxxxx], expected / observed = [xx] / [xx].
043     Could not allocate memory as required.
044     Addressing Test failed.
045     Could not deallocate memory.
046     Average maximum seek test failed (Timing), cyl = [ddd].
046     Average maximum seek test (forward) failed (Timing), cyl = [ddd].
046     Average maximum seek test (reverse) failed (Timing), cyl = [ddd].
047     exinstall failed.
048     Installed interrupt failed to interrupt.
049     exremove failed.
050     Interrupt test failed.
051     Acceptable values for variable 'test' are '0' to '2'.
052     Ecc test compares failed, status = [xx], I-J-K = [d], [d], [d].
053     Ecc test failed (Read Track header, Write, Read, or Write all).
054     No good sectors on track 0 diag cylinder, ecc test stopped.
055     Mode = [d], Ecc test failed (Write, Readall, or Write all).
056     Ecc test failed (Read with ecc mode 0, 1, 2, or 3), status = [xx].
057     Ecc test failed (Read with ecc mode 2), expected = 0xFF, read = [xx].
058     Ecc test failed, pattern is (1's or 0's), status = [xx].
059     Ecc test failed, pattern is (1's or 0's).I-J-K = [d], [d], [d].
060     Ecc test failed, pattern is (1's or 0's). I-J = [d], [d].
061     Average seek test forward failed (Timing), cyl = [ddd].
061     Average seek test reverse failed (Timing), cyl = [ddd].
062     CORRUPT LABEL!!
063     MISPLACED LABEL!!
064     NO PRIMARY LABEL.
065     No backup label found.
066     No logical partitions.
067     Acceptable values for variable 'num' are '0' to '6'.
068     Label test failed :Primary label is corrupted.
069     Label test failed :secondary label - (1, 2, 3, 4, or 5) is corrupted.
070     label test failed (READ).
071     NO SECONDARY (ONE, TWO, THREE, FOUR, or FIVE) LABEL.
072     Pattern test failed (reset, Write, or Read).
073     Pattern test failed, head = [dd], sector = [dd], long word = [d], expected/observed = [xx] ,
074     Buss error (reset, read, or write), controller bad or not installed.
075     Controller reset failed, CSRT failed to clear.
076     Register test failed, IOPB addr reg (0, 1, 2, or 3) expected / observed = [xx] / [xx].
```

Program Errors, Continued

```
077     Register test failed, IOPB addr mod reg, expected / observed = [xx] / [xx].
078     Register test failed, error reg, expected / observed = [xx] / [xx].
079     Register test failed, relocation (low or hig)h byte, expected / observed = [xx] / [xx].
080     Register test failed, address (low or high) byte, expected / observed = [xx] / [xx].
081     Register write failed, Bus error. addr = [xx], value = [xx].
082     Register write failed (register and bit), Bus error.
083     Register read failed, Bus error.
084     Given value for option 'start' is out of range.
085     Given value for option 'end' is out of range.
086     Seek test failed (Oscillating), cyl = [dddd].
086     Seek test failed (Sequential), cyl = [dddd].
086     Seek test failed (Random), cyl = [dddd].
086     Seek test failed (Timing), cyl = [dddd].
086     Seek test failed (Long), cyl = [dddd].
087     Interface Test failed (status, reset or set drive size).
088     Allowable strings for this test for 'option' are 'cylinder'
        or 'head'.
089     given value for option 'cylinder' is out of range.
090     Cylinder switch test failed (reset).
091     Cylinder switch test failed - (1, 2, or 3) (Read).
092     Switch Test failed, cyl = [xxxx], byte = [xx], expected / observed = [xx] / [xx].
093     Acceptable values for variable 'test' are '0' to '5'.
094     Xy450 controller and disc are not xfer rate compatible.
095     Wrong controller name is given for ct[d].
096     Wrong drive name is given for c[d]d[d].
097     Acceptable values for variable 'pass' should be in decimal.
097     Acceptable values for variable 'retry' should be in decimal.
097     Acceptable values for variable 'register' should be in hex.
097     Acceptable values for variable 'pattern' should be in hex.
097     Acceptable string for variable 'dual' is Dual.
097     Acceptable values for variable 'test' should be in decimal.
097     Acceptable string for variable 'option' are 'cylinder', 'track', 'sector', or 'all' only.
097     Acceptable values for variable 'cylinder' should be in decimal.
097     Acceptable values for variable 'Track' should be in decimal.
097     Acceptable values for variable 'start' should be in decimal.
097     Acceptable values for variable 'end' should be in decimal.
098     IOPB chaining test failed.
```

17.10. Diagnostic Variables

Parameters can be given on any command line. A command uses only the relevant ones. Parameters not given are loaded from a set of diagnostic variables. These variables retain their values until the diagnostic exits. They are set to default values when the diagnostic starts.

The list below shows the diagnostic variables, and to what values you can set them. The capital letters given for a variable are the minimum letters that you must type.

CT0 - CT3=xy*controller_type*
This variable sets the type of the first SMD controller. *controller_type* might be `xy450` for a Xylogics 450, `xy451` for a Xylogics 451 controller, or `xy7053` for a Xylogics 7053 controller.

C0 - C3D3=*ascii_string*
This variable sets the drive type for controller zero, drive zero. The

following Fujitsu drive names might replace *ascii_string*: `fuj2312, fuj2284, fuj2322, fuj2351, fuj2333, fuj2361, fuj2294, CDC9720, NEC2363, HIT815-10,or none.`

PASs=*decimal_number*
This variable sets the number of times the test runs (in decimal).

DATa=*hex_number*
The variable sets the hexadecimal pattern to be written to a register.

TEst=*decimal_number*
This variable sets the sub-test to run. This number is dependent on the individual command. The relevant numbers are described in the command information.

OPTion=*ascii string*
This variable specifies where a disk test should be done on a "Cylinder", "Track", "Sector" or "All" three. Its meaning varies between tests.

CYLinder=*decimal_number*
This variable determines the cylinder on which the test is performed.

TRAck=*decimal_number*
This variable determines the track or head on which the test is performed.

STart=*decimal_number*
This variable determines the starting cylinder, track, or sector on which the test is performed.

ENd=*decimal_number*
This variable determines the ending cylinder, track, or sector on which the test is performed.

17.11. Glossary

Cylinder
Most common disk devices contain a number of Platters mounted on a spindle spinning at a high speed. Cylinders are the tracks at a certain radius (from the center of the spindle) on all platters. Disks store information on a thin magnetic coating.

Head
To read and write information on the surface of the disk, a number of heads are mounted on a common arm so they travel together. Usually, there are two heads for each platter(one for the top and one for the bottom surfaces).

Mbytes
1,048,576 (approximately one million) bytes.

Sector
Each track is divided into equal segments called sectors. A sector is the basic unit of storage on the disk. On Sun systems, each sector contains 512 bytes of data, and is surrounded by a header and trailer containing addressing and error correction information.

SMD
Storage Module Device.

Track

The portion of a disk passing under a stationary head while the disk is rotating is called a track. It is similar to a track on a record album.

Cylinders, tracks and sectors each inhabit a distinct dimension that separates and locates them; cylinders are located by distance from the spindle, tracks are separated from each other by surface location on platters, and sectors are separated from each other by time.

18

1/2-Inch Tape Diagnostic

18

1/2-Inch Tape Diagnostic

18.1. General Description

The 1/2-inch Tape Subsystem Diagnostic tests all of the 1/2" tape transports currently supported by Sun, and supports the Xylogics 472 Tape Controller Board. This diagnostic does not support the Tapemaster tape controller board.

18.2. Hardware Requirements

In order to run this diagnostic, you must at least have the following in your system:

- A working Sun CPU board.
- A working Keyboard.
- A working Monitor.
- A 1/2" tape controller (Xylogics 472).
- A 1/2" tape transport (Fujitsu M2444 or CDC 92181).
- A boot device (local disk, local tape or remote disk through Ethernet).
- A 1/2" scratch tape.

NOTE *Both the Fujitsu 2444 and the CDC 92181 have built in diagnostics that can be actuated from their front panel. Try executing these tests before running this diagnostic.*

18.3. Set-Up Procedures

There are parameters that should be checked and possibly changed prior to running the Tape Diagnostic. The program defaults for the type of tape drive and controller are:

ct0=xy472 (*the first controller is a Xylogics 472*)
c0d0=fuj2444 (*the tape drive is a Fujitsu 2444*)
retry=1 (*an unsuccessful test will be repeated once*)
pass=1 (*the test will be executed once*)

These values may be changed at any time during the testing process. The parameters are divided into *controller selection, transport selection* and *options selection.*

Controller Selection

To select the controller(s) to be tested, use a command similar to this:

command **ct***number*=**xy472**

command may be any command from any of the Tape Diagnostic menus, except ?. *number* may be any controller number, from 0 - 3. Use multiple **ct***number*= commands to configure more than one tape drive for testing at the same time.

Transport Selection

To select the transport to be tested, use a command like this:

command **c***number***d***number*=*drive type*

command may be any command from any of the Tape Diagnostic menus, except the ? command. **c***number* may be any controller number from 0 - 3. **d***number* may be any transport number, from 0 - 7. Use multiple **c***number***d***number* commands to select multiple drives. The value for *drive type* may be either fuj2444 or cdc92181.

Test Options

To select test options, use a command such as:

command **option mode= pass= retry=**

command may be any command from any of the tape diagnostic menus, except for ?. **mode=** determines the method of writing and may be pe or gcr. **pass=** determines the number of passes the program or test will execute. **retry=** determines the number of times the I/O routine (driver) will try to complete an unsuccessful operation.

18.4. Menus

Commands are displayed in the form of menus. Each menu handles commands for a different group of tests. Each menu has a help command (?) to provide syntax information, and *Chapter 2* explains how to use the Diagnostic Executive command line syntax to invoke diagnostics and set parameters from the Exec level.

This chapter describes Tape Subsystem Diagnostic Main Menu options first, followed with Sub-Menu descriptions.

Main Menu

The main menu provides access to the sub-menus. It also contains a fast and default test command. The main menu is shown below:

```
1/2" Tape Subsystem Diagnostic REVx.x xx/xx/xx  Main Menu

All             All test sequence
Controller      Controller tests menus
Transport       Transport tests menu
Only            Controller test only
Utilities       Utilities menu
Quick           Quick test sequence
Parameters      Parameters display only
?               Display the command syntax of this menu

Command:
```

NOTE *The device parameters can be changed in any command line. Once set, a parameter remains fixed until set to a new value by another test option. This is true for both default and user selected values. The* `parameters` *menu selection allows you to change drive and controller parameters.*

Valid parameters for Main Menu commands are `PASs=` and `Mode=`.

PASs= is the number of times the command executes. If no pass parameter is given, the program executes once (default).

Mode sets the recording method used. If no mode is selected, the program uses the method presently in use, or the default value, phase encoded (`PE`). A list of valid *Mode=* entries follows:

Valid Modes	
`PE`	Phase Encoded mode
`GCR`	Group Code Recording mode
`BOTH`	PE and GCR mode

The contents of the main menu are described below. When part of a parameter entry is shown in upper case letters, you need enter only those letters, followed with the appropriate value.

a *PASs= Mode=*

The *All tests* command executes all the 1/2-inch tape tests except the All, Quick and Parameters tests.

CAUTION **Do not specify more than one pass at a time when you run this diagnostic; the diagnostic will not function if you do so.**

Refer to the table that follows the Main Menu example for a list of valid *Mode=* entries.

c The *Controller tests menu* command displays the controller menu, from which you may select controller tests. Simply enter **c** to bring up this menu.

t The *Transport tests menu* command displays the tape transport menu, from which you may select transport tests. Simply enter **t** to bring up this menu.

o *PASs=*

The *controller Only test* command executes only the controller tests that don't require any hardware other than the controller. It executes the controller's on-board diagnostics, NOP test, Register test, and Controller addressing test.

u

The *Utilities menu* command displays the utility menu, which contains routines used for debugging and repairing the 1/2-inch tape subsystem. Simply enter **u** to bring up this menu.

q *PASs= Mode=*

The *Quick test* command performs all of the tests specified by `All` command, but executes them in a shorter time period. You may enter the number of times the Quick test is executed.

Refer to the table that follows the Main Menu example for a list of valid *Mode=* entries.

p*ct#= c#d#=*

The *Parameters menu* command displays the tape system parameters in use. Enter **p** to view the parameters.

The "#" symbol represents a number that you are to enter; DO NOT enter the "#" symbol.

You must press a key on the keyboard to make the program continue after the parameters are displayed.

To set the controller number to be tested, enter:

p ct0=xy472 *for the first Xylogics 472 Controller board*

Enter **ct1=xy472** for the second controller board. At this time, the program only accepts `xy472` as the tape controller board parameter.

To set the drive number and type, enter:

c0d0=fuj2444 *for the first Fujitsu 2444 Tape Drive*

Replace *c0d0* with *c0d1*, *c0d2*, or *c0d3*, depending on which drive you are specifying. At the time of this writing, the program accepts only

```
fuj2444
or
cdc92181
```

as drive types.

?

The *help* command displays the syntax of all the commands. Simply enter **?** to receive help with commands and their parameters.

Controller Tests Menu

The following tests are offered for a 1/2-inch tape controller:

```
1/2" Tape Subsystem Diagnostic  REV x.x xx/xx/xx  Controller Tests Menu


All             All controller tests sequence
Register        Registers test
Nop             Nop test
Diagnostic      Controller diagnostic tests
Controller      Controller addressing tests
Quick           Quick controller test sequence
Parameters      Parameters display only

?               Display the command syntax for this menu

Command :
```

You may add a *Pass=* parameter to all Controller Test Menu commands except for the `Parameters` command. *Pass=* is the number of times the test is executed. The default number of passes is one.

a *PASs=*
The *All tests* command executes all the tests shown in the Controller Test menu, except for the Parameters and Quick tests.

r *PASs=*
The *Registers test* command verifies the addresses of the in and out registers.

n *PASs=*
The *Nop test* command makes the controller read the I/O process block (IOPB) and mark it complete.

d *PASs=*
The *Controller diagnostic tests* command executes the controller's on-board diagnostics — if they are enabled.

c *PASs=*
The *Controller addressing test* command executes the addressing test, which tests the address lines using NOP IOPB.

p*ct#= c#d#=*

The "#" symbol represents a number that you are to enter; DO NOT enter the "#" symbol.

The *Parameters menu* command displays the tape system parameters in use. Enter **p** to view the parameters. You must press a key on the keyboard to make the program continue after the parameters are displayed.

To set the controller number to be tested, enter:

p ct0=xy472 *for the first Xylogics 472 Controller board*

Enter **ct1=xy472** for the second controller board. At this time, the program only accepts `xy472` as the tape controller board parameter.

To set the drive number and type, enter:

dt=*c0d0=fuj2444* *for the first Fujitsu 2444 Tape Drive*

Replace *c0d0* with *c0d1*, *c0d2*, or *c0d3*, depending on which drive you are specifying. At the time of this writing, the program accepts only

`fuj2444`
or
`cdc92181`

as drive types.

q *PASs=*
The *Quick test* command performs the tests specified by the `All` command, but executes them faster. You can set the number of times the Quick test is executed.

?
The *help* command displays the syntax for each command. To invoke `help`, enter **?**.

Transport Tests Menu

The menu for testing a 1/2" tape transport is shown below:

```
1/2" Tape Subsystem Diagnostic  REVx.x mm/dd/yy  Transport Tests Menu

All             All transport tests
Data            Data test
Handler         Tape handler test
Parameters      Parameter display only
Quick           Quick transport test
?               Display the command syntax for this menu

Command:
```

Valid entries for and descriptions of the parameters shown with each command are:

PASs= sets the number of times the test executes. If no value is given, the test runs once (by default). *Mode=* sets the recording method used. If no mode is selected, the program uses the current method, or the default of `PE`. Refer to the table following the Main Menu example for valid `mode=` entries.

The *PATtern=* argument sets the fixed pattern used during testing. You may enter any data pattern that does not exceed 32 bits. If no pattern is specified, the default of all patterns is used.

This menu tests the tape transport interface and operations. There are five commands in this menu. They are:

a *PASs= Mode=*
The *All tests* command first executes the Data test, followed with the Tape Handler test. Refer to the description following the Transport Tests Menu

for information on parameter entries.

d *Mode= PATtern= PASs=*

The *Data test* command writes, then reads, varying or fixed data patterns of a fixed block size, using the current recording mode. Refer to the description following the Transport Tests Menu for information on parameter entries.

h *Mode= PASs=*

The *tape Handler test* command checks the positioning of tape. This includes such things as "skip # blocks, skip # file marks", and so on.

p *ct#= c#d#=*

The *Parameters menu* command displays the tape system parameters in use. Enter **p** to view the parameters. You must press a key on the keyboard to make the program continue after the parameters are displayed.

To set the controller number to be tested, enter:

```
p ct0=xy472        for the first Xylogics 472 Controller board
```

Enter **ct1=xy472** for the second controller board. At this time, the program only accepts `xy472` as the tape controller board parameter.

To set the drive number and type, enter:

```
dtc0d0=fuj2444     for the first Fujitsu 2444 Tape Drive
```

Replace *c0d0* with *c0d1*, *c0d2*, or *c0d3*, depending on which drive you are specifying. At the time of this writing, the program accepts only

```
fuj2444
or
cdc92181
```

as drive types.

q *PASs= Mode=*

The *Quick test* command performs all of the tests specified by `All` command, but executes them in a shorter time period. You can enter the number of times the Quick test is executed.

?

The *help* command displays the syntax of each command. To get help with a command, enter **?**

Utilities Menu

The menu below contains the utilities routines used for debugging or repairing the 1/2" tape subsystem.

```
1/2" Tape Subsystem Diagnostic  REVx.x mm/dd/yy  Utilities Menu

Nop             NOP instruction
Write           Write tape
EOF             Write EOF
Read            Read tape
Erase           Erase tape
Space           Space tape
REWind          Rewind tape
Unload          Unload tape
RESet           Reset tape
STatus          Status tape
Mode            Change modes
Diagnostics     Controller diagnostics
Parameters      Parameter display only
?               Display the command syntax for this menu

Command:
```

Parameters for the Utilities Menu are:

PASs=
: which sets the number of times the test is executed. If no options are selected, the current parameter values are used.

PATtern=
: sets the data pattern written to tape. You may enter any data pattern that does not exceed 32 bits.

Mode=
: sets the recording mode used. If no mode is selected, the default mode is used. Refer to the table following the Main Menu example for valid mode= entries.

The commands described below perform special testing and scoping:

n*PASs=*
: The *Nop* command performs a NOP loop until the pass count is reached. Refer to the paragraphs following the Main Menu example for information on parameter entry.

w *PATtern= PASs= Mode=*
: The *Write* command writes to the tape until the pass count is zero. *PATtern* sets the data pattern written to tape. *PASs* sets the number of records written if EOT is not reached. The operation terminates when EOT is detected. *Mode* sets the recording mode used. If no mode is selected, the default mode is used. Refer to the table following the Main Menu example for valid *Mode=* entries.

eof *PASs= Mode=*
: The *Write EOF* command writes file marks on the tape from its present position until the pass count is reached. This routine uses two parameters.

PASs= sets the number of times an EOF is written if EOT is not detected. *Mode*= sets the recording mode used. If no mode is selected, the default value is used. Refer to the table following the Main Menu example for valid *Mode*= entries. If no options are selected, the current parameter values are used.

r *PASs*=

The *Read* command reads the contents of the tape from its present position until the pass count is reached. The data read should have been written with the write or write EOF routines. This routine uses only one parameter. *PASs* sets the number of records to be read before terminating. If the number of passes is not entered, the current value is used.

e *PASs*=

The *Erase tape* command erases the tape one record at a time until the pass count is reached. *PASs*= sets the number of erasures done. The default is one erasure.

s *PASs*= *Dir*=

The *Space tape* command spaces the tape, one record at a time, until the pass count is reached.

PASs= sets the number of records to skip. If pass isn't set, only one record is skipped (by default).

Dir= (direction) sets whether the tape will space forward or backward.

Enter **dir=fwd** to space forward.

Enter **dir=rev** to space backward, or reverse direction.

rew

The *Rewind tape* command rewinds the selected tape.

u The *Unload tape* command unloads the tape unit selected.

res *PASs*=

The *Reset tape* command resets the selected tape unit.

PASs= sets the number of resets issued.

st *PASs*=

The *Status tape* command displays the status of the selected tape unit. *PASs*= sets the number of status requests made. You must press a key on the keyboard to make the program continue after the status is displayed.

m *PASs*= *Mode*=

The *Mode select* command sets the mode to the value selected in the `Mode` option.

PASs= sets the number of times the test runs. If the number of passes is not set, the default is one.

Mode= sets the recording method used. If the `mode` is not set, the program uses the current method. Refer to the table the follows the Main Menu example for valid *Mode*= entries.

d *PASs=*

The *Controller diagnostics* command activates the tape controller on-board diagnostics.

PASs= sets the number of times the diagnostics execute. If not set, the tests runs once (by default).

p The *Parameter* command displays the current tape system parameters. Enter **p** to view the parameters. You must press a key on the keyboard to make the program continue after the parameters are displayed.

?

The *help* command displays the syntax of each command. Enter **?** for information on commands and their parameters.

18.5. Error Reporting

Errors are reported through the Diagnostic Libraries. The program displays an error message, then logs it — if error logging is selected. All error messages are in "English text", such as:

```
35, controller test, read, data compare error, expected xxxxx, read yyyyyy.
```

The fields in the message shown above are described in the paragraphs that follow.

`35:`
An index number associated with the message to allow a lookup table for foreign languages (future option).

`controller test:`
The test that failed.

`read:`
The operation the test was performing.

`data compare error:`
The type of error detected.

`expected` *xxxxxx:*
The data expected (good).

read yyyyyy:
The data actually read (bad).

Error Messages

The table that follows lists all the error message numbers possible in this diagnostic. The description indicates the type of error encountered.

Table 18-1 *Tape Diagnostic Error Messages*

Error #	*Error text*
01	Nop failed.
02	status failed.
03	Reset failed.
04	Rewind failed.
05	Space reverse failed.
05	Space forward failed.
06	Write failed.
07	Unload failed.
08	Write EOF failed.
09	Read reverse failed.
09	Read forward failed.
10	Erase failed.
11-19	Reserved.
20	Controller, Controller failed to complete reset, Timeout.
21	Address test failed.
22	Command does not exist.
23-29	Reserved.
30	Controller is busy, Timeout.
31	Controller is busy, Timeout.
32	Driver failed, error = 0x, cmd = 0x, subfun = 0x, cnt = d, retry = d
33	Driver(Interrupt Test), Controller is busy, Timeout.
34	Driver(Interrupt Test), Controller is busy, Timeout.
35	Driver failed, error = 0x, retry = d.
36	Driver, Controller is busy, Timeout.
37	Driver, Controller is busy, Timeout.
38	Driver failed, error = 0x, retry = d.
39	Reserved.
40	NOP Test Failed.
41	Controller diagnostic test failed (Self Test).
42	Register test failed, relocation low byte, expected/observed = x/x.
43	Register test failed, relocation high byte, expected/observed = x/x.
44	Register test failed, address low byte, expected/observed = x / x.
45	Register test failed,address high byte, expected/observed = x / x
46	Register test failed, csr, err, ipnd, ans areq bits, expected / observed = x / x.
47-49	Reserved.
50	Tape handler test failed (rewind).
51	Tape handler test failed (Write record).
52	Tape handler test failed (rewind).
53	Tape handler test (space records) failed.
54	Tape handler test failed (Write EOF).
55	Tape handler test failed (rewind).
56	Tape handler test (space files) failed.
57	Tape handler test failed (write record), record # = d.
58	Tape handler test failed (rewind).
59	Tape handler test failed (read forward).

Table 18-1 *Tape Diagnostic Error Messages— Continued*

Error #	*Error text*
60	Tape handler test failed (read forward), expected = 0x, read = 0x, word cnt = d.
61	Tape handler test failed (read reverse).
62	Tape handler test failed (read reverse), expected = 0x, read = 0x, word cnt = d.
63	Tape handler test failed (space test).
64	Tape handler test failed (read forward).
65	Tape handler test failed (space record), expected = 0x, read = 0x, word cnt = d.
66	Tape handler test failed (space test), record cnt = 0x.
67	Tape handler test failed (read forward).
68	Tape handler test failed (read reverse), expected = 0x, read = 0x, word cnt = d.
69	Tape handler test failed (write EOF).
70	Tape handler test failed (write record).
71	Tape handler test failed (rewind).
72	Tape handler test failed (file space test), file cnt = 0x.
73	Tape handler test failed (read forward).
74	Tape handler test failed (read), expected = 0x, read = 0x, word cnt = d.
75	Tape handler test failed (file space test), file cnt = 0x.
76	Tape handler test failed (read forward).
77	Tape handler test failed (read), expected = 0x, read = 0x, word cnt = d.
78	Tape handler test failed (rewind).
79	Reserved.
80	Write/read data test failed (rewind).
81	Write/read data test failed (Write).
82	Write/read data test failed (Read reverse).
83	Write/read data test failed (compare), word count = 0x, expected/observed = 0x / 0x.
84	Write/read data test failed (rewind).
87	Write/read data test failed (compare), 88
89	Reserved.
90	Could not allocate 33k memory as required.
91	Addressing Test failed.
92	Could not deallocate 33k memory.
93	Interrupt auto vector 37 install, failed.
93	Interrupt 3 install, failed.
94	Interrupt auto vector 36 install, failed.
94	Interrupt 3 install, failed.
95	Interrupt test failed.
96	Interrupt removal failed in Interrupt test.
97	Unable to set drive parameters.
98-99	reserved.

Procedural Error Messages

The list below shows all of the error messages that result from user input error. They are self-explanatory.

```
Acceptable variable for 'pass' should be in decimal.
Acceptable variable for 'pattern' should be in hex.
Acceptable variables for 'direction' are FWD or REV.
Acceptable variable for 'mode' are 'pe', 'gcr', or 'both'.
Acceptable variables for 'ct0' is 'xy472'.
Acceptable variables for 'ct1' is 'xy472'.
Acceptable variables for 'c#d#' are' fuj2444' or 'cdc92181'.
```

Xylogics 472 status codes

The table below lists the current status codes and their meaning as given by the vendor. The codes and their definition are subject to change by the vendor at any time, and were correct at the time this document was prepared.

Table 18-2 *Xylogics Tape Controller Status Codes*

Code	*type*	*Definition*
00	Status	Successful completion - No errors.
01	Hard	Interrupt pending.
02	N/A	Reserved.
03	Hard	Busy conflict.
04	Hard	Operation timeout.
05	N/A	Reserved.
06	Hard	Uncorrectable data.
07-0D	N/A	Reserved.
0E	Hard	SLave ACK error (Non-existent memory).
0F-13	N/A	Reserved.
14	Hard	Write-protect error.
15	Hard	Unimplemented command.
16	Hard	Drive off-line.
17-19	N/A	Reserved.
1A	Hard	Self test A failed.
1B	Hard	Self test B failed.
1C	Hard	Self test C failed.
1D	Hard	Tape mark failure.
1E	Hard	Tape mark detected on read.
1F	Status	Corrected data.
20-21	N/A	Reserved.
22	Hard	Record length short.
23	Hard	Record length long.
24-29	N/A	Reserved.
30	Hard	Reverse into BOT.
31	Hard	EOT detected.
32	Status	ID burst detected.
33	Hard	Data late detected.

18.6. Glossary

Head
There are two types of heads: write/read and erase. The write/read head writes and reads the magnetic tape. The erase head erases only. It always erases the magnetic tape before it is written by the read/write head.

Mbytes
Megabytes (1024 kilobytes).

IPS
Inches per second, the speed at which the tape moves across the tape head.

Track
The portion of the tape passing under the tape head and running the length of the tape. Generally there are nine tracks on a 1/2" tape.

PE
Phase encoded, a method used to read and write magnetic tape.

GCR
Group code recording, a method used to read and write magnetic tape.

Pertec
A hardware interface developed for peripherals by Pertec Inc., most commonly used on tape systems. The other standard used for tape interfaces is the STC interface, developed by Storage Technology Corp.

IOPB
I/O process block, contains the operation to be executed and the buffer to be used if needed.

19

Sun Video Diagnostic

19

Sun Video Diagnostic

19.1. General Description

The Sun Video Diagnostic tests color video ONLY when the color circuitry is present on the CPU board, as on-board circuitry or as a daughter board. To test systems with a color board that resides in a separate slot, run either the Color2 or Color3 Diagnostic.

The Sun Video diagnostic provides a tool to operate the video section of the Sun processor boards in near normal conditions for failure detection and isolation. It provides flexible tests for debugging as well as the easy-to-use default sequence tests. You can quickly determine whether the video section is functional. If you discover failures, focus on the problem area with the individual device tests and the associated test loops.

The way the test patterns are presented provides flexibility in test sequencing and parameter setting. All tests can be broken down so that primitive actions can be performed on command, which is useful for isolating problems when debugging boards. At a higher level, a default test sequence is invoked with a single command character.

Map of Video Frame Buffers

The Video frame Buffers on the various Sun-3 products have the following characteristics:

- **Sun-3/160 or 3/180** — has a single Monochrome Frame Buffer that is 128K Bytes long (0x00 - 0x20000), Beginning at address 0xff000000.
- **Sun-3/50** — has a single monochrome Frame Buffer that is 128K Bytes long (0x00 - 0x20000), beginning at address 0x00100000.
- **Sun-3/60** — has three frame buffers: a single, monochrome frame buffer, 128K (256K) bytes long, 0x00 - 0x20000 (0x00 - 0x40000), beginning at 0x1F000000; an enable plane, 128 Kbytes long (0x20000) beginning at address 0x1F600000; and a color plane 1 Mbyte long (0x100000) beginning at 0x1F800000. The color map begins at address 0x1F200000.
- **Sun-3/2XX** — has a single monochrome frame buffer that is 256K bytes long (0x0 - 0x40000), high resolution, beginning at 0xff000000.
- **Sun-3/110** — has three (3) frame buffers: a single monochrome frame buffer, 128K Bytes long (0x00 - 0x20000) beginning at address 0xff000000; an enable plane, 128K Bytes long (0x20000), beginning at address 0xfe400000; and a color plane 1 Mbyte long (0x100000), beginning at

0xfe800000. The enable plane selects either the monochrome display when filled with all ones (0xff), or the color display, when filled with all zeroes (0x00).

The color map, an array of three (3) 256 Byte long (Red, Green, Blue), eight (8) bit wide locations, is located at address 0x000e0000, 0x000e0100, 0x000e0200 for Red, Green and Blue, respectively.

19.2. Menus

When testing any Sun-3 system other than the Sun-3/60 or Sun-3/110, the Frame Buffer Menu comes up in place of the Main Menu, and represents the only test choices for those systems. The Main Menu for the 3/60 and 3/110 includes a color pattern test menu and a status bit test, in addition to the Frame Buffer Menu. The Video diagnostic first reads the CPU board's IDPROM to determine on which system it is loaded. It then loads the appropriate menu.

Monochrome frame buffer testing is used in most Sun-3's, and the color frame buffer testing is used for the combination of color and monochrome type frame buffers in the Sun-3/110 and 3/60.

Here is an example of the Main Menu for Sun-3/60 or 3/110:

```
Sun Video Diagnostic  Rev x.x  mm/dd/yy 3/110 or 3/60 Main Menu

    All         All Test Sequence
    Default     Default Test Sequence
    FB          Frame Buffer Menu
    Pattern     Pattern Test Menu
    Status      Status Bit Test (First Half & Too Late)

Command ==>
```

Following are descriptions for the Sun-3/110 and 3/60 Main Menu choices:

All

The *All Test Sequence* runs the frame buffer tests automatically. This command runs the following tests in sequence:

```
A ; C ; NTA ; R ; U
```

As they run, the tests record any errors in the error log and print error messages on the screen.

Default

The *Default Test Sequence* automatically runs a set of frame buffer tests. This command will run every test in sequence, recording any errors it encounters in the error log, and printing error messages on the screen. This default test sequence is for all Sun-3's except the Sun-3/110 and 3/60 systems:

This default test sequence is for Sun-3/110 and 3/60 systems only:

```
FB ; A ; C ; NTA ; U ; R ;          frame buffer ram tests
P ;                                 start patterns
Solid  (All Color Sequence '#8')    solid patterns
Stripe (All Color Sequence '#8')    shaded stripes
Block  (All Color Sequence '#8')    Block Pattern
```

The *Frame Buffer Menu* command displays the diagnostic's Frame Buffer Menu, described later.

Pattern

The *Pattern Menu* command displays the diagnostic's Pattern Menu, described later.

Status (Sun-3/60 and 3/110)

The *Status Bit Test* reads the status registers to make sure they are being set during the vertical retrace cycle. If the vertical retrace period is in its first half, the **First Half** bit should be set. When the retrace has progressed too far into its cycle to begin a write to the Color Maps in the DAC chip, the **Too Late** bit should set. The test waits in a loop until one of these two bits are set or a time-out occurs.

The `Help` Command

The *Help* command displays a message describing the commands in the main menu.

19.3. Sun-2 Main Menu

When the Sun Video Diagnostic runs on a Sun-2/160, the menu heading names the Sun-2/50; the two CPUs are functionally the same.

On Sun-2 systems, the Main Menu looks something like this:

```
Sun Video Diagnostic Rev xx.x dd/mm/yy  Sun2 Video Main Menu

All             All Test Sequence
Default         Default Test Sequence
Control         Control Register Menu
Serial          Serial Communication Controller Menu
Video           Video Memory Menu
```

The Sun-2 Video Menu `All Test Sequence` and the `Default Test Sequence` run all the Control Register, Video Memory and SCC internal options.

The `Control`, `Serial`, and `Video` entries from the Sun-2 menu bring up sub-menus, shown on the following pages.

Video Control Register

If you enter **c** from the Sun-2 Video Main Menu, this sub-menu is offered:

```
Sun Video Diagnostic Rev xx.x mm/dd/yy Control Register Menu

All             All Test Sequence
Default         Default Test Sequence
Memory          Memory Test
Jumper          Jumper Test
Screen          Screen    Enable/Disable Test
Copy            Copy      Enable/Disable Test
Interrupt       Interrupt Enable/Disable Test
```

The video control register is used to enable or disable various functions. The starting address for the control register is 0x781800 (hex) for Multibus systems and 0x020000 (hex) for VME systems. Each one of the Control Register tests is described as follows:

Memory
: Bits 01-06 correspond to the bus address lines 17-22. This test writes values to bits 01-06 of the control register and reads them back to do a comparison to check for any type of a data mismatch.

Jumper
: This option displays bits 08-11, which are the configuration jumpers.

Screen
: This test enables and disables the video display bit (15). The screen will blink off and on. The processor writes to this bit reads it back to check that it is set. If the bit is one, the video display is on, and if it is zero the display is completely black.

Copy
: This test enables and disables the copy-enable bit. The Copy mode enables read-modify cycles to a main memory shadow buffer to write to the video board memory. The values in both main and video memory are then checked for consistency.

Interrupt
: This test generates a video interrupt and reports whether nor not it has occurred. The interrupt bit is bit 13. An interrupt is caused by setting bit 13 to one, or to zero if no interrupt is desired. If bit 13 is set, the interrupt status bit, bit 12, will get set on the next interval and the video board will issue an interrupt. Bit 12 stays set until the processor resets it.

Serial Communications Controller(SCC)

The SCC is a Zilog 8530 serial communications controller used to communicate with the keyboard and the mouse. The keyboard is known as channel A and the mouse is known as channel B.

If you enter **s** from the Sun-2 Video Main Menu, this sub-menu is offered:

```
Sun Video Diagnostic    Rev xx.x   mm/dd/yy SCC Menu

All             All Test Sequence
Default         Default Test Sequence
Internal        Internal SCC Test
Within          Within Channel External SCC Test
Between         Between Channel External SCC Test
```

Each of the SCC tests are described as follows:

Both the keyboard and the mouse channels of the SCC are tested at baud rates of 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400 and 76800.

Internal
: The internal loopback test of the SCC is executed.

Between
: The external loopback test of the SCC is executed. It checks for data flow between channels A(B) and B(A).

Within
: The external loopback test of the SCC is executed. A loopback test is performed on both channels.

Video Memory

The video memory is addressed with a megabyte of address. The base address is 0x700000 for Multibus systems and 0x000000 for VME systems.

If you enter **v** from the Sun-2 Video Main Menu, this sub-menu is offered:

```
Sun Video Diagnostic  Rev xx.xx  mm/dd/yy Frame Buffer Menu

All             All Test Sequence
Default         Default Test Sequence
ADdress         Address Pattern Test
Constant        Constant Pattern Test
NTA             NTA Pattern Test
Random          Random Pattern Test
Unique          Uniqueness Pattern Test
```

The Video Memory tests are the same as the frame buffer tests for Sun-3 systems. For descriptions of these menu selections, refer to the the *Frame Buffer Menu* section.

19.4. Frame Buffer Menu

This test menu may be brought up from the Sun-3/110 and Sun-3/60 Main Menu. It automatically comes up *instead of the main menu* when testing any other Sun-3 system. The menu looks something like this:

```
Sun Video Diagnostic Rev X.X mm/dd/87 Frame Buffer Menu

    All         All Test Sequence
    Default     Default Test Sequence
    Address     Address Pattern Test
    Constant            Constant Pattern Test
    NTA         NTA Pattern Test
    Random      Random Pattern Test
    Unique      Uniqueness Pattern Test

Command ==>
```

The following text describes the Frame Buffer Menu choices.

All

The *All Test Sequence* runs the frame buffer tests automatically. This command runs the following tests in sequence:

```
A ; C ; NTA ; U ; R
```

As they run, the tests record any errors they encounter in the error log, and print error messages to the screen.

Default

The *Default Test Sequence* runs a set of the frame buffer tests automatically. This command will run every test in the diagnostic in sequence, recording any errors it encounters in the error log, and printing error messages to the screen. systems:

Address

The *Address Pattern Test* checks the specified block of memory using the low order address bits of each location, or its complement, as data. This test runs in byte, word, or long word mode. The command syntax is:

Address *Offset= Datamode= Pass=*

The default offset is 0, at the beginning of each frame buffer under test. The size of the monochrome plane is 0x20000 (128K Bytes) or 0x40000 (256K Bytes) for the Sun-3/2XX. The size of the monochrome plane on the Sun-3/60 can be either 128 Kbytes or 256 Kbytes. The *datamode* parameter determines whether a memory test will run in byte, word, or long word mode. The syntax is:

Datamode= *0=byte/1=word/2=long*

The default setting is `Datamode=2`.

The default number of passes is one.

Constant

The *Constant Pattern Test* checks the specified block of memory by using the specified data pattern. The memory block is filled with data, then read back and compared with the specified data pattern. If the data read from an address location does not match the original data pattern, an error is flagged. This test runs in byte, word, or long word mode. The command syntax is:

Constant *Offset= Pattern= Datamode= Pass=*

Defaults are:

Default Offset=0; beginning of each Frame Buffer under test.
Default pattern=0xa5; values 0x00 - 0xffffffff are acceptable.
Default datamode=2; 0=byte, 1=word 2=long
Default pass=1.

The size of the monochrome plane is 0x20000 (128K Bytes) or 0x40000 (256K Bytes) for the Sun-3/2XX. The size of the monochrome plane on the Sun-3/60 is either 128 Kbytes or 256 Kbyates.

NTA *Offset= Pass=*

The *NTA* test detects stuck-at faults, coupling faults, and pattern sensitivity faults in the memory under test. The test goes through 4 steps to verify memory. The table below shows the values the NTA test writes into memory. Each cycle is a pass through memory, starting at the high end of memory and running down, or starting at the low end of memory and running up. Some steps only read the values; others read, then change the values.

Table 19-1 *Values used in NTA Test*

Step	*Values (in Hex)*	*Read/Write*	*Start at*
0	*N/A* -> 0X00	W	Low
1	0x00 -> 0x01	R/W	Low
1	0x01 -> 0x00	R	High
2	0x00 -> 0x5F	R/W	Low
2	0x5F -> 0x11	R	High
3	0x11 -> 0xCC	R/W	Low
3	0xCC -> 0xDB	R	High
4	0xDB -> 0x6D	R/W	Low
4	0x6D -> 0xB6	R/W	High
5 .	0xB6 -> 0xFF	R/W	Low

Command syntax is:

NTA *Offset= Pass=*

Default Offset=0x00, at the beginning of each Frame Buffer under test.
Default Pass=1.

The size of the monochrome plane is 0x20000 (128K Bytes) or 0x40000 (256K Bytes) for Sun-3/2XXs.

Random *Offset= Datamode= Seed= Pass=*

The *Random Pattern Test* checks the specified block of memory using a sequence of random numbers generated from the specified seed. It uses the random number generator found in the "C" run-time library. A block of memory is filled with data, the random sequence is reseeded, then the data is read back and compared with what was originally written. If the data read from an address location does not match the original data pattern, an error is flagged. This test runs in byte, word, or long word mode.

Defaults are:

Default Offset=0x00; beginning of each Frame Buffer under test.
Default Datamode=2; 0=byte, 1=word 2=long
Default Seed=1; values 0x00 - 0x09 are acceptable.
Default Pass=1.

The size of the monochrome plane is 0x20000 (128K Bytes) or 0x40000 (256K Bytes) for Sun-3/2XXs.

Unique

The *Uniqueness Pattern Test* checks for address uniqueness. It runs in byte, word, or long word mode. The command syntax is:

Unique *Offset= Datamode= Seed=*

Defaults are:

Default Offset=0x00; at the beginning of each Frame Buffer under test.
Default Datamode=2;0=byte, 1=word 2=long
Default Seed=1;values 0x00 - 0x09 are acceptable.

The size of the monochrome plane is 0x20000 (128K Bytes) or 0x40000 (256K Bytes) for Sun-3/2XXs.

Patterns (Sun-3/60, 3/110 Only)

The main menu presented when a Sun-3/60 or Sun-3/110 is tested provides a `Pattern` menu choice. These patterns aid in debugging the hardware, using the video frame buffer RAM. Software has no access to this circuitry, so you must watch for failures. The `Mono` and `Color` tests are included in this menu to check the ability of the Sun-3/60 and 3/110 to set the enable plane for either monochrome or color.

While displaying the patterns, the color map is updated during the video blanking period. No flashing colors should be visible during changes.

The color parameters are listed in the table below:

Table 19-2 *Color Values*

Value	*Color*
-	
0	black
1	red
2	green
3	blue
4	yellow
5	cyan
6	magenta
7	white

```
Sun Video Diagnostic   Rev X.X   mm/dd/87    Pattern Menu


   All              All Test Sequence
   Default          Default Test Sequence
   Color            Color Frame Enable
   Mono             Monochrome Frame Enable
   SOlids           Solid Color Plane Test
   STripes          Venetian Stripe Color Test
   Block            Block Color Test

Command ==>
```

`All`

The *All Test Sequence* runs the pattern tests automatically. This command runs the following tests in sequence:

```
C ; CC ; M ; CM ; SO ; ST ; B
```

As they run, the tests record any errors they encounter in the error log, and print error messages to the screen.

Default

The *Default Test Sequence* runs a set of the pattern tests automatically. This command runs every test in the diagnostic in sequence, recording any errors it encounters in the error log, and printing error messages to the screen. The default test runs the same set of tests as the *All* command.

Color

The *Color Frame Enable* command enables the color frame; the enable plane is filled with 0's. To use this command, enter **Color**.

Mono

The *Monochrome Frame Enable* command enables the monochrome frame buffer; the enable plane is filled with 1's. To enable the monochrome buffer, enter **Mono**.

Solid

The *Solids* command initializes the color frame buffer to 0. Then the color frame is enabled by writing 0's to the enable frame. Next, the color map is initialized according to the color parameter selected. Using this default value, each of the seven colors are displayed. To use this command, enter: **Solid** *color*.

The default color = 8. (Refer to the color/number chart in the previous section.)

Stripe

The *Stripes* command fills the color frame buffer with an address pattern and enables the color frame buffer. Then the appropriate color map is filled with an address pattern according to the primary (increasing) and secondary (decreasing) color parameters selected. To use this command, enter:

Stripe *Color Color2*

The defaults are:

Default Color=8.
Default Color2=1.

Refer to the color chart in the previous section.

Block

The *Blocks* command selects the Color plane. A 12x12 box pattern is written into the frame buffer and the color map is updated with a special pattern that spreads the colors out fairly evenly. Next a random number generator selects random parts of the color map and distributes the color boxes across the color plane, looping 16 times. To use this command, enter:

Block *color*

Defaults are:

Default color=8.

Refer to the color chart in the previous section.

19.5. Glossary

- **Frame Buffer** — Refers to the Memory (RAM) that the video circuitry uses to display the entire screen.
- **Status Bit(s)**— Sun-3/110 and 3/60 — status bits that are read by the diagnostic to determine if the vertical retrace cycle has started or is complete. Located at 0x000e300, one byte wide, only bits 6 & 7 are valid, the others are masked off.
- **Monochrome Plane** — The black and white or grayscale frame buffer.
- **Enable Plane** — Sun-3/60, 3/110 — 128 Kbyte long Frame Buffer.

  Located at address 0xfe400000 for 3/110.

  For Sun-3/60, the address is 0x1F6000000.
- **Color Plane** — Sun-3/60, 3/110 only — 1 Megabyte long frame buffer, eight (8) bits per pixel, located at address 0xfe800000 for 3/110. Located at 0x1f800000 for the 3/60. Used to index the color map locations.
- **Color Map** - 3 separate arrays, 256 bytes long, 8 bits wide, one each for Red, Green, and Blue, located at addresses 0x000e0000 to 0x000e02ff. The color map is located on the board in the DAC s, and occupies Type 1 space. For the 3/60, the color map address is 0x1f200000.

20

Sun Video Monitor Diagnostic

20

Sun Video Monitor Diagnostic

20.1. General Description

NOTE *This diagnostic is intended for use on a Sun monitor. It is NOT intended to be used to diagnose the display on a terminal that is attached to a CPU board Serial port.*

Before running this diagnostic, make sure that the EEPROM on the system CPU board is set correctly for the monitor type. To check these settings, you may use the EEPROM Editing Tool described in *Chapter 6* of this manual, selecting `SH` to view all the Primary Terminal Type, Monitor Resolution and High-resolution monitor column/row settings. To change settings, use `T` from the main menu for the console type, `R` to change the display resolution, and `H` to change the high-resolution monitor setting.

Or, you may wish to use the PROM monitor `q` command to view the contents of location 0x016 for the screen size; locations 0x050 and 0x051 for a high resolution monitor (usually found in Sun-3/2xx and Sun-4 systems); and location 0x01F, to select which device will act as the system console. The EEPROM Editing Tool is the easiest way to check or change EEPROM parameters; however, if you wish to use the `q` command from the monitor, access the PROM monitor, and then use the `q` command to check or change the parameters shown below:

NOTE *"0x" is not entered or displayed; it is used in Sun documentation to represent hexadecimal values.*

The value of location 0x016 will usually be "00", for the standard 1152 x 900 pixel screen. If you have a high resolution monitor, the value would be "13" for a 1600 x 1280 pixel screen size. Changing the display size of the monitor requires a hardware change on the CPU board in addition to EEPROM programming.

For a high resolution monitor, location 0x050 should contain "50", the hexadecimal representation of 80 columns, and location 0x051 should contain "22", the hexadecimal representation of 34 rows.

Depending on the device being used as the primary display, or console, one of these values should be in location 0x01F:

```
0x00    Use B/W Monitor
0x10    Use Serial Port A
0x11    Use Serial Port B
0x12    Use Color Monitor
0x20    Use ''first'' head of multi-headed P4 card
```

NOTE *If you attempt to run this diagnostic from a serial port, only the main Sun Video Monitor Diagnostic menu will appear on the terminal display. The "icon" menus shown on the following pages, from which you select the test patterns, are directed for output ONLY to a color or monochrome Sun monitor display.*

The Sun Video Monitor Diagnostic is a set of patterns designed to allow the quick and easy test and setup of monitors for Sun-2 and Sun-3 products. These patterns, along with a special template, allow you to quickly test a monitor for all the common parameters that affect its "usability" in our product. They also allow the set-up of certain important variables such as focus and linearity.

The objective of the Video Monitor Diagnostic is to provide a minimum set of patterns with which you can determine the acceptability of a monitor as a display for a Sun workstation. It also provides monitor adjustment capability for trained technicians.

The Video Monitor Diagnostic depends on the operator to decide the success or failure of the tests that are associated with the patterns.

20.2. Hardware Requirements

1. A Sun-2/3 system appropriate for the monitor being tested
2. Related hardware (cables; the diagnostic on a server, disk, or tape)
3. Ethernet transceiver cable and a transceiver box if using a server.

20.3. User Interface

The user interface of the Video Monitor Diagnostic consists of a menu of patterns displayed as icons. The pattern icons are arranged in a grid of 3 across by up to 7 down. You select a pattern, using the `R` and `L` function keys on the sides of the keyboard. The instruction box on the screen describes the use of the pattern select keys and other alphabetic keys.

There are a number of command keys that work in any of the pattern menus:

- The **`L`** command allows you to loop continuously through icons R1 - R4.
- The **`S`** command allows you to single step from R1 through R4. Stepping to the next icon is accomplished by pressing the space bar.
- The **`I`** command inverts a pattern's black and white components.

- The **R** command or the space bar returns you to the menu if a pattern is being displayed.
- The [Esc] command returns you to the main menu of the Exec.

When running this diagnostic from an off-board frame buffer that resides in a separate slot, such as that found on the Color2 or Color3 board, DO NOT use the "Control-C" sequence to exit; doing so will lock-up the Video Monitor Diagnostic displays. Use [Esc] instead.

- The ● selection either erases or redisplays the timer. The timer is a clock that appears in the lower right portion of the Video Monitor Diagnostic display:

  000:00:00:00

  During a test, the time shown indicates how long the pattern you are viewing has been on the screen. If you exit that test, the time shown represents the elapsed time since you selected one of the Monitor Diagnostic sub-menus shown on the following pages.

20.4. Standard Patterns

The standard patterns (1 - 4) appear in each of the menus (black and white, grayscale and color). These patterns are invoked with the R1 thru R4 selections in each of the menus.

Error Message

If you make an inappropriate choice (for example, the workstation has a color monitor and you choose the Monochrome Monitor Menu), an error message is displayed:

```
Invalid choice of monitor
```

The diagnostic then returns to the Main Menu.

20.5. Main Menu

The user interface of the Video Monitor Diagnostic consists of a main menu that allows you to select a sub-menu of icon patterns appropriate for the monitor being tested. The main menu gives you three choices: Monochrome Monitor Menu, Grayscale Monitor Menu, or Color Monitor Menu. The appropriate sub-menu is invoked by typing a minimum of the first letter of any submenu choice.

```
Video Monitor Diagnostic

Monochrome          Monochrome Monitor Menu

Grayscale           Grayscale Monitor Menu

Color               Color Monitor Menu


Command==>
```

Monochrome
Selecting *Monochrome* from the Main Menu brings up the Monochrome Menu. Use it only if the diagnostic is running on a monochrome monitor.

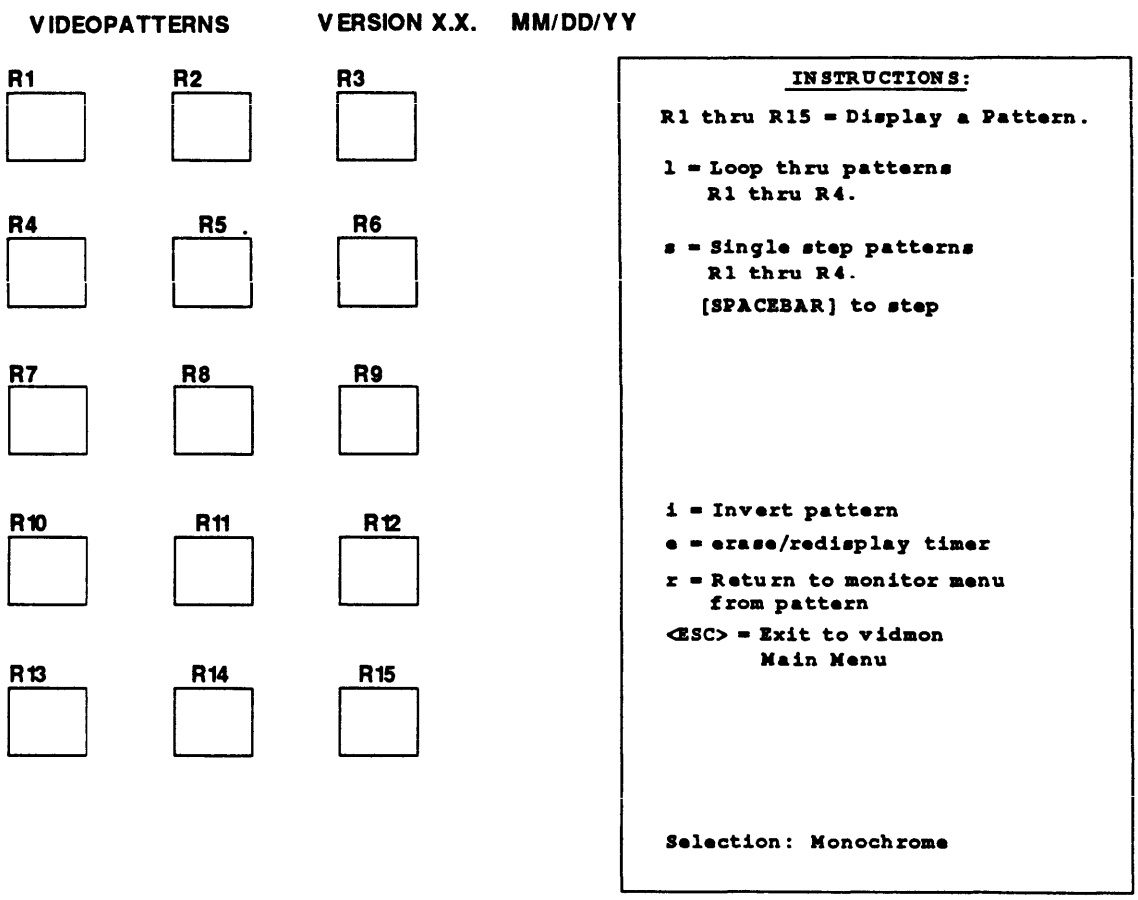
Grayscale
Selecting *Grayscale* from the Main Menu brings up the Grayscale Menu. Use it only if the diagnostic is running on a grayscale monitor.

Color
Selecting *Color* from the Main Menu brings up the Color Menu. Use it only if the diagnostic is running on a color monitor.

20.6. Monochrome Menu

A menu that looks something like this is displayed when you select `Monochrome` from the main Video Monitor Diagnostic menu. The squares will be filled with examples of the test patterns represented by R1 - R15.

Figure 20-1 *Monochrome Video Pattern Menu*



R1

The *Screen Geometry Test* verifies that the screen is properly aligned, the video amplifier is working correctly, and the horizontal/vertical scan amplifiers are operating correctly.

Description:

The pattern has a "cross" centered in the middle of the display area. The cross consists of thin lines separated by a short space. This tests for precision and sweep linearity. One-fourth of the distance from the center in each of the four legs of the cross are four perpendicular lines. They provide four boxes for testing horizontal and vertical centering.

In the left and right eighth of the screen are a set of gray, black, and white bars to test for video amplifier ringing. In the upper left quadrant and one-eighth of the way from the center is a horizontal black line with two thin white lines entering a wide, vertical, half-gray line. This pattern tests for

over/undershoot in the video amp. (On systems where there is no grayscale, the gray will be created by half-tone shading of the area, with every other pixel turned on and the adjacent pixel turned off.)

In the lower right quadrant of the screen is another horizontal black bar to maintain symmetry of the overall pattern. There are two white bars one-eighth of the way above and below the center of the screen. The lower bar penetrates halfway into the vertical gray bars and the upper bar completely covers the gray bars.

R2

The *Focus and Brightness Test* verifies that the focus controls are properly set and that the monitor can be focused. The peak white level is set by this test.

Description:

The pattern consists of m's (the character m) and e's (the character e) filling the screen with a three inch white box in the middle. The m's and e's provide 3 short vertical lines connected at the top by a short horizontal line. The legs of the m's and e's are one pixel in width and are spaced two pixels apart.

This pattern is used to set and check the focus of the monitor. The focus adjustment is closely related to correct brightness so both adjustments are done here. Use the white square in the middle to set the screen's brightness. Its presence doesn't affect the focus of the screen. Use the m's and e's to set the screen's focus. The worst case focus occurs at the edges of the monitor, where the electron beam strikes the phosphor at an angle. If the dots are correct at the edge and one-fourth of the way to the center, they should be correct at the center.

There is occasionally a problem with focus linearity. The circuit that corrects focus, depending on the location of the dot on the screen, may be defective. If this is a problem, you may have to use the outline pattern to set the screen brightness and fill the white box with m's and e's.

R3

The *Black & White Convergence Test* verifies that the beam convergence is properly set.

Description:

This pattern consists of 25 large squares (white lines on black) with a white dot in the middle of each square. Use the horizontal and vertical lines to make sure the beams from the color gun(s) converge properly. Correctly set beams produce pure white lines and dots. A multicolored pattern indicates poor convergence.

R4

The *Luminance Uniformity Test* checks for uniform screen luminance (brightness) and correct high voltage regulation.

Description:

The monitor displays a white screen.

R5 - R15

The *Voltage and Geometry tests* check screen voltages and screen dimensions

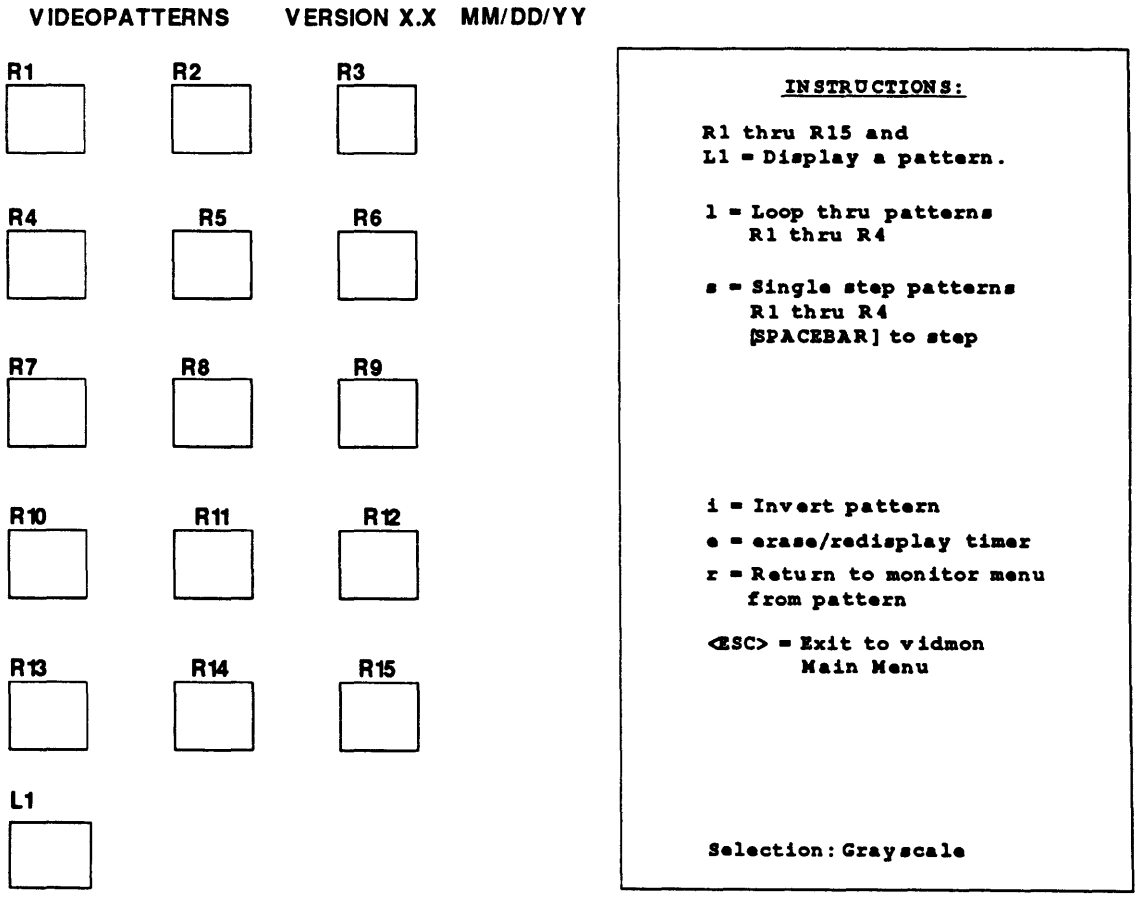
Description:

The patterns R5 thru R15 are useful for checking video shadowing (ringing), high voltage regulation, and screen geometry.

20.7. Grayscale Menu

A menu that looks something like this is displayed when you select `Grayscale` from the main Video Monitor Diagnostic menu. The squares will be filled with examples of the test patterns represented by R1 - R15 and L1.

Figure 20-2 *Grayscale Video Pattern Menu*



R1

The *Screen Geometry Test* verifies that the screen is properly aligned, the video amplifier is working correctly, and the horizontal/vertical scan amplifiers are operating correctly.

Description:

The pattern has a "cross" centered in the middle of the display area. The cross consists of thin lines separated by a short space. This tests for precision and sweep linearity. One-fourth of the distance from the center in each of the four legs of the cross are four perpendicular lines. They provide four boxes for testing horizontal and vertical centering.

In the left and right eighth of the screen are a set of gray, black, and white bars to test for video amplifier ringing. In the upper left quadrant and one-eighth of the way from the center is a horizontal black line with two thin white lines entering a wide, vertical, half-gray line. This pattern tests for over/undershoot in the video amp. (On systems where there is no grayscale, the gray will be created by half-tone shading of the area, with every other pixel turned on and the adjacent pixel turned off.)

In the lower right quadrant of the screen is another horizontal black bar to maintain symmetry of the overall pattern. There are two white bars one-eighth of the way above below the center of the screen. The lower bar penetrates halfway into the vertical gray bars and the upper bar completely covers the gray bars.

R2

The *Focus and Brightness Test* verifies that the focus controls are properly set and that the monitor can be focused. The peak white level is set by this test.

Description:

The pattern consists of m's (the character `m`) and e's (the character `e`) filling the screen with a three inch white box in the middle. The m's and e's provide 3 short vertical lines connected at the top by a short horizontal line. The legs of the m's and e's are one pixel in width and are spaced two pixels apart.

This pattern is used to set and check the focus of the monitor. The focus adjustment is closely related to correct brightness so both adjustments are done here. Use the white square in the middle to set the screen's brightness. Its presence doesn't affect the focus of the screen. Use the m's and e's to set the screen's focus. The worst case focus occurs at the edges of the monitor, where the electron beam strikes the phosphor at an angle. If the dots are correct at the edge and one-fourth of the way to the center, they should be correct at the center.

There is occasionally a problem with focus linearity. The circuit that corrects focus depending on the location of the dot on the screen may be defective. If this is a problem, you may have to use the outline pattern to set the screen brightness and fill the white box with m's and e's.

R3

The *Black & White Convergence Test* verifies that the beam convergence is properly set.

Description:

This pattern consists of 25 large squares (white lines on black) with a white dot in the middle of each square. Use the horizontal and vertical lines to make sure the beams from the color gun(s) converge properly. Correctly set beams produce pure white lines and dots. A multicolored pattern indicates poor convergence.

R4

The *Luminance Uniformity Test* checks for uniform screen luminance (brightness) and correct high voltage regulation.

Description:

The monitor displays a white screen.

R5 - R15

The *Voltage and Geometry tests* check screen voltages and screen dimensions

Description:

The patterns R5 thru R15 are useful for checking video shadowing (ringing), high voltage regulation, and screen geometry.

L1

The *Grayscale Luminance Test* checks the uniformity of the monitor display with uniformly varying shades of gray.

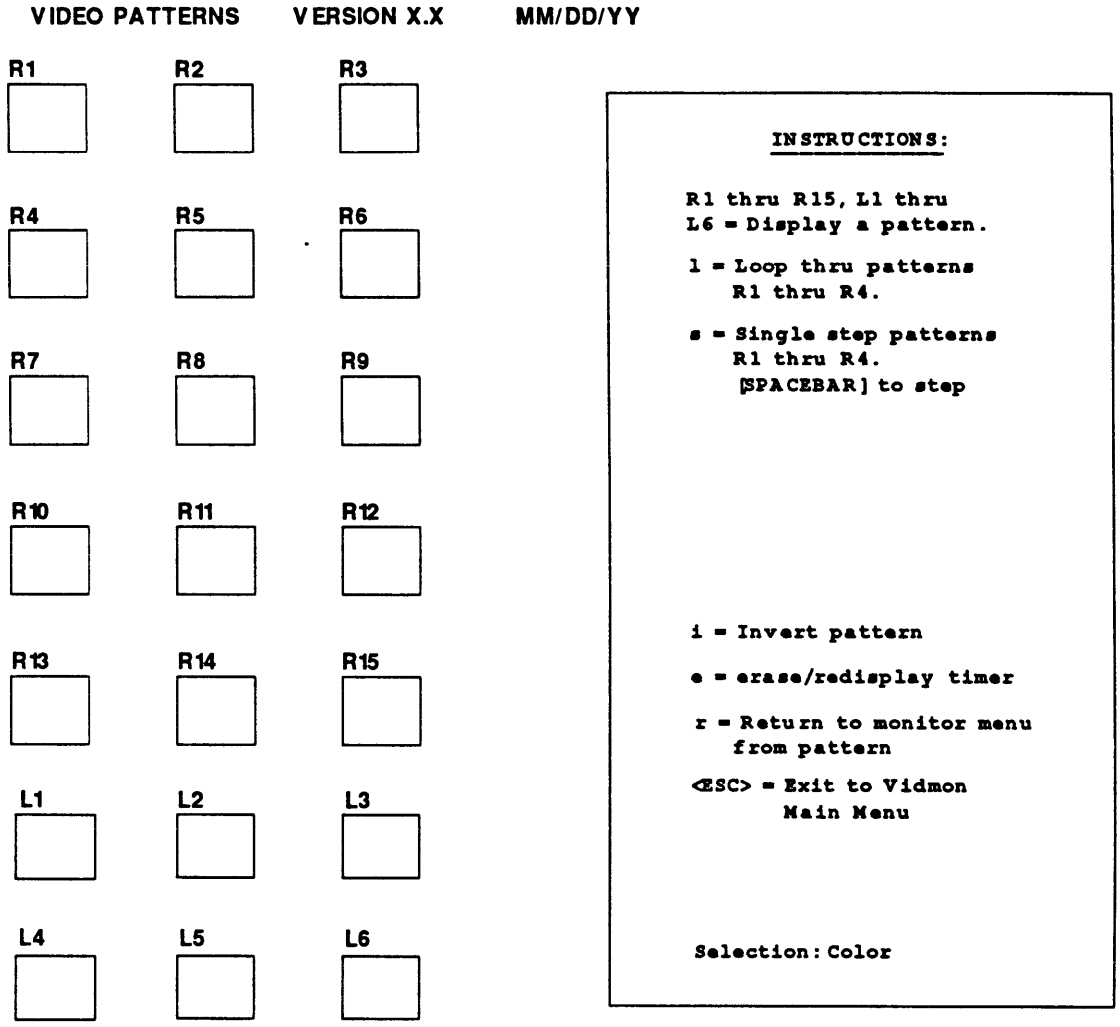
Description:

The pattern is made up of two rows of eight rectangles each on a gray background. The top row begins with a black rectangle. Each succeeding rectangle in the row has a uniformly lower saturation. The last rectangle in the row is white. The lower row of rectangles starts with a white rectangle. Each succeeding rectangle is more saturated until the last rectangle in the row is black.

20.8. Color Menu

A menu that looks something like this is displayed when you select `Color` from the main Video Monitor Diagnostic menu. The squares will be filled with examples of the test patterns represented by R1 - R15 and L1 - L6.

Figure 20-3 *Color Video Pattern Menu*



R1

The *Screen Geometry Test* verifies that the screen is properly aligned, the video amplifier is working correctly, and the horizontal/vertical scan amplifiers are operating correctly.

Description:

The pattern has a "cross" centered in the middle of the display area. The cross consists of thin lines separated by a short space. This tests for precision and sweep linearity. One-fourth of the distance from the center in each of the four legs of the cross are four perpendicular lines. They provide four boxes for testing horizontal and vertical centering.

In the left and right eighth of the screen are a set of gray, black, and white bars to test for video amplifier ringing. In the upper left quadrant and one-eighth of the way from the center is a horizontal black line with two thin white lines entering a wide, vertical, half-gray line. This pattern tests for over/undershoot in the video amp. (On systems where there is no grayscale, the gray will be created by half-tone shading of the area, with every other pixel turned on and the adjacent pixel turned off.)

In the lower right quadrant of the screen is another horizontal black bar to maintain symmetry of the overall pattern. There are two white bars one-eighth of the way above below the center of the screen. The lower bar penetrates halfway into the vertical gray bars and the upper bar completely covers the gray bars.

R2

The *Focus and Brightness Test* verifies that the focus controls are properly set and that the monitor can be focused. The peak white level is set by this test.

Description:

The pattern consists of m's (the character `m`) and e's (the character `e`) filling the screen with a three inch white box in the middle. The m's and e's provide 3 short vertical lines connected at the top by a short horizontal line. The legs of the m's and e's are one pixel in width and are spaced two pixels apart.

This pattern is used to set and check the focus of the monitor. The focus adjustment is closely related to correct brightness so both adjustments are done here. Use the white square in the middle to set the screen's brightness. Its presence doesn't affect the focus of the screen. Use the m's and e's to set the screen's focus. The worst case focus occurs at the edges of the monitor, where the electron beam strikes the phosphor at an angle. If the dots are correct at the edge and one-fourth of the way to the center, they should be correct at the center.

There is occasionally a problem with focus linearity. The circuit that corrects focus, depending on the location of the dot on the screen, may be defective. If this is a problem, you may have to use the outline pattern to set the screen brightness and fill the white box with m's and e's.

R3

The *Black & White Convergence Test* verifies that the beam convergence is properly set.

Description:

This pattern consists of 25 large squares (white lines on black) with a white dot in the middle of each square. Use the horizontal and vertical lines to make sure the beams from the color gun(s) converge properly. Correctly set beams produce pure white lines and dots. A multicolored pattern indicates poor convergence.

R4

The *Luminance Uniformity Test* checks for uniform screen luminance (brightness) and correct high voltage regulation.

Description:

The monitor displays a white screen.

R5 - R15

The *Voltage and Geometry tests* check screen voltages and screen dimensions

Description:

The patterns R5 thru R15 are useful for checking video shadowing (ringing), high voltage regulation, and screen geometry.

L1

The *Red Luminance Test* checks the luminance uniformity of the red color gun.

Description:

Pattern L1 is a pure red screen.

L2

The *Green Luminance Test* checks the luminance uniformity of the green color gun.

Description:

Pattern L2 is a pure green screen.

L3

The *Blue Luminance Test* checks the luminance uniformity of the blue color gun.

Description:

Pattern L3 is a pure blue screen.

L4

The *Magenta Color Convergence Test* verifies alignment of the red and blue color guns.

Description:

Pattern L4 is a white background behind a magenta convergence pattern.

L5

The *Cyan Color Convergence Test* verifies alignment of the green and blue color guns.

Description:

Pattern L5 is a white background behind a cyan convergence pattern.

L6

The *Color Rainbow Test* verifies that the red, green, blue, and sync cables

are attached to the proper connectors.

Description

This pattern consists of a gray background behind two rows of eight rectangles each. The bottom row of rectangles, from left to right, are gray, red, orange, yellow, green, blue, magenta, and white. The top left rectangle is black, and is followed by half values of red, orange, yellow, green, blue, magenta, and white.

20.9. Error Messages

1 - <function> - Unable to map EEPROM

2 - <function> - Unable to unmap EEPROM

3 - <function> - Unable to allocate frame buffer

4 - <function> - unable to map device < >

5 - <function> - Unable to open frame buffer

21

Sun VME Interface Diagnostic

21

Sun VME Interface Diagnostic

21.1. General Description

VMEbus compatible board products can be combined to produce a wide range of system configurations ranging from the very simple to the very complex. The VMEbus specification defines specific functional entities or elements which can be implemented on a VMEbus compatible board. The VMEbus Interface implementation for the Sun product line provides Master, Slave, Interrupt Handler, and Arbiter VME bus functions. This diagnostic checks the VME interface on the CPU board.

21.2. Hardware Requirements

The required hardware is listed below.

NOTE *Installing any additional unspecified VMEbus boards can cause the diagnostic to produce unreliable and erroneous reports.*

- A 12-slot or smaller cardcage with sufficient power capacity to support two CPU boards.
- The Unit Under Test (UUT): a Sun CPU board.
- The Test Support CPU (TSCPU): a second, functionally sound, Sun CPU board.

1. Sun Memory board(s) if no memory is on CPU board(s).
2. Two Sun consoles (monitor and keyboard) or two "dumb" terminals (TeleVideo, Wyse, etc.)
3. A boot device (such as Ethernet).
4. Two ethernet transceiver cables (P/N 530-1241-01,"Assy., Cable transceiver 15 meter") and transceiver boxes (3COM 3C100 or equivalent). Alternatively, a single ethernet drop may be shared by the two CPU boards.
5. An InterProcessor Communication Serial Cable (IPC Serial Cable) connecting the UUT to the TSCPU by way of the "B" serial ports.

Configuration

The following is an example of a test system configuration that uses the hardware listed above:

1. The Unit Under Test and Test Support CPU are installed in the VMEbus cardcage, in slots that do *not* share a common P2 bus. Memory board(s) are

installed as required for the specific CPU board. The ARBITER and the VME Clock must be disabled on the board which takes the role of theTSCPU.(Consult CPU board configuration procedure for jumper locations.)

2. Either a Sun console (monitor and keyboard) or a dumb terminal connected to serial port A on the Unit Under Test is required for the user interface. A second console or a terminal attached to serial port A must be connected to the TSCPU for booting the SunDiagnostic Executive and viewing messages that may be displayed while testing is in progress.
3. An interprocessor serial communication cable is connected between serial port B on the UUT and serial port B on the TSCPU.
4. The two Ethernet transceiver boxes are connected to the network containing the File Server from which the diagnostic is downloaded. The transceiver cables connect each CPU (UUT and TSCPU) to a transceiver box.

21.3. Hardware Set-Up

UUT/TSCPU configuration

The hardware jumper placement and EEPROM settings required for correct operation of the VME diagnostic are described below.

Jumper Placement

Refer to the appropriate CPU board configuration document for jumper locations, which vary, depending on which CPU board is being tested.

Jumper Description	*UUT*	*TSCPU*
Requester only	off	on
Arbiter/Requester	on	off
Reset slave	off	on
Reset master	on	off
VMEclock enable	on	off

UUT EEPROM

EEPROM location 0x1f must be set to 0x10 in order to use a terminal on serial port A.

TSCPU EEPROM

EEPROM location 0x1f must be set to 0x10 in order to use a terminal on serial port A.

21.4. User Interface

The user interface consists of a Main Menu with five sub-menus. Each of the menus makes a number of visible and invisible options available to the user. Only the visible options are discussed in this chapter. *Chapter 2* provides interface support for menus and command line interpretation.

Many of the tests are composed of several sub-tests that check a particular aspect of an interface function. Any of the sub-tests may be executed individually for debugging purposes by choosing the appropriate parameter values. Default parameter values cause all subtests to run. Local copies of the global environmental parameters can be set and used by those tests that have a double asterisk (**) in their parameter lists. See the *Local Environment* section for details.

Each test description states:

1. The purpose of the test.
2. The parameters required.
3. The function of the test.

Note that a parameter range specified as "1__7" means the values 1 through 7 inclusive are allowed. If the base is "hex" and the description contains option values such as `1=walking zero, 2=walking 1,` or `4=constant,` values may be combined to select multiple options. For example, combining 1 and 4 (1+4=5) would cause "walking zero" and "constant" options to be selected. A range specified as "1,2,4" means only the values 1 or 2 or 4 exclusively are allowed. In that case, values MAY NOT BE COMBINED to select multiple options; e.g. "1+4=5" would not function correctly.

Command Line Description

The SunDiagnostic Executive command line interpreter is more powerful and provides more flexibility than previous command line interpreters. Multi-character commands are allowed. Parameters are optional. Default values are automatically provided for parameters not entered. Parameters may be entered in any order. Global environmental parameters can be set from the options menu. Refer to *Chapter 2* for more information on command line syntax.

Starting the Diagnostic

The VME diagnostic expects to see a pair of CPU's connected by a synchronization link. This design requires the special start-up procedure described below.

Start-up Procedure

The following procedure assumes that a single Ethernet drop is shared between the two CPU boards.

1.Check Set Up

- Make sure both CPU's are configured and installed correctly.
- Make sure the IPSC Cable is connected to serial port "B" of each CPU board.

2.Boot IPC Master

- Determine which CPU has its Arbiter enabled and connect the Ethernet cable to it. This board is the IPC Master and must be booted first.
- To Boot the SunDiagnostic Executive on the IPC Master, enter the following from the PROM monitor mode:

```
> b ie() stand/exec
```

- From the SunDiagnostic Executive's main menu type:

```
Command==> di;vme
```

to start the VME diagnostic. After the VME diagnostic has started, it will repeatedly display the message: `hit any key to display menu.` ***Don't press any keys at this time!***

3.Boot IPC Slave

- Disconnect the Ethernet cable from the IPC Master and connect it to the CPU that is the IPC Slave.
- Now boot the IPC Slave CPU by typing:

```
> b ie() stand/exec
```

- From the SunDiagnostic Executive's main menu type:

```
Command==> di;vme
```

to start the VME diagnostic. After the VME diagnostic has started, it will repeatedly display the message: `hit any key to display menu.` ***Don't press any keys at this time!***

- At this point, you must decide which CPU board is to be the UUT; press the spacebar on the keyboard attached to that CPU board. By default, the other CPU board will become the TSCPU. The CPU roles can be reversed later by changing the CPUMode setting from Options Menu, if desired.

  The UUT should display the Main Menu and the TSCPU should occasionally display the message `SLAVE: waiting for command.` ***Don't press any keys on the TSCPU keyboard!*** The TSCPU keyboard is not used unless the CPUMode is set to ASYNC and the Async Tests Menu is selected.

4.Run Tests

- The diagnostic is now ready use. Any of the default sequences can be used to run the synchronous Master and Slave tests. The CPU roles may be reversed by setting CPUMode=TSCPU on the Options Menu. Asynchronous mode may be selected from the Options Menu by setting CPUMode=ASYNC.

21.5. The Main Menu

The Main Menu, at the top level of this diagnostic, has a total of nine visible options. The Master, Slave, ASync, DEBug and Options menus are all available here. All the tests can be run at once by selecting `All`. The Error log displays a list of all the error messages generated by the diagnostic since it started. To go to a sub-menu, you must at least type in the letters shown in the menu in upper case. If the entry results in a command, that command will run; if it brings up a menu, that menu is displayed.

```
Sun VME Interface Diagnostic  REV x.x xx/xx/xx Main Menu

All         perform all tests in sequence
Default     perform default test sequence
Quick       perform quick test sequence
Master      Master tests menu
Slave       Slave tests menu
ASync       Asynchronous tests menu
DEBug       Debugging aids menu
Options     Options menu
Error       Error log display

message line

message line
 .
Command==>
```

All
Typing this command runs all tests in the sequence shown on the menu. The sequence is:

m ; a ; Esc ; s ; a ; Esc ;

Default
Typing this command runs a standard subset of tests that are comprehensive but complete in a more reasonable time. The sequence is:

m ; d ; Esc ; s ; d ; Esc ;

Quick
Typing this command runs the minimal set of tests that offer reasonable coverage. It will complete in a very short time, generally two minutes or less. The sequence is:

m ; q ; s Esc ; Esc ;

Master
Typing this command brings up the Master menu.

Slave
Typing this command brings up the Slave menu.

ASync
Typing this command brings up the ASync menu.

DEBug
Typing this command brings up the DEBug menu.

Options
Typing this command brings up the Options menu.

Error
Typing this command displays the error log.

21.6. The Master Tests Menu

The Master tests are designed to test the functionality of the VMEbus Master interface. The Arbiter test is included in this group to avoid creating an additional menu with only a single test entry. This menu has a total of nine visible options.

```
Sun VME Interface Diagnostic  REV x.x xx/xx/xx  Master Tests Menu

All         perform all tests in sequence
Default     perform default test sequence
Quick       perform quick test sequence
ADdress     address lines test
DAta        data lines tests
UNalign     unaligned transfer test
PRot        protection, timeout, & memory error tests
BUs         bus arbiter tests
Error       error log display

message line

message line

Command==>
```

All

Typing this command runs all tests in the sequence shown on the menu. The sequence is:

```
ad ; da ; un ; pr ; bu ;
```

Default

Typing this command runs a standard sub-set of tests that are comprehensive but complete in a more reasonable time. The sequence is:

```
ad ; da ; un ; pr ; bu ;
```

Quick

Typing this command runs the minimal set of tests that offer reasonable coverage. It will complete in a very short time, generally two minutes or less. The sequence is:

```
ad ; da ; un ; pr ; bu ;
```

ADdress

Address line tests verify the Master correct manipulation of DTB address lines over specified address range and identify failing address line combinations.

To run these tests, use this syntax, with or without the parameters shown. The minimum entry is shown in upper case; you need only enter **ad**, for example, to invoke the address line tests.

ADdress *DVma= OFFSET= OPSize= OPType= POrt= PType= **=*

The parameters are:

dvma

Enter **dv** to select the DVMA space to use. Enter **dv=1** for system DVMA. Enter **dv=2** for user DVMA.

offset

Offset specifies an address offset from the VME MAPPING TABLE address to use during a read or write operation.

opsize

opsize specifies the number of bytes to use during a read or write operation.

optype

specifies the type of operation to be tested. Enter **opt=1** for write, or **opt=2** for read.

port

specifies the VME port to use.

Enter **po=0** for the VME slave port.

Enter **po=1** for the VME 16-bit master port.

Enter **po=2** for the VME 32-bit master port.

** Refer to the *Local Environment* subsection for a list of global parameters that can be set for this test.

The following table shows default values and the acceptable range of values you may enter for each parameter.

parameter	*default*	*range*	*base (radix)*
dvma	3	1__3	hex
offset	0	0__1fff	hex
opsize	4	1,2,4	hex
optype	3	1__3	hex
port	3	1__3	hex

During each pass, the Address Line Tests toggle the address bits and execute write/read/compare operations, then report `bus` or `compare` errors. The parameters `dvma`, `opsize`, `optype`, `port`, and `offset` allow you to customize the test for debugging purposes. For example, by specifying `OPtype=2` means that a read only test is performed.

DAta

Data line tests verify Master's correct manipulation of DTB data lines over specified data range. They also identify failing data line combinations.

The command syntax is:

DAta *OPSize= OPType= PATtern= POrt= PType= **=*

The parameters are:

opsize

Opsize specifies the number of bytes to use during a read or write operation.

optype
specifies the type of operation to be tested. Enter **opt=1** for write, or **opt=2** for read.

pattern
is the constant value to be used if `ptype=4` (constant pattern type) is selected.

port
VME port to use.

Enter **po=1** for the VME 16-bit master port.

Enter **po=2** for the VME 32-bit master port.

ptype
is the pattern type to be applied to data lines. 1 = walking zero, 2 = walking one, 4 = constant.

** Refer to the *Local Environment* subsection for a list of global parameters that can be set for this test.

The following table shows default values and the acceptable range of values you may enter for each parameter.

parameter	*default*	*range*	*base (radix)*
opsize	4	1,2,4	hex
optype	3	1__3	hex
pattern	12345678	0__ffffffff	hex
port	3	1__3	hex
ptype	3	1__7	hex

During each pass, the Data Line Tests apply the pattern type specified by `ptype` to all of the data bits, execute write/read/compare operations, then report *bus* or *compare* errors. The parameters `dvma`, `opsize`, `optype`, `pattern`, `port`, and `offset` allow you to customize the test for debugging purposes. For example, by specifying `OPtype=2` a read only test is performed.

UNalign

Unaligned transfer tests verify the Master's ability to handle non-aligned transfer without error.

You need only enter the letters shown in upper case, or you may enter the entire command or parameter word. The command syntax is:

UNalign*AIndex= OFFSet= OPType= POrt= SEnse= **=*

The following paragraphs describe the parameters:

aindex

The `address index` parameter specifies the map entry to be used in the VME MAPPING TABLE.

offset

specifies an address offset from the VME MAPPING TABLE address to use during a read or write operation.

optype

specifies the `operation type` to be tested. 1 = write, 2 = read.

port

specifies which VME port to use.

Enter **po=0** for the VME slave port.

Enter **po=1** for the VME 16-bit master port.

Enter **po=2** for the VME 32-bit master port.

sense

specifies whether bit ON(1) or bit OFF(0) addresses are used.

** Refer to the *Local Environment* subsection for a list of global parameters that can be set for this test.

The table that follows shows default values and the acceptable range of values you may enter for each Unalign Transfer Test parameter.

parameter	*default*	*range*	*base (radix)*
aindex	0	0__31	dec
offset	0	0__1fff	hex
optype	3	1__3	hex
port	3	1__3	hex
sense	0	0__1	hex

During each pass, the Unalign Transfer Tests force VME Master writes/reads to/from locations not modulo 4 aligned, then compare values written to the Master with values actually read from Slave and Master to verify correct operation. Any data corruption is reported as an error. The parameters `aindex`, `offset`, `optype`, `port`, and `sense` allow the you to customize the test for debugging purposes. For example, by specifying `OPtype=2` means a read only test is performed.

PRot

Protection, timeout and memory error tests verify correct operation of error reporting circuitry specific to the Master's VMEbus interface.

The command syntax is:

PRot *SUbtest*= **=

subtest

specifies the sub-test(s) to execute during each test pass. 1 = read_only, 2 = supervisor_only, 4 = invalid_page, 8 = timeout, 0x10 = slave_error.

The table that follows shows default values and the acceptable range of values you may enter for the sub-test parameter.

parameter	*default*	*range*	*base (radix)*
subtest	1f	1__1f	hex

Each sub-test tests a specific error or exception reporting function by forcing conditions under which the exceptions should occur and then checking for an appropriate exception report. The read_only, supervisor_only, and invalid_page subtests test MMU generated exceptions, while the timeout and slave_error sub-tests test externally generated exception conditions.

** Refer to the *Local Environment* subsection for a list of global parameters that can be set for this test.

BUs

Bus arbiter tests verify correct bus arbiter operation on the Unit Under Test.

The command syntax is:

Bus*ITerations*= *LEn*= **=

iterations

Number of times the data block is written/read per pass.

len Length of the data block to use in bytes.

** Refer to the *Local Environment* subsection for a list of global parameters that can be set for this test.

The table that follows shows default values and the acceptable range of values you may enter for the parameters.

parameter	*default*	*range*	*base (radix)*
iterations	1000	1__f0000	hex
len	100	1__400	hex

During each pass, the Bus Arbiter Tests set up Master and Slave on both the UUT and TSCPU so that each Master can write/read the Slave on the other CPU. The tests let each Master write *len* data bytes, read them back, and test for corrupted data, memory errors, and timeout exceptions. The *iterations* parameter specifies the number of iterations.

21.7. The Slave Tests Menu

The Slave tests test the functionality of the VMEbus Slave interface.

The Slave Tests Menu has a total of eight visible options.

```
Sun VME Interface Diagnostic  REV x.x xx/xx/xx  Slave Tests Menu

All     perform all tests in sequence
Default perform default test sequence
Quick   perform quick test sequence
ADdress address lines tests
DAta    data lines tests
PRot    protection & memory error tests
ENable  dvma enable register tests
Error   error log display

message line

message line

Command==>
```

All

Typing this command runs all tests in the order shown in the menu. The sequence is:

```
ad ; da ; pr ; en ;
```

Default

Typing this command runs a standard subset of tests that are comprehensive but complete in a more reasonable time. The sequence is:

```
ad ; da ; pr ; en ;
```

Quick

Typing this command runs the minimal set of tests that offer reasonable coverage. It will complete in a very short time, generally two minutes or less.

The sequence is:

```
ad ; da ; pr ; en ;
```

ADdress

Address lines tests verify the Slave's correct response to manipulation of DTB address lines over a specified address range and identify failing address line combinations.

When entering the test command and specifying parameters, you need only enter the letters shown here in upper case. The command syntax is:

ADdress *DVma= OFFSET= OPSize= OPType= POrt= PType= **=*

The parameters are:

dvma
specifies which toDVMAspace

offset
specifies an address offset from the VME MAPPING TABLE address to use during a read or write operation.

opsize
specifies the number of bytes to use during a read or write operation.

optype
specifies the Operation Type to be tested. 1 = write, 2 = read.

port
specifies which VME port to use.

Enter **po=0** for the VME slave port.

Enter **po=1** for the VME 16-bit master port.

Enter **po=2** for the VME 32-bit master port.

ptype
specifies the Pattern Type to be applied to data lines. 1 = walking zero, 2 = walking one, 4 = constant.

** Refer to the *Local Environment* subsection for a list of global parameters that can be set for this test.

The table that follows shows default values and the acceptable range of values you may enter for the parameters.

parameter	*default*	*range*	*base (radix)*
dvma	3	1__3	hex
offset	0	0__1fff	hex
opsize	4	1,2,4	hex
optype	3	1__3	hex
port	3	1__3	hex
ptype	7	1__7	hex

During each pass, the Address Lines Tests toggle the address bits and execute write/read/compare operations. They report `bus` or `compare` errors. The parameters `dvma`, `offset`, `opsize`, `optype`, `port`, and `ptype` allow you to customize the test for debugging purposes. For example, by specifying `OPtype=2` ,a read only test is performed.

DAta

Data line tests verify the Slave's correct response to manipulation of DTB data lines over specified data range and identify failing data line combinations.

Command syntax is:

DAta *OPSize= OPType= PATtern= POrt= PType= **=*

The parameters are:

opsize

specifies the number of bytes to use during a read or write operation.

optype

specifies the Operation Type to be tested. 1 = write, 2 = read.

pattern

specifies a constant to be written to local memory location if constant pattern type is selected (`ptype`).

port

specifies which VME port to use.

Enter **po=0** for the VME slave port.

Enter **po=1** for the VME 16-bit master port.

Enter **po=2** for the VME 32-bit master port.

ptype=

specifies the pattern type to be applied to data lines. 1 = walking zero, 2 = walking one, 4 = constant.

** Refer to the *Local Environment* subsection for a list of global parameters that can be set for this test.

The table that follows shows default values and the acceptable range of values you may enter for the parameters.

parameter	*default*	*range*	*base (radix)*
opsize	4	1,2,4	hex
optype	3	1__3	hex
pattern	12345678	0__ffffffff	hex
port	3	1__3	hex
ptype	3	1__7	hex

During each pass, the Data Line Tests apply the pattern type specified by `ptype` to all of the data bits, then execute write/read/compare operations. They report `bus` or `compare` errors. The parameters `opsize`, `optype`,

`pattern`, `port`, and `ptype` allow you to customize the test for debugging purposes. For example, by specifying `OPtype=2`, a read only test is performed.

PRot

Protection & memory error tests verify correct operation of Slave VMEbus interface error reporting circuitry.

The command syntax is:

PRot *SUbtest*= **=

subtest

specifies which Sub-test(s) to perform:

1 = nontype0
2 = sdvma_invalid
4 = sdvma_read_only
8 = udvma_invalid
0x10 = udvma_read_only
0x20 = udvma_supervisor_only.

** Refer to the *Local Environment* subsection for a list of global parameters that can be set for this test.

The table that follows shows default values and the acceptable range of values you may enter for the *subtest* parameter.

parameter	*default*	*range*	*base (radix)*
subtest	ff	0__ff	hex

During each pass of the DVMA Enable Register Tests, the subtest(s) specified by executed.`subtest`are Each sub-test checks a specific error/exception reporting function by forcing conditions under which the exceptions should occur, then checking for an appropriate exception report. The sub-tests check MMU and memory generated exceptions as well as the Slave VMEbus error reporting circuitry.

ENable

DVMA Enable register tests verify correct operation of the VMEbus DVMA enable circuitry for System and User DVMA.

Command syntax is: **ENable** *SUbtest= COntext= **=*

subtest

specifies which sub-test(s) to perform. 1 = System DVMA, 2 = User DVMA.

context

specifies which context(s) to test when User DVMA is specified. The parameter requires an 8-bit entry with same format as the User DVMA enable register.

** Refer to the *Local Environment* subsection for a list of global parameters that can be set for this test.

The table that follows shows default values and the acceptable range of values you may enter for the parameters.

parameter	*default*	*range*	*base (radix)*
subtest	3	1__3	hex
context	ff	00__ff	hex

During each pass, the Enable Register Tests execute the subtest(s) specified by `subtest`. The System DVMA subtest checks proper function of the System DVMA enable bit. The User DVMA subtest checks proper function of each enable bit in the User DVMA enable register specified by the `context` parameter.

21.8. The Asynchronous Tests Menu

The Asynchronous tests are designed to provide you with a set of asynchronous functions similar to those found in the *Carpvme* PROMs. Interprocessor communication is not used to synchronize operation of the two CPU boards; therefore the distinction between UUT and TSCPU becomes meaningless. Instead, the terms LOCAL and REMOTE are used to indicate on which CPU board memory the operations are taking place.

LOCAL memory is the memory on the CPU board connected to the terminal or console through which you are currently entering commands. REMOTE memory is the memory on the "other" CPU board, accessed over the VMEbus. Since operation of the two CPU boards is asynchronous, any combination of functions may be run when two terminals or consoles are used.

The Asynchronous Tests Menu has a total of 11 options.

```
Sun VME Interface Diagnostic REV x.x xx/xx/xx  Async Tests Menu

All          perform all tests in sequence
Default      perform default test sequence
Quick        perform quick test sequence
AX           write local memory
BX           read & display local memory
CX           write remote memory (over VME)
DX           read remote memory (over VME)
GX           w/r remote memory space (over VME)
KX           w/r remote memory offset data (over VME)
Error        error log display

message line

message line

Command==>
```

`All`

Typing this command runs all tests in the sequence they appear in the menu. The sequence is:

```
gx; kx;
```

Default

Typing this command runs a standard subset of tests that are comprehensive but complete in a more reasonable time. The sequence is:

```
gx; kx;
```

Quick

Typing this command runs the minimal set of tests that offer reasonable coverage. It will complete in a very short time, generally two minutes or less. The sequence is:

```
gx; kx;
```

AX

Write Local Memory writes a pattern into a local memory location. The other CPU board can access this pattern over the VMEbus, using remote read or write operations.

You need only enter the letters shown in upper case, followed by the "equals" sign and the parameter entry. The command syntax is:

AX *AIndex= OFFSet= OPSize= PATtern= PType= SEnse=* ***=**

The parameters are:

aindex

Address Index specifies the map entry to be used in the VME MAPPING TABLE.

offset

specifies an address offset from the VME MAPPING TABLE address, for use during a read or write operation.

opsize

specifies the number of bytes to use during a read or write operation.

pattern

Pattern constant value to be used if constant pattern type is selected.

ptype

specifies the pattern type to be applied to the data lines. 1 = walking zero, 2 = walking one, 4 = constant.

sense

specifies whether bit ON(1) or bit OFF(0) addresses are used.

** Refer to the *Local Environment* subsection for a list of global parameters that can be set for this test.

The table that follows shows default values and the acceptable range of values you may enter for the parameters.

parameter	*default*	*range*	*base (radix)*
aindex	0	0__31	dec
offset	0	0__1fff	hex
opsize	4	1,2,4	hex
pattern	12345678	0__ffffffff	hex
ptype	3	1__7	hex
sense	0	0__1	hex

The *Write Local Memory* command writes the specified pattern type to a local memory location for the number of iterations specified by `pass`. If `pattern type = constant` (`ptype=4`), the test uses the constant specified in `pattern`.

BX

Read & Display Local Memory reads a local memory location and displays the value observed. This value may be accessed by the other CPU board over the VMEbus, using remote read or write operations.

The command syntax is:

BX *AIndex= SEnse= OPSize= OFFSet= **=*

Command parameters are:

aindex

Address index specifies the map entry to be used in the VME MAPPING TABLE.

sense

specifies whether bit ON(1) or bit OFF(0) addresses are used.

opsize

specifies the number of bytes to use during a read or write operation.

offset

specifies an address offset from the VME MAPPING TABLE address for use during a read or write operation.

** Refer to the *Local Environment* subsection for a list of global parameters that can be set for this test.

The table that follows shows default values and the acceptable range of values you may enter for the parameters.

parameter	*default*	*range*	*base (radix)*
aindex	0	0__31	dec
sense	0	0__1	hex
opsize	4	1,2,4	hex
offset	0	0__1fff	hex

The `Read and Display Local Memory` test reads the local memory location and displays the observed value for the number of iterations specified by `pass`. The local memory location is obtained by adding `offset` to the address specified in the VME MAPPING TABLE, indexed by

aindex,sense,VME_SLAVE (see end of chapter).

CX

Write Remote Memory (over VMEbus) writes a pattern into a remote memory location over the VMEbus.

Command syntax is:

CX *AIndex= OFFSet= OPSize= PATtern= POrt= PType= SEnse= **=*

Command parameters are:

aindex

Address index specifies the map entry to be used in the VME MAPPING TABLE.

offset

specifies an address offset from the VME MAPPING TABLE address to use during a read or write operation.

opsize

specifies the number of bytes to use during a read or write operation.

pattern

specifies the *Pattern constant* to be written to a remote memory location if `constant pattern type` (`ptype=4`) is selected.

port

specifies which VME port to use:

Enter **po=0** for the VME slave port.

Enter **po=1** for the VME 16-bit master port.

Enter **po=2** for the VME 32-bit master port.

ptype

specifies the *pattern type* to be applied to the data lines. 1 = walking zero, 2 = walking one, 4 = constant.

sense

specifies whether bit ON(1) or bit OFF(0) addresses are used.

** Refer to the *Local Environment* subsection for a list of global parameters that can be set for this test.

The table that follows shows default values and the acceptable range of values you may enter for the parameters.

parameter	*default*	*range*	*base (radix)*
aindex	0	0__31	dec
offset	0	0__1fff	hex
opsize	4	1,2,4	hex
pattern	12345678	0__ffffffff	hex
port	1	1__2	hex
ptype	3	1__7	hex
sense	0	0__1	hex

The `Write Remote Memory` test writes the specified pattern type into the remote memory location through the VMEbus for the number of iterations specified by `pass`. If pattern type = constant (`ptype=4`), the test uses the constant specified in `pattern`. For each iteration the test uses 16- or 32-bit master data space as specified by *port*. If a bus error occurs and the loop-on-error flag is set, a scope loop is entered.

DX

Read Remote Memory (over VMEbus) reads from a remote memory location over the VMEbus and displays the observed value.

Command syntax is: **DX** *AIndex= OFFSet= OPSize= POrt= SEnse= **=*

The parameters are:

aindex

address index specifies the map entry to be used in the VME MAPPING TABLE.

offset

specifies an address offset from the VME MAPPING TABLE address for use during a read or write operation.

opsize

specifies the number of bytes to use during a read or write operation.

port

specifies the VME port to use:

Enter **po=0** for the VME slave port.

Enter **po=1** for the VME 16-bit master port.

Enter **po=2** for the VME 32-bit master port.

sense

specifies whether bit ON(1) or bit OFF(0) addresses are used.

** Refer to the *Local Environment* subsection for a list of global parameters that can be set for this test.

The table that follows shows default values and the acceptable range of values you may enter for the parameters.

parameter	*default*	*range*	*base (radix)*
aindex	0	0__31	dec
offset	0	0__1fff	hex
opsize	4	1,2,4	hex
port	1	1__2	hex
sense	0	0__1	hex

The `Read Remote Memory` test reads from the remote memory location over the VMEbus and displays the observed value. The number of iterations is specified in the pass count limit. For each iteration, the test uses 16 or 32 bit master data space as specified by the parameter `port`. If a bus error occurs and loop on error flag is set, a scope loop is entered.

GX

Write/Read Remote Memory Space (over VMEbus) verifies correct operation while exercising all address bits.

Command syntax is: **GX** *DVma= OFFSet= OPSize= POrt= PType= **=*

The parameters are:

dvma

specifies which DVMA space to use: System or User. 1 = System, 2 = User.

offset

specifies an address offset from the VME MAPPING TABLE address to use during a read or write operation.

opsize

specifies the number of bytes to use during a read or write operation.

port

VME port to use:

Enter **po=0** for the VME slave port.

Enter **po=1** for the VME 16-bit master port.

Enter **po=2** for the 32-bit master port.

ptype

specifies the pattern type to be applied to the data lines. 1 = walking zero, 2 = walking one, 4 = constant.

** Refer to the *Local Environment* subsection for a list of global parameters that can be set for this test.

The table that follows shows default values and the acceptable range of values you may enter for the parameters.

parameter	*default*	*range*	*base (radix)*
dvma	3	1__3	hex
offset	0	0__1fff	hex
opsize	4	1,2,4	hex
port	3	1__3	hex
ptype	7	1__7	hex

The `Write/Read Remote Memory Space` Test executes the test for the number of iterations specified in the `pass` count limit parameter. For each iteration, the test applies the pattern `data=` *address* while toggling all VMEbus address bits at least once, using the Slave DVMA space specified by DVMA. The test compares actual values read from the Slave to unique values written to the Slave and reports differences as failures. Selection of 16 or 32 bit data is determined by `port`. If a bus error or data compare error occurs, and the loop-on-error flag is set, a scope loop is entered.

Appropriate address truncation and skipping is be performed automatically to conform to hardware and software constraints. These constraints include address sizes, Slave VMEbus addressing, and preservation of bootPROM monitor variables.

KX

Write/Read Remote Memory Offset Data (over VMEbus) verifies the Master's ability to handle non-aligned transfer without error.

Command syntax is:

KX *AIndex= OFFSet= POrt= SEnse= **=*

aindex

address index specifies the map entry to be used in the VME MAPPING TABLE.

offset

specifies an address offset from the VME MAPPING TABLE address for use during a read or write operation.

port

VME port to use:

Enter **po=0** for the VME slave port.

Enter **po=1** for the VME 16-bit master port.

Enter **po=2** for the 32-bit master port.

sense

specifies whether bit ON(1) or bit OFF(0) addresses are used.

** Refer to the *Local Environment* subsection for a list of global parameters that can be set for this test.

The table that follows shows default values and the acceptable range of values you may enter for the parameters.

parameter	*default*	*range*	*base (radix)*
aindex	0	0__31	dec
offset	0	0__1fff	hex
port	1	1__2	hex
sense	0	0__1	hex

The `Write/Read Remote Memory Offset Data` test executes the test for the number of iterations specified in the `pass` count limit. For each iteration it forces remote writes/reads to/from locations not modulo 4 aligned. It compares written values with values actually read back to verify correct operation. It reports any data corruption as an error. If a bus or data compare error occurs and the loop on error flag is set, a scope loop is entered.

21.9. The Debugging Aids Menu

The Debugging Aids are functions that allow tight loop repetition of VMEbus atomic functions (writes or reads) with user specified address or data values. This aids oscilloscope debugging in situations where a closed automatic test is insufficient or not available.

The Debugging Aids Menu has a total of five visible options.

```
Sun VME Interface Diagnostic  REV x.x xx/xx/xx  Debugging Aids Menu

ADdress address lines exerciser
DAta    data line exerciser
Error   error log display

message line

message line

Command==>
```

ADdress

The *Address line exerciser* exercises address lines for debugging with an oscilloscope or logic analyzer.

The command syntax is:

ADdress *AIndex= OFFSet= OPSize= OPType= POrt=* ***=**

Parameters are:

aindex

address index specifies the map entry to be used in the VME MAPPING TABLE.

offset

specifies an address offset from the VME MAPPING TABLE address for use during a read or write operation.

opsize

specifies the number of bytes to use during a read or write operation.

optype

specifies the Operation Type to be tested. 1 = write, 2 = read.

port

VME port to use:

Enter **po=0** for the VME slave port.

Enter **po=1** for the VME 16-bit master port.

Enter **po=2** for the VME 32-bit master port.

** Refer to the *Local Environment* subsection for a list of global parameters that can be set for this test.

The table that follows shows default values and the acceptable range of values you may enter for the parameters.

parameter	*default*	*range*	*base (radix)*
aindex	0	0__31	dec
offset	0	0__1fff	hex
opsize	4	1,2,4	hex
optype	1	1,2	hex
port	0	0,1,2	hex

During each pass, the `Address Line Exerciser` toggles the address bit specified by `aindex`. The parameters `offset`, `opsize`, `optype`, and `port` allow you to customize the test for debugging purposes. For example, by specifying `OPtype=2`, a read only test is performed.

DAta

The *Data line exerciser* exercises data lines for debugging with an oscilloscope or logic analyzer.

Command syntax is:

DAta *AIndex= OFFSet= OPSize= OPType= PATtern= POrt= PType= SEnse= **=*

Parameters are:

aindex

address index specifies the map entry to be used in the VME MAPPING TABLE.

offset

specifies an address offset from the VME MAPPING TABLE address for use during a read or write operation.

opsize

specifies the number of bytes to use during a read or write operation.

optype

specifies the Operation Type to be tested. 1 = write, 2 = read.

pattern

specifies the *Pattern constant* to be written to a remote memory location if `constant pattern type` (`ptype=4`) is selected.

port

VME port to use:

Enter **po=0** for the VME slave port.

Enter **po=1** for the VME 16-bit master port.

Enter **po=2** for the VME 32-bit master port.

ptype

specifies the pattern type to be applied to the data lines. 1 = walking zero, 2 = walking one, 4 = constant.

sense

specifies whether bit ON(1) or bit OFF(0) addresses are used.

** Refer to the *Local Environment* subsection for a list of global parameters that can be set for this test.

ptype

The table that follows shows default values and the acceptable range of values you may enter for the parameters.

parameter	*default*	*range*	*base (radix)*
aindex	0	0__31	dec
offset	0	0__1fff	hex
opsize	4	1,2,4	hex
optype	3	1__3	hex
pattern	12345678	0__ffffffff	hex
port	0	0,1,2	hex
ptype	7	1__7	hex
sense	0	0__1	hex

During each pass, the Data Line Exerciser applies the pattern type specified by `ptype`. The parameters `aindex`, `offset`, `opsize`, `optype`, `pattern`, `port`, `ptype`, and `sense` allow you to customize the test for debugging purposes. For example, by specifying `OPtype=2`, a read-only test is performed.

21.10. The Options Menu

The options are non-test, non-exerciser functions that control and display program environment variables, flags, and the error log.

The Options Menu has a total of nine visible options.

```
Sun VME Interface Diagnostic REV x.x xx/xx/xx Options Menu

CPUMode=    Set cpu mode - UUT,TSCPU,ASYNC currently:
MV=         Message verbosity - Off,Terse,Verbose currently:
Pass=       Pass count currently:
SCope=      Scopeloop on error currently:
SOft=       Soft error retry count currently:
STop=       Stop on Nth error currently:
WAit=       Wait on error for message viewing currently:
Def         Set default values
Error       error log display

message line

message line

Command==>
```

CPUMode=

Set CPU mode sets the program CPU mode variable.

To set this mode, enter

CPUMode=

followed by one of the parameters listed below under *range.*

variable	*default*	*range*	*base (radix)*
cpumode	ASYNC	ASYNC,UUT,TSCPU	string

MV=

sets message verbosity.

To set this mode, enter

MV=

followed by **0** for no messages, **1** for short messages, or **2** for full verbosity.

variable	*default*	*range*	*base (radix)*
mv	1	0:2	hex

Pass=

sets the pass count limit.

To set this mode, enter

Pass=

followed by a hexadecimal value within the range shown below:

variable	default	range	base (radix)
pass	1	1:0xfffffffe	hex

The `pass` parameter sets a global pass count limit to value entered. This limit allows a test to run until the number of iterations exceeds the pass count limit, unless an "on error limit" is exceeded first.

SCope=

sets the scopeloop on error flag.

To set this flag, enter

SCope=

followed by **0** to turn scope looping OFF, or **1** to turn it ON.

variable	*default*	*range*	*base (radix)*
scope	0	0:1	hex

This parameter sets a global scopeloop-on-error flag. The loop on error flag may be ignored for tests or functions for which the loop on error action has no meaning. The loop-on-error function displays a message indicating the type of operation(s), the address, and the data used while looping. The DIAGNOSTIC bit in the enable register is set to "1" before the operation(s) and set to "0" after the operation(s) while looping. Use this as a SYNC trigger for scope/analyzer debugging. No messages are displayed while looping in order to keep the loop as tight as possible.

SOft

sets the soft error retry limit.

To set the number of times a test will retry after an error occurs, enter

SOft=

followed with a value within the range shown below.

variable	*default*	*range*	*base (radix)*
soft	0	0:0xfffffffe	hex

`Soft` sets the soft error retry count limit to the value entered. Applicable tests retry if an error occurs, until the number of retrys exceeds the soft error retry count limit.

STop

sets stop on nth error limit.

To set this parameter, enter

STop=

followed with the hexadecimal value that represents the number of errors the test is to allow before halting.

variable	*default*	*range*	*base (radix)*
stop	1	1:0xfffffffe	hex

`Stop` sets stop on nth error limit to the value entered. This limit causes testing to be stopped when the number of errors exceeds the value entered. A "continue on error" parameter can be simulated by setting `STop` to some large value.

WAit=

sets the wait on error flag.

To set this parameter, enter

WAit=

followed by **0** to turn the flag OFF, or **1** to turn the flag ON.

variable	*default*	*range*	*base (radix)*
wait	0	0:1	hex

The `wait` parameter sets a global wait on error flag. If an error occurs and the wait on error flag is "1", the program pauses for about 15 seconds. This permits messages on the screen to be read before the menu is repainted.

Def

sets default values for all global variables and flags, except CPUmode.. To set this parameter, enter

Def

Error

Error Log Display displays the list of errors recorded in the errlog. Optional user entered parameters are provided for controlling the display verbosity and flushing the errlog.

The command syntax is:

Error *VErb= FLush=*

verb

controls display verbosity. 1 = terse, 2 to f = verbose.

flush

controls whether or not the errlog is flushed. 0 = noflush, 1 = flush.

parameter	*default*	*range*	*base (radix)*
verb	1	1:0xf	hex
flush	0	0,1	hex

Local Environment

The local environmental parameters are a subset of the global environmental parameters available on the *Options Menu.* Any of these parameters may be entered on a command line following a test command, along with any test parameters required for that test.

The local environmental parameter value exists only for the duration of the test command which it follows. If a local environmental parameter is not entered, the current value of the equivalent global environmental parameter (displayed in the *Options Menu*) becomes the default value.

The parameters that follow are applicable to all the tests that show double asterisks (**) in the command syntax example.

Local environmental parameters

provide a facility for locally modifying the value of the global environmental parameters on a test by test basis, without changing the current global values or forcing the user to continually go to The Options Menu.

The command syntax is:

TESTCMD TESTPARAMETER= MV= Pass= SCope= SOft= STop= WAit=

After entering the test command and any of the test parameters discussed on the previous pages, you may enter the letters shown in upper case in the syntax example above, followed with the values shown below.

mv sets message verbosity. 0 = off, 1 = terse, 2 = verbose.

pass

sets the pass count limit, which is the number of times a test is to be executed.

scope

sets the scope flag. 0 = off, 1 = on.

soft

sets soft error retry limit.

stop

sets the stop on nth error limit.

wait

sets the wait flag. 0 = off, 1 = on.

The table below shows the default values for the local environmental parameters, the acceptable ranges, and whether the entry should be in hexadecimal or decimal.

parameter	*default*	*range*	*base (radix)*
mv	1	0:2	hex
pass	1	1:*	decimal
scope	0	0:1	hex
soft	0	0:0xfffffffe	hex
stop	1	1:0xfffffffe	hex
wait	0	0:1	hex

The local environmental parameters command line entry executes the specified test, using the test and local parameter values entered in place of *TESTCMD* and *TESTPARAMETER* in the syntax example. For parameters not entered on the command line, the program uses the local default values for the test parameters and current global environmental parameter values as defaults for the local environmental parameters.

21.11. Glossary

ARBITER
A functional module that accepts bus requests from a master or interrupt handler and grants control to only one such requester at a time.

CPU
Central Processing Unit in this chapter is a reference to a Sun-3 CPU board, containing the MC68020 microprocessor chip.

DTB
Data Transfer Bus - One of four buses provided on the VMEbus backplane. Used to transfer binary data between a Master and a Slave on the VMEbus.

ETHERNET
Communication link between systems, using coaxial cable.

FRU
Field Replaceable Unit

INTERRUPTER
A functional module that generates interrupt requests on the PIB.

INTERRUPT HANDLER
A functional module that detects and responds to requests generated by an Interrupter.

IPC
InterProcessor Communication, used in references to the communication link that exists between the UUT and the TSCPU for process synchronization purposes.

IPC Master
InterProcessor Communication Master is the CPU that initiates commands on the IPC link and whose console is used for controlling synchronous testing. Not to be confused with VMEbus MASTER below, or the SunIPC board.

IPC Slave
InterProcessor Communication Slave is the CPU that responds to commands on the IPC link and whose console is display-only during synchronous testing. Not to be confused with VMEbus SLAVE below, or the SunIPC board.

MASTER
A functional module that initiates DTB cycles in order to transfer data between itself and a Slave module.

PIB
Priority Interrupt Bus - One of four buses provided on the VMEbus backplane. Used by an interrupter to send interrupt requests to an interrupt handler.

SLAVE
A functional module that detects DTB cycles initiated by a Master and, when those cycles specify its participation, transfers data between itself and the Master.

TSCPU
The Test Support CPU is a second, "known good" Sun-3 CPUboard installed in the VMEbus cardcage and used to support testing of the UUT.

UUT
Unit Under Test is the Sun-3 CPU board whose VMEbus interfaces are being tested.

VMEbus
An interfacing system used to interconnect data processing, data storage, and peripheral control devices.

21.12. VME Map Table

Table 21-1 *VME Map Table*

(i,p,s)	bit no. value	sense	port/ context	VME bus address	virtual address	physical address
(00,0,0)	00 0x00000001	OFF	Slave/S*	0x00000030	0x0ff00030	**********
(00,0,1)	00 0x00000001	ON	Slave/S*	0x00000031	0x0ff00031	**********
(00,1,0)	00 0x00000001	OFF	D16	0x00000030	**********	0x00000030
(00,1,1)	00 0x00000001	ON	D16	0x00000031	**********	0x00000031
(00,2,0)	00 0x00000001	OFF	D32	0x00000030	**********	0x00000030
(00,2,1)	00 0x00000001	ON	D32	0x00000031	**********	0x00000031
(01,0,0)	01 0x00000002	OFF	Slave/S*	0x00000050	0x0ff00050	**********
(01,0,1)	01 0x00000002	ON	Slave/S*	0x00000052	0x0ff00052	**********
(01,1,0)	01 0x00000002	OFF	D16	0x00000050	**********	0x00000050
(01,1,1)	01 0x00000002	ON	D16	0x00000052	**********	0x00000052
(01,2,0)	01 0x00000002	OFF	D32	0x00000050	**********	0x00000050
(01,2,1)	01 0x00000002	ON	D32	0x00000052	**********	0x00000052
(02,0,0)	02 0x00000004	OFF	Slave/S*	00000000	0x0ff00000	**********
(02,0,1)	02 0x00000004	ON	Slave/S*	0x00000004	0x0ff00004	**********
(02,1,0)	02 0x00000004	OFF	D16	00000000	**********	00000000
(02,1,1)	02 0x00000004	ON	D16	0x00000004	**********	0x00000004
(02,2,0)	02 0x00000004	OFF	D32	00000000	**********	00000000
(02,2,1)	02 0x00000004	ON	D32	0x00000004	**********	0x00000004
(03,0,0)	03 0x00000008	OFF	Slave/S*	00000000	0x0ff00000	**********
(03,0,1)	03 0x00000008	ON	Slave/S*	0x00000008	0x0ff00008	**********
(03,1,0)	03 0x00000008	OFF	D16	00000000	**********	00000000
(03,1,1)	03 0x00000008	ON	D16	0x00000008	**********	0x00000008
(03,2,0)	03 0x00000008	OFF	D32	00000000	**********	00000000
(03,2,1)	03 0x00000008	ON	D32	0x00000008	**********	0x00000008
(04,0,0)	04 0x00000010	OFF	Slave/S*	00000000	0x0ff00000	**********
(04,0,1)	04 0x00000010	ON	Slave/S*	0x00000010	0x0ff00010	**********
(04,1,0)	04 0x00000010	OFF	D16	00000000	**********	00000000
(04,1,1)	04 0x00000010	ON	D16	0x00000010	**********	0x00000010
(04,2,0)	04 0x00000010	OFF	D32	00000000	**********	00000000
(04,2,1)	04 0x00000010	ON	D32	0x00000010	**********	0x00000010
(05,0,0)	05 0x00000020	OFF	Slave/S*	00000000	0x0ff00000	**********
(05,0,1)	05 0x00000020	ON	Slave/S*	0x00000020	0x0ff00020	**********
(05,1,0)	05 0x00000020	OFF	D16	00000000	**********	00000000
(05,1,1)	05 0x00000020	ON	D16	0x00000020	**********	0x00000020
(05,2,0)	05 0x00000020	OFF	D32	00000000	**********	00000000
(05,2,1)	05 0x00000020	ON	D32	0x00000020	**********	0x00000020
(06,0,0)	06 0x00000040	OFF	Slave/S*	00000000	0x0ff00000	**********
(06,0,1)	06 0x00000040	ON	Slave/S*	0x00000040	0x0ff00040	**********
(06,1,0)	06 0x00000040	OFF	D16	00000000	**********	00000000
(06,1,1)	06 0x00000040	ON	D16	0x00000040	**********	0x00000040
(06,2,0)	06 0x00000040	OFF	D32	00000000	**********	00000000
(06,2,1)	06 0x00000040	ON	D32	0x00000040	**********	0x00000040

Table 21-1 *VME Map Table— Continued*

(i,p,s)	bit no.	value	sense	port/ context	VME bus address	virtual address	physical address
(07,0,0)	07	0x00000080	OFF	Slave/S*	00000000	0x0ff00000	**********
(07,0,1)	07	0x00000080	ON	Slave/S*	0x00000080	0x0ff00080	**********
(07,1,0)	07	0x00000080	OFF	D16	00000000	**********	00000000
(07,1,1)	07	0x00000080	ON	D16	0x00000080	**********	0x00000080
(07,2,0)	07	0x00000080	OFF	D32	00000000	**********	00000000
(07,2,1)	07	0x00000080	ON	D32	0x00000080	**********	0x00000080
(08,0,0)	08	0x00000100	OFF	Slave/S*	00000000	0x0ff00000	**********
(08,0,1)	08	0x00000100	ON	Slave/S*	0x00000100	0x0ff00100	**********
(08,1,0)	08	0x00000100	OFF	D16	00000000	**********	00000000
(08,1,1)	08	0x00000100	ON	D16	0x00000100	**********	0x00000100
(08,2,0)	08	0x00000100	OFF	D32	00000000	**********	00000000
(08,2,1)	08	0x00000100	ON	D32	0x00000100	**********	0x00000100
(09,0,0)	09	0x00000200	OFF	Slave/S*	00000000	0x0ff00000	**********
(09,0,1)	09	0x00000200	ON	Slave/S*	0x00000200	0x0ff00200	**********
(09,1,0)	09	0x00000200	OFF	D16	00000000	**********	00000000
(09,1,1)	09	0x00000200	ON	D16	0x00000200	**********	0x00000200
(09,2,0)	09	0x00000200	OFF	D32	00000000	**********	00000000
(09,2,1)	09	0x00000200	ON	D32	0x00000200	**********	0x00000200
(10,0,0)	10	0x00000400	OFF	Slave/S*	00000000	0x0ff00000	**********
(10,0,1)	10	0x00000400	ON	Slave/S*	0x00000400	0x0ff00400	**********
(10,1,0)	10	0x00000400	OFF	D16	00000000	**********	00000000
(10,1,1)	10	0x00000400	ON	D16	0x00000400	**********	0x00000400
(10,2,0)	10	0x00000400	OFF	D32	00000000	**********	00000000
(10,2,1)	10	0x00000400	ON	D32	0x00000400	**********	0x00000400
(11,0,0)	11	0x00000800	OFF	Slave/S*	00000000	0x0ff00000	**********
(11,0,1)	11	0x00000800	ON	Slave/S*	0x00000800	0x0ff00800	**********
(11,1,0)	11	0x00000800	OFF	D16	00000000	**********	00000000
(11,1,1)	11	0x00000800	ON	D16	0x00000800	**********	0x00000800
(11,2,0)	11	0x00000800	OFF	D32	00000000	**********	00000000
(11,2,1)	11	0x00000800	ON	D32	0x00000800	**********	0x00000800
(12,0,0)	12	0x00001000	OFF	Slave/S*	00000000	0x0ff00000	**********
(12,0,1)	12	0x00001000	ON	Slave/S*	0x00001000	0x0ff01000	**********
(12,1,0)	12	0x00001000	OFF	D16	00000000	**********	00000000
(12,1,1)	12	0x00001000	ON	D16	0x00001000	**********	0x00001000
(12,2,0)	12	0x00001000	OFF	D32	00000000	**********	00000000
(12,2,1)	12	0x00001000	ON	D32	0x00001000	**********	0x00001000
(13,0,0)	13	0x00002000	OFF	Slave/S*	00000000	0x0ff00000	**********
(13,0,1)	13	0x00002000	ON	Slave/S*	0x00002000	0x0ff02000	**********
(13,1,0)	13	0x00002000	OFF	D16	00000000	**********	00000000
(13,1,1)	13	0x00002000	ON	D16	0x00002000	**********	0x00002000
(13,2,0)	13	0x00002000	OFF	D32	00000000	**********	00000000
(13,2,1)	13	0x00002000	ON	D32	0x00002000	**********	0x00002000
(14,0,0)	14	0x00004000	OFF	Slave/S*	00000000	0x0ff00000	**********
(14,0,1)	14	0x00004000	ON	Slave/S*	0x00004000	0x0ff04000	**********
(14,1,0)	14	0x00004000	OFF	D16	00000000	**********	00000000

Table 21-1 *VME Map Table— Continued*

(i,p,s)	bit no.	value	sense	port/ context	VME bus address	virtual address	physical address
(14,1,1)	14	0x00004000	ON	D16	0x00004000	**********	0x00004000
(14,2,0)	14	0x00004000	OFF	D32	00000000	**********	00000000
(14,2,1)	14	0x00004000	ON	D32	0x00004000	**********	0x00004000
(15,0,0)	15	0x00008000	OFF	Slave/S*	00000000	0x0ff00000	**********
(15,0,1)	15	0x00008000	ON	Slave/S*	0x00008000	0x0ff08000	**********
(15,1,0)	15	0x00008000	OFF	D16	00000000	**********	00000000
(15,1,1)	15	0x00008000	ON	D16	0x00008000	**********	0x00008000
(15,2,0)	15	0x00008000	OFF	D32	00000000	**********	00000000
(15,2,1)	15	0x00008000	ON	D32	0x00008000	**********	0x00008000
(16,0,0)	16	0x00010000	OFF	Slave/S*	00000000	0x0ff00000	**********
(16,0,1)	16	0x00010000	ON	Slave/S*	0x00010000	0x0ff10000	**********
(16,1,0)	16	0x00010000	OFF	D16	00000000	**********	00000000
(16,1,1)	16	0x00010000	ON	D16	0x00010000	**********	0x00010000
(16,2,0)	16	0x00010000	OFF	D32	00000000	**********	00000000
(16,2,1)	16	0x00010000	ON	D32	0x00010000	**********	0x00010000
(17,0,0)	17	0x00020000	OFF	Slave/S*	0x00002004	0x0ff02004	**********
(17,0,1)	17	0x00020000	ON	Slave/S*	0x00022004	0x0ff22004	**********
(17,1,0)	17	0x00020000	OFF	D16	0x00002004	**********	0x00002004
(17,1,1)	17	0x00020000	ON	D16	0x00022004	**********	0x00022004
(17,2,0)	17	0x00020000	OFF	D32	0x00002004	**********	0x00002004
(17,2,1)	17	0x00020000	ON	D32	0x00022004	**********	0x00022004
(18,0,0)	18	0x00040000	OFF	Slave/S*	0x00002008	0x0ff02008	**********
(18,0,1)	18	0x00040000	ON	Slave/S*	0x00042008	0x0ff42008	**********
(18,1,0)	18	0x00040000	OFF	D16	0x00002008	**********	0x00002008
(18,1,1)	18	0x00040000	ON	D16	0x00042008	**********	0x00042008
(18,2,0)	18	0x00040000	OFF	D32	0x00002008	**********	0x00002008
(18,2,1)	18	0x00040000	ON	D32	0x00042008	**********	0x00042008
(19,0,0)	19	0x00080000	OFF	Slave/S*	0x00002010	0x0ff02010	**********
(19,0,1)	19	0x00080000	ON	Slave/S*	0x00082010	0x0ff82010	**********
(19,1,0)	19	0x00080000	OFF	D16	0x00002010	**********	0x00002010
(19,1,1)	19	0x00080000	ON	D16	0x00082010	**********	0x00082010
(19,2,0)	19	0x00080000	OFF	D32	0x00002010	**********	0x00002010
(19,2,1)	19	0x00080000	ON	D32	0x00082010	**********	0x00082010
(20,0,0)	20	0x00100000	OFF	Slave/U0	0x80082020	0x00082020	**********
(20,0,1)	20	0x00100000	ON	Slave/U0	0x80182020	0x00182020	**********
(20,1,0)	20	0x00100000	OFF	D16	0x80082020	**********	0x80082020
(20,1,1)	20	0x00100000	ON	D16	0x80182020	**********	0x80182020
(20,2,0)	20	0x00100000	OFF	D32	0x80082020	**********	0x80082020
(20,2,1)	20	0x00100000	ON	D32	0x80182020	**********	0x80182020
(21,0,0)	21	0x00200000	OFF	Slave/U0	0x80082040	0x00082040	**********
(21,0,1)	21	0x00200000	ON	Slave/U0	0x80282040	0x00282040	**********
(21,1,0)	21	0x00200000	OFF	D16	0x80082040	**********	0x80082040
(21,1,1)	21	0x00200000	ON	D16	0x80282040	**********	0x80282040
(21,2,0)	21	0x00200000	OFF	D32	0x80082040	**********	0x80082040
(21,2,1)	21	0x00200000	ON	D32	0x80282040	**********	0x80282040

Table 21-1 *VME Map Table— Continued*

(i,p,s)	*bit no.*	*value*	*sense*	*port/ context*	*VME bus address*	*virtual address*	*physical address*
(22,0,0)	22	0x00400000	OFF	Slave/U0	0x80082080	0x00082080	**********
(22,0,1)	22	0x00400000	ON	Slave/U0	0x80482080	0x00482080	**********
(22,1,0)	22	0x00400000	OFF	D16	0x80082080	**********	0x80082080
(22,1,1)	22	0x00400000	ON	D16	0x80482080	**********	0x80482080
(22,2,0)	22	0x00400000	OFF	D32	0x80082080	**********	0x80082080
(22,2,1)	22	0x00400000	ON	D32	0x80482080	**********	0x80482080
(23,0,0)	23	0x00800000	OFF	Slave/U0	0x80082100	0x00082100	**********
(23,0,1)	23	0x00800000	ON	Slave/U0	0x80882100	0x00882100	**********
(23,1,0)	23	0x00800000	OFF	D16	0x80082100	**********	0x80082100
(23,1,1)	23	0x00800000	ON	D16	0x80882100	**********	0x80882100
(23,2,0)	23	0x00800000	OFF	D32	0x80082100	**********	0x80082100
(23,2,1)	23	0x00800000	ON	D32	0x80882100	**********	0x80882100
(24,0,0)	24	0x01000000	OFF	Slave/U0	0x80082200	0x00082200	**********
(24,0,1)	24	0x01000000	ON	Slave/U0	0x81082200	0x01082200	**********
(24,1,0)	24	0x01000000	OFF	D16	0x80082200	**********	0x80082200
(24,1,1)	24	0x01000000	ON	D16	0x81082200	**********	0x81082200
(24,2,0)	24	0x01000000	OFF	D32	0x80082200	**********	0x80082200
(24,2,1)	24	0x01000000	ON	D32	0x81082200	**********	0x81082200
(25,0,0)	25	0x02000000	OFF	Slave/U0	0x80082400	0x00082400	**********
(25,0,1)	25	0x02000000	ON	Slave/U0	0x82082400	0x02082400	**********
(25,1,0)	25	0x02000000	OFF	D16	0x80082400	**********	0x80082400
(25,1,1)	25	0x02000000	ON	D16	0x82082400	**********	0x82082400
(25,2,0)	25	0x02000000	OFF	D32	0x80082400	**********	0x80082400
(25,2,1)	25	0x02000000	ON	D32	0x82082400	**********	0x82082400
(26,0,0)	26	0x04000000	OFF	Slave/U0	0x80082800	0x00082800	**********
(26,0,1)	26	0x04000000	ON	Slave/U0	0x84082800	0x04082800	**********
(26,1,0)	26	0x04000000	OFF	D16	0x80082800	**********	0x80082800
(26,1,1)	26	0x04000000	ON	D16	0x84082800	**********	0x84082800
(26,2,0)	26	0x04000000	OFF	D32	0x80082800	**********	0x80082800
(26,2,1)	26	0x04000000	ON	D32	0x84082800	**********	0x84082800
(27,0,0)	27	0x08000000	OFF	Slave/U0	0x80083000	0x00083000	**********
(27,0,1)	27	0x08000000	ON	Slave/U0	0x88083000	0x08083000	**********
(27,1,0)	27	0x08000000	OFF	D16	0x80083000	**********	0x80083000
(27,1,1)	27	0x08000000	ON	D16	0x88083000	**********	0x88083000
(27,2,0)	27	0x08000000	OFF	D32	0x80083000	**********	0x80083000
(27,2,1)	27	0x08000000	ON	D32	0x88083000	**********	0x88083000
(28,0,0)	28	0x10000000	OFF	Slave/U0	0x80083004	0x00083004	**********
(28,0,1)	28	0x10000000	ON	Slave/U1	0x90083004	0x00083004	**********
(28,1,0)	28	0x10000000	OFF	D16	0x80083004	**********	0x80083004
(28,1,1)	28	0x10000000	ON	D16	0x90083004	**********	0x90083004
(28,2,0)	28	0x10000000	OFF	D32	0x80083004	**********	0x80083004
(28,2,1)	28	0x10000000	ON	D32	0x90083004	**********	0x90083004
(29,0,0)	29	0x20000000	OFF	Slave/U0	0x80083008	0x00083008	**********
(29,0,1)	29	0x20000000	ON	Slave/U2	0xa0083008	0x00083008	**********
(29,1,0)	29	0x20000000	OFF	D16	0x80083008	**********	0x80083008

Table 21-1 *VME Map Table— Continued*

(i,p,s)	bit no.	value	sense	port/ context	VME bus address	virtual address	physical address
(29,1,1)	29	0x20000000	ON	D16	0xa0083008	**********	0xa0083008
(29,2,0)	29	0x20000000	OFF	D32	0x80083008	**********	0x80083008
(29,2,1)	29	0x20000000	ON	D32	0xa0083008	**********	0xa0083008
(30,0,0)	30	0x40000000	OFF	Slave/U0	0x80083010	0x00083010	**********
(30,0,1)	30	0x40000000	ON	Slave/U4	0xc0083010	0x00083010	**********
(30,1,0)	30	0x40000000	OFF	D16	0x80083010	**********	0x80083010
(30,1,1)	30	0x40000000	ON	D16	0xc0083010	**********	0xc0083010
(30,2,0)	30	0x40000000	OFF	D32	0x80083010	**********	0x80083010
(30,2,1)	30	0x40000000	ON	D32	0xc0083010	**********	0xc0083010
(31,0,0)	31	0x80000000	OFF	Slave/S*	0x00083020	0x0ff83020	**********
(31,0,1)	31	0x80000000	ON	Slave/U0	0x80083024	0x00083024	**********
(31,1,0)	31	0x80000000	OFF	D16	0x00083020	**********	0x00083020
(31,1,1)	31	0x80000000	ON	D16	0x80083024	**********	0x80083024
(31,2,0)	31	0x80000000	OFF	D32	0x00083020	**********	0x00083020
(31,2,1)	31	0x80000000	ON	D32	0x80083024	**********	0x80083024

A

SunDiagnostic Executive Bug Report Form

A

SunDiagnostic Executive Bug Report Form

A.1. Overview

Use this form to report bugs found in the executive programs. To report a SunDiagnostic Executive software problem, please do the following:

Isolate the problem

Try to repeat the error. Can you make it happen consistently? If not, how often does it occur? Does the error happen on other machines?

Describe the environment

Carefully describe the hardware environment of the machine with the problem. Are you using any unusual hardware configurations?

Following these steps is very important. We need detailed information in order to repeat the bug in our labs. Some problems, like an obscure error message, don't need this level of detail. If you are in doubt, the more information you supply, the better our chances of fixing the problem.

A.2. Who to Send the report to

When you have completed filling in this form, please mail it to the following address.

```
Attn: Customer Support Engineering
      Mail Stop 2-34
      Sun Microsystems
      2550 Garcia Avenue
      Mountain View, CA
            94043
```

If it is more convenient, and you have access to the uucp network, you can send the information in electronic form to *sun!exec-bugs*. Please try to conform to the format of this form as closely as possible.

A.3. Who we can contact

Please provide the name and telephone number of someone we can reach to get more information about the problem. It is also useful to have your account number for our records.

Contact

Name: _____

Phone: _____

Account

Name: _____

Number: _____

A.4. Description of Problem

Please answer the following questions about the bug in question

Diagnostic

Part of Executive System that failed _____

How often does it fail?

☐ fails every time

☐ fails about *failures* out of *times_run* times

☐ fails erratically

☐ fails when *event* happens: *event* = _____

_____

Part number of the Tape: _____

How did you boot the SunDiagnostic Executive?

☐ from local tape

☐ from remote tape

☐ from disk (what disk, partition?) _____

Hardware Environment

Type of Machine:

_____

Serial Number of Machine:

_____

Mbytes of Memory:

_____

PROM Revision:

Perform a **kb** command from the monitor prompt to determine your Boot PROM revision level.

_____

Disk(s) Used:

_____

Special Hardware:

_____

Type of Hardware Error

Describe the hardware problem that made you run the executive.

What part of the system failed?: _____

What are the symptoms?: _____

Type of Executive Error

Describe how the executive failed.

☐ Executive not reporting a hardware problem

☐ Executive reporting a non-existent hardware problem

☐ Executive giving misleading or incomplete Error Messages

☐ Executive Hanging/Not running

Describe reason executive didn't run

☐ Executive Broken

☐ Incorrect Manual description

☐ Other: _____

Any other error information

_____

_____

Action Taken:

_____

_____

_____

_____

_____

What hardware problem was actually found:

_____

What component was replaced ?

_____

Serial Number of component: _____

Service Order Number:

_____

Manual

Describe inaccuracies or shortcomings you found in the manual

Miscellaneous

Any information you feel is relevant

B

Standalone Cache and ECC Tests

B

Standalone Cache and ECC Tests

B.1. Introduction

This appendix discusses the standalone Sun-3 Cache and ECC Memory Tests. The tests do not run under the SunDiagnostic Executive but are included on the tape.

B.2. Standalone Cache Test

The Sun-3/2xx cache is a virtual, write-back cache that translates through a high speed, associative memory between the virtual address bus and the 64-bit Sun-3/2xx memory bus.

How the Cache Functions

When enabled, all type 0 memory addresses except video are cached if mapped cache enabled. That is, when a virtual memory address is accessed, the 16 byte memory block containing that address is read into the cache. If there is valid memory data already in the referenced cache block one of two things can occur: if the block is dirty, that is, if it has been modified since being read into the cache, it is first written back to memory prior to reading in the accessed block. If not, it is merely overwritten. In any case, the cache always contains the most recently accessed memory data.

Three other concepts require some introduction: cache hit, cache miss, and cache flush.

A cache hit occurs with the cache enabled when a virtual memory access occurs and the cache block for that address is in the cache. In this case the data access occurs within the cache and not to memory. A cache miss occurs when the data for the access address is not in the cache. In this case the entire 16-byte block for that address is read into the cache. If valid data for a different set associate address was already in the cache and had been modified since being read into the cache, it is first written back to memory prior to reading in the accessed block, if not, it is merely overwritten.

The last concept to be defined is that of a cache flush. The cache flush operation is performed in order to purge the cache of any modified (dirty) data in order to make memory consistent with the cache and invalidate all valid tags within the cache. This is typically done prior to switching context or remapping memory.

B.3. Diagnostic Function

The program performs cache design verification. It tests all types of cache operations. It verifies byte alignment/misalignment between the memory and the CPU by the bypass path, cache to CPU for all byte, word, and longword write and read operations.

It permits selection of standard test control operations such as looping/not looping on error, and printing or not printing error messages. It displays a meaningful LED pattern that indicates which test is being performed.

B.4. Hardware Requirements

The program requires at least the CPU board and a working Sun-3/2xx memory board, which must be the lowest numbered memory board in the system.

At least the CPU to EPROM and memory data paths must be functional in order load and execute the test. Also, a boot path (ethernet, disk, or tape) must be functional in order to boot load the standalone program into Sun-3/200 memory. For the cache logic, only the cache off/cache miss path between the CPU and Sun-3/2xx memory must be functional in order for the program's instructions and data to be fetched and executed from Sun-3/2xx memory.

The power-up Boot PROM tests will have verified that the MMU, Cache, video ram, and data path to memory are functional. The Sun-3/2xx CPU board should have a Boot PROM revision of 1.7 or later.

B.5. Limitations

The program uses the current, Sun-3 standalone token eat/parse user command interface. No error logging is performed; errors are reported as they occur.

B.6. Loading and Starting

You must be in the PROM monitor program in order to extract this standalone diagnostic from the SunDiagnostic Executive tape, if it is not already present in a directory such as `/pub/stand`.

When you have the > monitor prompt, type in **`k2`** to reset the Sun-3/2xx to its initial state.

NOTE *This diagnostic will not function correctly if the reset it not performed.*

Check the SunDiagnostic Executive tape table of contents shown in *Chapter 2* of the *SunDiagnostic Executive User's Guide* for the `cache3.diag` file number. Convert the number to decimal and boot the diagnostic directly off the tape, as shown in Chapter 2.

If cache3.diag is already loaded onto disk, you may boot it with a command such as:

```
b stand/cache3.diag
```

User Command Interface

The cache tests described in this appendix are selected from a main menu. After specifying the test control options, you enter the appropriate command to execute the test.

The menu looks something like this:

```
Sun3xxx Cache Diagnostic Rev x.x mm/dd/yy

  Selections:

    d - cache data tests menu
    t - cache tag tests menu
    r - cache read hit menu
    w - cache write hit menu
    R - cache read miss menu
    W - cache write miss menu
    p - cache phy address cmpr menu
    e - cache writeback error menu
    f - cache flush menu
    b - block copy menu
    q - quick tests
    D - default tests
    P - default single pass
    E - exerciser menu
    X - exerciser tests
    M - cache RAM memory
    l - long tests
    L - loop
    o - options menu
    ? - help
    ^ - quit

  Command :
```

If option `d` on the main menu is selected, a sub-menu containing all of the various cache data tests is presented. The cache data tests menu is discussed later.

Option **t** on the main menu brings up a sub-menu containing all of the various cache tag tests. The cache tag tests menu is discussed later.

Option **r** on the main menu brings up a sub-menu containing all of the cache read hit tests. The cache read hit tests menu is discussed later.

Option **w** on the main menu brings up a sub-menu containing all of the various cache write hit tests. The cache write hit menu is discussed later.

Option **m** brings up the Cache MMU Tests menu, which is discussed later.

Option **R** on the main menu brings up a sub-menu containing all of the various cache read miss tests. The cache read hit menu is discussed later.

Option **W** on the main menu brings up a sub-menu containing all of the various cache write miss tests. The cache write miss menu is discussed later.

Option **p** on the main menu brings up a sub-menu containing all of the various cache physical address compare tests. The cache physical address compare menu is discussed later.

Option **e** on the main menu brings up a sub-menu containing all of the various cache writeback error tests. The cache writeback error tests menu is discussed later.

Option **f** on the main menu brings up a sub-menu containing all of the various cache flush tests.

Option **b** on the main menu brings up a sub-menu containing all of the block copy test options. The block copy tests menu is discussed later.

Option **q** on the main menu executes a quick test sequence, one time. It is discussed under *Test Sequences.*

Option **D** on the main menu executes the default test sequence of tests continuously. At the end of each pass of the default test sequence, a message listing the pass count, and the total errors which have occurred since starting the default test sequence will be displayed.

Option **P** executes a single pass of default tests, discussed under *Test Sequences.*

Option **E** on the main menu brings up a sub-menu containing all of the various exerciser tests The exerciser tests menu is discussed under *Test Sequences.*

Option **X** on the main menu selects the exerciser tests sequence, which runs continuously. At the end of each pass of the exerciser test sequence, a message listing the pass count, and the total errors which have occurred since starting the test sequence will be displayed.

Option **M** brings up the Cache RAM Memory menu, discussed under *Test Sequences.*

Option **l** on the main menu selects the long tests sequence. which executes continuously. At the end of each pass of the long test sequence, a message listing the pass count, and the total errors which have occurred since starting the default test sequence will be displayed.

Option L on the main menu permits looping a command sequence. To use the L command, you enter a command string in which each command and its arguments are followed by a space and a ";" character. You then terminate the command string with the number of times to loop the command. For example, to perform the cache data pattern write/read test with a pattern of all ones followed by the cache data address test for two times, you would enter the following command string: sequence:

```
d ; P ; 0 ; fffc ; a ; 0 ; fffc ; ^ ; L 2
```

Or, to perform it an infinite number of times, enter * for the loop count argument as follows:

```
d ; P ; 0 ; fffc ; a ; 0 ; fffc ; ^ ; L *
```

If option o is selected, a sub-menu containing all of the various test control options.

Option Menu

The Option Menu provides three choices and access to other level menus. When the option is set it applies to all tests that execute after that. The option menu looks something like this:

```
Sun3 xxx Cache Diagnostic Rev x.x mm/dd/yy

  Selections:

    l - loop on error
    h - halt on error
    n - inhibit error messages
    ^ - pop up a menu level
    ? - help

  Command :
```

Option `l` causes a loop on the failing test.

Option `h` causes the test to halt upon error and return to the menu command input.

Option `n` causes *only* fatal error messages to be printed.

Entering `^` brings up the next higher level menu.

The Cache Data Tests Menu

The Cache Data Tests Menu has three commands and provides access to other menus. The Cache Data Tests test the cache control space accessed data RAM. The Cache Data Tests Menu looks something like this:

```
Sun3 xxx Cache Diagnostic Rev x.x mm/dd/yy

  Selections:

    d - cache data write/read test
    a - cache data address test
    p - cache data 3-pattern test
    ^ - pop up a menu level
    ? - help

  Command :
```

The Cache Data Tests Menu selections are described below. The tests themselves are described later.

Entering the **d** command causes the execution of the cache data write/read test.

Entering the **a** command causes the execution of the cache data address test.

Entering the **p** command causes the execution of the cache data 3-pattern test.

The `^` command brings you up to the menu above the one you are in.

Cache Tags Tests Menu

```
Sun3 xxx Cache Diagnostic Rev x.x mm/dd/yy

  Selections:

    d - cache tags write/read test
    p - cache tags 3-pattern test
    ^ - pop up a menu level
    ? - help

  Command :
```

The Cache Tags Data Tests Menu selections are described below. The tests are described later.

Entering the **d** command causes the execution of the cache tags write/read test.

Entering the **p** command causes the execution of the cache tags 3-pattern data test.

The Cache Read Hit Tests Menu

```
Sun3 XXX  Cache Diagnostic Rev x.x. mm/dd/yy

Selections:

    r - cache read hit test
    c - cache read hit (cx different) test
    b - cache read hit byte alignment test
    l - cache read hit longword alignment test
    ^ - pop up a menu level
    ? - help

  Command :
```

The Cache Read Hit Tests Menu selections are described below.

Entering the

r *[addr > 0x20000] [size] [npass]*

command causes the execution of the cache read hit test to be performed from base address *addr* which must be greater than 0x20000 for memory *size* for a number of passes entered in place of *npass*.

Entering the

c *[addr > 0x20000] [size] [npass]*

command causes the execution of the cache read hit (cx different) test to be performed from base address *addr*, which must be greater than 0x20000 for memory size specified by *size* for a number of passes specified by *npass*.

Entering the

b *[npass]*

command will cause the execution of the cache read hit byte alignment test to be

performed for a number of passes entered in place of *npass*.

Entering the

l *[npass]*

command causes the execution of the cache read hit longword alignment test to be performed for a number of passes given in place of *npass*.

The Cache Writeback Error Tests Menu

```
Sun3 XXXX Cache Diagnostic Rev X.X mm/dd/yy

  Selections:

    t - cache writeback timeout err test
    e - cache writeback translation err test
    ^ - pop up a menu level
    ? - help

  Command :
```

The Cache MMU Tests Menu selections are described below.

Entering the

t *[npass]*

command executes the cache writeback timeout err test for the number of passes given in place of *npass*.

Entering the

e *[npass]*

command executes the cache writeback translation err test for the given number of passes.

The Cache Write Hit Tests Menu

```
Sun3 Sirius Cache Diagnostic Rev x.x mm/dd/yy

Selections:

    w - cache first write hit test
    m - cache modify write hit test
    b - cache write hit byte alignment test
    l - cache write hit longword alignment test
    ^ - pop up a menu level
    ? - help

  Command :
```

Entering the

w *[addr > 0x20000] [size] [npass]*

command executes the cache write hit test, performed from base address *addr*,

which must be greater than 0x20000 for memory size specified by *size*. The number of passes is specified by *npass*.

Entering the

m *[addr > 0x20000] [size] [npass]*

command executes the cache modify write hit test, performed from base address *addr*, which must be greater than 0x20000 for the memory size specified by *size*. The number of passes is specified by *npass*.

Entering the

b *[npass]*

command executes the cache write hit byte alignment test, performed for a given number of passes.

The Cache Read Miss Tests Menu

```
Sun3 XXX Cache Diagnostic Rev x.x mm/dd/yy

  Selections:

    n - cache read miss/no writeback (not dirty) test
    d - cache read miss/writeback (valid & dirty) test
    ^ - pop up a menu level
    ? - help

  Command :
```

The Cache Read Miss Tests Menu selections are described below.

Entering the

n *[addr > 0x20000] [size] [npass]*

command executes the cache read miss/no writeback (not dirty) test, performed from base address *addr*, which must be greater than 0x20000 for the memory size specified by *size*. The number of passes is specified in place of *npass*.

Entering the

d *[addr > 0x20000] [size] [npass]*

command executes the cache read miss/writeback (valid & dirty) test, performed from base address *addr*, which must be greater than 0x20000 for the memory size specified by *size*. Enter the number of passes in place of *npass*.

The Cache Write Miss Tests Menu

```
Sun3 xxx Cache Diagnostic Rev x.x mm/dd/yy

Selections:

    n - cache write miss/no writeback (not dirty) test
    d - cache write miss/writeback (valid & dirty) test
    ^ - pop up a menu level
    ? - help

  Command :
```

The Cache Write Miss Tests Menu selections are described below. Entering the

n *[addr > 0x20000] [size] [npass]*

command executes cache write miss/no writeback (not dirty) test, performed from base address *addr*, which must be greater than 0x20000 for the memory size specified by *size*. Specify the number of passes in place of *npass*.

Entering the

d *[addr > 0x20000] [size] [npass]*

command executes the cache write miss/writeback (valid & dirty) test, performed from base address *addr*, which must be greater than 0x20000 for the memory size specified by *size*. Specify the number of passes in place of *npass*.

The Cache Block Copy Tests Menu

```
Sun3 xxx Cache Diagnostic Rev x.x mm/dd/yy

  Selections:
    m - bcopy (src & des blks invalid) test
    s - bcopy (src valid, des invalid) test
    d - bcopy (src invalid, des valid) test
    v - bcopy (src valid, des valid) test
    ^ - pop up a menu level
    ? - help

  Command :
```

The following paragraphs describe the Cache Block Copy Tests Menu selections.

Entering the

m *[addr > 0x20000] [size <= 0x8000] [npass]*

command executes the bcopy (src & des blks invalid) test, performed from base address *addr*, which must be greater than 0x20000 for the memory size specified by *size <= 0x8000* (which must be less than 0x8000). Enter the number of passes in place of *npass*.

LP Entering the

s *[addr > 0x20000] [size <= 0x8000] [npass]*

command executes the bcopy (src valid, des invalid) test, performed from base address *addr*, which must be greater than 0x20000 for the memory size specified by *size <= 0x8000* (which must be less than 0x8000). Replace *npass* with the number of passes.

Entering the

m *[addr > 0x20000] [size <= 0x8000] [npass]*

command executes bcopy (src invalid, des valid) test, performed from base address *addr*, which must be greater than 0x20000 for the memory size specified by *size <= 0x8000*. Replace *npass* with the number of passes.

Entering the

m *[addr > 0x20000] [size <= 0x8000] [npass]*

command executes bcopy (src valid, des valid) test, performed from base address *addr*, which must be greater than 0x20000 for the memory size specified by *size <= 0x8000*. Replace *npass* with the number of passes.

The Cache Flush Tests Menu

```
Sun3 xxx Cache Diagnostic Rev x.x mm/dd/yy

  Selections:

    c - context flush test
    p - page flush test
    s - segment flush test
    m - read mod write flush (single cx) test
    n - read mod write flush (multiple cx) test
    i - read mod write flush (interspersed cx) test
    ^ - pop up a menu level
    ? - help

  Command :
```

The following paragraphs describe the Cache Flush Tests Menu selections.

Entering the

c *[addr >=0x20000] [size <= 0x10000] [npass]*

command executes the context flush test, performed from the base address *addr >=0x20000*, for memory size *size <= 0x10000*. Replace *npass* with the number of passes.

Entering the

p *[page_base_addr] [size <= 0x2000] [npass]*

command executes the Page Flush test, performed from the base address *page_base_addr* for memory size *size <= 0x2000*. Replace *npass* with the number of passes.

Entering the

s *[segment_base_addr] [size >= 0x20000] [npass]*

command executes the Segment Flush Test, performed from the base address *segment_base_address* for memory size *size >=0x20000*. Replace *npass* with the number of passes.

Entering the

m *[addr >=0x20000] [size <= 0x10000] [npass]*

command executes the Read Mod Write Flush (single cx) test, performed from the base address *addr >=0x20000* for memory size *size <= 0x10000*. Replace *npass* with the number of passes.

Entering the

i *[addr >=0x20000] [size <=0x10000] [npass]*

command executes the Read Mod Write Flush (interspersed cx) test to be performed from the base address *addr >=0x20000* for memory size *size <=0x10000*. Replace *npass* with the number of passes.

The Cache Physical Address Compare Tests Menu

```
Sun3 xxx Cache Diagnostic Rev x.x mm/dd/yy

  Selections:

    r - cache read/physical address compare test
    w - cache write/physical address compare test
    ^ - pop up a menu level
    ? - help

  Command :
```

Entering the

r *[address_is_mult_of_0x20000] [size >= 0x20000] [npass]*

command executes the Cache Read/Physical Address Compare test, performed from the base address *address_is_mult_of_0x20000* for memory size *size >= 0x20000*. Replace *npass* with the number of passes.

Entering the

w *[address_is_mult_of_0x20000] [size >= 0x20000] [npass]*

command executes the Cache Write/Physical Address Compare test, performed from the base address *address_is_mult_of_0x20000* for memory size *size >= 0x20000*. Replace *npass* with the number of passes.

The Exerciser Tests Menu

```
Sun3 xxx Cache Diagnostic Rev x.x mm/dd/yy

Selections:

  w - cached memory write/read test
  f - cached memory write/flush/read test
  e - cached execution/cached memory write/flush/read test
  ^ - pop up a menu level
  ? - help

Command :
```

The following paragraphs describe the Exerciser Tests Menu selections.

Entering the

w *[addr] [size <= memory_available] [npass]*

command executes the Cached Memory Write/Read test, performed from the base address for the memory size *size <= memory_available.* Replace *npass* with the number of passes.

Entering the

f *[addr] [size <= memory_available] [npass]*

command executes the Cached Memory Write/Flush/Read test, performed from the base address for the memory size *size <= memory_available*, *npass* times.

Entering the

e *[addr] [size <= memory_available] [npass]*

command executes the Cached Execution Memory Write/Flush/Read test, performed from the base address for the memory size *size <= memory_available* for a number of passes specified by *npass*.

Exerciser Test Sequence

Option X on the main menu selects the exerciser tests sequence, which executes continuously. At the end of each pass of the exerciser test sequence, a message listing the pass count, and the total errors which have occurred since starting the default test sequence will be displayed.

B.7. Test Descriptions

The tests described in this section are:

- Cache data tests
- Cache tag tests
- Cache read hit tests
- Cache write hit tests
- Cache MMU tests
- Cache read miss tests

- Cache write miss tests
- Cache physical address compare tests
- Cache writeback error tests
- Cache flush tests
- Block copy tests

Cache Data Write/Read Test

The Cache Data Write/Read Test is a test of the write/read data integrity of the cache data static RAM on the CPU board. The cache data space, which contains 64 Kilobytes of long-word addressable data, is write/read tested by writing each address with a data pattern, then inverting the pattern and writing the next address. This process is followed by a read back of the original address and a compare of the data read with the original, noninverted longword data pattern. This test insures that all signals dynamically swing within the allowed access time. The patterns used insure that every bit of the cache data RAM is written with a one and zero and that adjacent RAM bits are different.

Error Description

Upon error, the test displays the failing address and longword data written and read, where *addr* is the cache data control space address, *exp* is the expected cache read longword data, and *obs* is the observed longword read data. If the loop-on-error option is set, it will then enter a scopeloop in which the failing write pattern is written to the failing address followed by a write of the inverse pattern to the next address, and finally a read of the failing address. The error message would look something like this:

```
addr 0x90000000 exp 0x5A972C5A, obs 0x5B972C5A
```

Cache Data Address Test

For all cache data addresses, each address is written with a longword data pattern which is the address. Then all addresses are read back and compared with the expected address as data pattern. This test verifies the addressing uniqueness of cache data RAM.

Error Description

Upon error, if the loop-on-error option is set, the test loops through the entire write/read pattern, and is therefore not ideal for scope looping except to verify the binary weighting of each address line. An error message would look something like this:

```
addr 0x80000040 exp 0x00000040, obs 0x00000000
```

Cache Inverse Data Address Test

For all cache data addresses, this test writes the inverse of the address as data at each address of cache data control space. Then, it reads back all addresses to verify that each cache data address contains its inverse address as data. This is a test of addressing uniqueness of the cache data store in control space. Upon error, the test loops thru the entire write/read and is therefore not ideal for scope looping except the binary weighting of the address lines which address the cache data space.

Error Description

Typical error messages are:

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

Cache Data 3-Pattern Test

Three passes of write/read tests are performed in the cache data control space. During the first pass, the pattern 0x5A972C5A,0x5A5A972C,0x2C5A5A97 is written repeatedly throughout cache data control space. During the second pass, 0x5A5A972C,0x2C5A5A97,0x5A972C5A is written and read throughout cache data control space. During the third pass, the patterns 0x2C5A5A97,0x5A972C5A,0x5A5A972C are written and read throughout cache data control space.

Error Description

Upon error, the test displays the failing address and longword data written and read, where *addr* is the cache data control space address, *exp* is the expected cache read longword data and *obs* is the observed longword read data. Upon error, if the loop-on error option is set, the entire pattern write/read throughout the cache data space is looped. This test is not ideal for scope looping! An error message would look something like this:

```
addr 0x90000000 exp 0x5A972C5A, obs 0x5B972C5A
```

Cache Data Pattern Write/Read Test

The Cache Data Pattern Write/Read Test writes and reads each address of the cache data blocks in device control space. This test writes the address with a pattern then inverts the pattern and writes the next address, then reads back the original address and compares it with the noninverted pattern.

Error Description

Typical error messages are:

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

Cache Data Walking Ones Test

For each cache data address the address is written with a float one pattern then read back to verify the correctness of the float one pattern.

Error Description

Typical error messages are:

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

Cache Data Walking Zeros Test

For each cache data address the address is written with a float zero pattern then read back to verify the correctness of the float zero pattern.

Error Description

Typical error messages are:

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

Cache Tags Write/Read Test

Each address of the cache tags static RAM in device control space is written with a data pattern. Then, the next address is written with the data pattern inverted, followed by a read and data compare of the first address.

Error Description

Upon error, the test will display the failing address and longword data written and read, where *addr* is the cache data control space address, *exp* is the expected cache read longword data, and *obs* is the observed longword read data. It will then enter a scopeloop in which the failing write pattern is written to the failing address followed by a write of the inverse pattern to the next address, and finally a read of the failing address. An error message would look something like this:

```
addr 0x90000000 exp 0x5A972C5A, obs 0x5B972C5A
```

Cache Tags 3-Pattern Test

Three passes of write/read tests are performed in the cache tag control space. During the first pass, the pattern 0x4A972400, 0x4A5A1700, 0x0C5A1200 is written repeatedly throughout cache tag control space. During the second pass, 0x4A5A1700, 0x0C5A1A00, 0x4A972400 is written and read throughout cache tag control space. During the third pass, the patterns 0x0C5A1A0C, 0x4A972400, 0x4A5A1700 are written and read throughout cache tag control space. Upon error, if the loop on error option is set, the entire pattern write/read throughout the cache tag space is looped. This test is not ideal for scope looping!

Cache Inverse Tag Address Test

For all cache tag addresses, this test writes the inverse of the address as data at each address of cache tag control space. Then it reads back all addresses and verifies that each cache tag address contains its inverse address as data. This is a test of addressing uniqueness of the cache tag data stored in control space. Upon error, the test loops through the entire write/read and is therefore not ideal for scope looping except the binary weighting of the address lines that address the cache data space.

Error Description

Typical error messages are:

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

Cache Tags Pattern Write/Read Test

This test writes and reads each address of the cache tag blocks in device control space. It writes the address with a pattern then inverts the pattern and writes the next address, then reads back the original address and compares it with the noninverted pattern.

Error Description

Typical error messages are:

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

Cache Tags Walking Ones Test

For each cache tag address the address is written with a float one pattern then read back to verify the correctness of the float one pattern.

Error Description

Typical error messages are:

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

Cache Tags Walking Zeros Test

For each cache tag address the address is written with a float zero pattern then read back to verify the correctness of the float zero pattern.

Error Description

Typical error messages are:

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

Cache Read Hit Test

This test verifies that doing a supervisor data operand read from system memory with the memory block address in cache "valid" causes a read from the cache and not from system memory. All memory being tested is first mapped to be cache-enabled then is written with a longword address pattern. The address being tested within the cache data space is written with an inverse address pattern and the cache block tags are set to valid, and, not modified. The cache is then enabled and the address is read, and then the cache is disabled. The data read is then compared with that expected; if it is the inverse of the address being read, it is correct, if not, it is incorrect. Thus the only data operand fetch that occurs from the cache is when the cache is enabled. This provides a timing window for scoping the test when troubleshooting.

Error Description

Upon error, the test displays the virtual address being read and the expected and observed read data. If the loop-on-error option is set, it will then enter a scope loop where the entire cache setup, enable, read, and disable sequence occurs. The relevant read cycle from the cache occurs when the data operand read during the enable/disable cache "window" occurs.

Error messages would look something like this:

```
With access address in cache, a cache hit shld occur

addr 0x00000010, exp 0xFFFFFFEF, obs 0x00000010
```

Cache Read Hit (context different) Test

This test is the same as the Cache Read Hit Test except that the context I.D. tag bits in cache for the block of addresses being tested are set differently from the context within which the test is executing. This is done to verify that, for supervisor mode, reads to the context of the cached data are "don't care" and a cache read hit rather than a miss will occur.

Error Description

Upon error, the test displays the virtual address being read and the expected and observed read data. It then enters a scope loop where the entire cache setup, enable, read, and disable sequence occurs. The relevant read cycle from the cache occurs when the data operand read during the enable/disable cache "window" occurs.

An error message would look something like this:

```
With access address in cache, a cache hit shld occur

addr 0x00000010, exp 0xFFFFFFEF, obs 0x00000010
```

Cache Read Hit User Violation Test

For the address being tested, the cache tags are set to valid, not modified, supervisor only. A read hit is then attempted in user mode for the cached address and a bus error with correct protection error status in the bus error register is verified.

Error Description

If no bus error occurs upon the attempted user mode read of the supervisor only protected block, the following error message and status is displayed:

```
A read hit of a supervisor only accessible address shld cause a bus error.

addr 0x00000010
```

If the read actually does occur, the following error message is displayed.

```
Attempting to read a supervisor only accessible address shld not read it.

addr 0x00000010, exp 0x00000000, obs 0xFFFFFFEF
```

If the bus error register does not indicate a protection bus error, this message is displayed:

```
A supervisor only user access error shld set prot err status in bus error reg.

bus err reg: exp 0x00000040, obs 0x00000000
```

Cache Read Byte Hit Byte Alignment (within block) Test

The byte alignment for all byte addresses within a cache block is verified for all move byte read transfers that are cache hits. To do this, the test stores a block pattern of 0x00010203, 0x10111213, 0x20212223, and 0x30313233 in a cache block and performs move byte read transfers of each byte.

Error Description

If the byte read into the CPU register is not correct, that is, is misaligned in its transfer from the cache to the CPU, the following error message is displayed:

```
Read hit byte alignment error during a byte read

addr 0x00000010, exp 0x00000020, obs 0x00000021
```

Cache Read Longword Hit Byte Alignment (Within Block) Test

The byte alignment for all byte addresses within a cache block is verified for all move longword read transfers that are cache hits. To do this, the test stores a block pattern of 0x00010203, 0x10111213, 0x20212223, and 0x30313233 in a cache block and performs move longword read transfers of each byte.

If the longword read into the CPU register is not correct, that is, if it is misaligned in its transfer from the cache to the CPU, an error message that look something like this is displayed: error message is displayed:

```
Read hit byte alignment error during a longword read

addr 0x00000010, exp 0x00000020, obs 0x00000021
```

Cache Read Miss/No Writeback (invalid) Test

For the address being tested, the cache tags for that block are set up to mark the block as invalid and inverse address data is stored in the cache block for that address. The address is then read during which the cache block should be updated from memory and the block marked as valid. The test checks to insure that the entire cache block is updated from memory correctly and that the cache block is set to valid, write enabled, to match the MMU setup for the page being accessed.

Error Description

If the cache block was not updated correctly from memory on the read miss, the following error message is displayed:

```
During a cache read miss, one or more longwords were not read into the cache.

addr 0x00000010, exp 0x00000010, obs 0x00000020
```

If the data was not read correctly by the bypass path from memory to the CPU during the read miss transfer, the following error message is displayed:

```
For a read miss, bypass data to CPU shld = memory data

addr 0x00000010, exp 0x00000010, obs 0x00000020
```

Cache Read Miss/No Writeback (not dirty) Test

For the address being tested, the cache tags for that block are set up to mark the block as valid, and inverse address data is stored in the cache block for that address. The address is then read, during which the cache block should be updated from memory and the block marked as valid. The test checks to insure that the entire cache block is updated from memory correctly and that the cache block is set to valid, write enabled to match the MMU setup for the page being accessed. Also, memory is checked for the old cache block address to verify that the old cache block, which is not dirty, is not written back to memory.

Error Description

If the cache block was not updated correctly from memory on the read miss, the following error message is displayed:

```
During a cache read miss, one or more longwords were not read into the cache.

addr 0x00000010, exp 0x00000010, obs 0x00000020
```

If the data was not read correctly by the bypass path from memory to the CPU during the read miss transfer, the following error message is displayed:

```
For a read miss, bypass data to CPU shld = memory data

addr 0x00000010, exp 0x00000010, obs 0x00000020
```

If the old cache block was written back to memory incorrectly as it was replaced by the new cache block, the following error message is displayed:

```
With old cache entry valid and not dirty, no writeback of it shld occur to memory.

addr 0x00000010, exp 0x00000010, obs 0x00000020
```

Cache Read Miss/Writeback (valid & dirty) Test

For the address being tested, the cache block is set to valid, dirty. A modulo cache size address is then read. The cache block for that set associative address is then checked to verify that the following is true:

1. The old cache block is written back to memory correctly.
2. The memory block for the new address is correctly copied into memory.
3. The tag bits are correctly set to valid, not dirty
4. The MMU protection bits are correctly copied from the MMU for the page being referenced.

In addition, the data read through the bypass path is verified to have been correctly read into the CPU register.

Error Description

If the cache block was not updated correctly from memory on the read miss, the following error message is displayed:

```
During a cache read miss, one or more longwords were not read into the cache.

addr 0x00000010, exp 0x00000010, obs 0x00000020
```

If the data was not read correctly by the bypass path from memory to the CPU during the read miss transfer, the following error message is displayed:

```
For a read miss, bypass data to CPU shld = memory data

addr 0x00000010, exp 0x00000010, obs 0x00000020
```

If the old cache block was written back to memory incorrectly as it was replaced by the new cache block, the following error message is displayed:

```
With old cache entry valid and not dirty, no writeback of it shld occur to memory.

addr 0x00000010, exp 0x00000010, obs 0x00000020
```

If the old cache block for the old address was not correctly written back into memory, the following error message is displayed:

```
With old cache entry valid and dirty, a writeback of it shld occur to memory.

addr 0x00000010, exp 0x00000010, obs 0x00000020
```

Cache Modify Write Hit Test

For the address being tested, the cache block is set to valid, modified, and inverse data is stored in the cache block. The address is then written-to and a cache write hit is verified. The cache block is then checked to verify that it was modified and that the valid, modified status is "1,1" and that the corresponding memory block is verified to have been written correctly.

Error Description

If the write hit incorrectly updated memory, the following error message is displayed:

```
A write hit of a valid cache entry shld not write memory.

mem addr 0x00000010, exp 0x00000010, obs 0xFFFFFFEF
```

If the cache block tags are not updated correctly, the following error message is displayed:

```
Cache entry not set valid and modified or incorrect tag field.

cache addr 0x00000010, exp 0xC0000000, obs 0x80000000
```

Cache Write Hit/ Write Protect Violation Test

For the address being tested, the cache block is set to valid, not write enabled, and inverse data is stored in the cache block. An attempt is then made to write the address, and a bus error is verified to occur with the bus error register status indicating a protection error.

Error Description

If the write hit of the address did not cause a bus error, the following error message is displayed:

```
A write hit to a nonwrite enabled address shld cause a bus error.

mem addr 0x00000010
```

If the bus error register was not set to indicate a protection type bus error, the following error message is displayed:

```
A write hit of a nonwrite enabled address shld set protect err status in bus error register.

bus err reg: exp 0x00000040, obs 0x00000000
```

Cache Write Byte Hit Byte Alignment (within block) Test

The byte alignment for all byte addresses within a cache block is verified for all move byte write transfers that are cache hits. To do this, the test stores a block pattern of all zeros in a cache block and performs move byte write transfers of 0xFF to each byte address of the cache block.

Error Description

If the byte write into the CPU register is not correct, that is, if it is misaligned in its transfer from the cache to the CPU, the following error message is displayed:

```
Read hit byte alignment error during a byte read

addr 0x00000010, exp 0x000000FF, obs 0x0000FF00
```

Cache Write Longword Hit Byte Alignment (within block) Test

The byte alignment for all byte addresses within a cache block is verified for all move longword write transfers that are cache hits. To do this, the test stores a block pattern of all zeros in the cache block and performs move longword write transfers, starting at each of the byte 0-12 boundaries within the cache block.

Error Description

If the longword write into the CPU register is not correct, that is, if it is misaligned in its transfer from the cache to the CPU, the following error message is displayed:

```
Read hit byte alignment error during a longword read.

addr 0x00000010, exp 0x10111213, obs 0x0010111213
```

Cache First Write Hit Test

This test verifies that doing a write operation to a valid but unmodified block in the cache causes the cache data to be updated and the valid, modified tag bits to be set to "1,1".

Error Description

If the memory address is written incorrectly, that is, if a write hit did not occur, the following error message will be displayed:

```
A write hit of a valid cache entry shld not write memory.

mem addr 0x00000010, exp 0x00000010, obs 0xFFFFFFEF
```

If the cache tags were not correctly updated to valid and modified with all other tag bits as expected, the following error message is displayed:

```
Cache entry not set valid and modified or incorrect tag field.

cache addr 0x00000010, exp 0xC0000000, obs 0x80000000
```

Cache Write Miss Tests

A cache "miss" on write cycles in device space may be caused by either an access to a cacheable type 0 page whose data are missing from the cache, or by an access to a "don't cache" page or a non-type 0 page. Reading the MMU page map determines whether the access data should be in the cache. If the MMU translates the access address to a valid type 0 page whose "don't cache" bit is inactive, then the cache will be updated at the block corresponding to the access address on the write cycle. The cache tags will be updated to show the access context and virtual address, the valid bit will be set active and the modified bit will be set.

The following tests may be selected from Cache Write Miss Tests Menu:

1. Cache write miss/no writeback (not dirty) test

2. Cache write miss/writeback (valid & dirty) test

Cache Write Miss/No Writeback (not dirty) Test

This test verifies that doing a supervisor data operand write to system memory with the memory block address not in the cache will cause a replacement and update the cache entry data, tag address and set the valid and the modified bit for the new cache entry. For the old cache entry verify that the old cache entry is not written back where the old cache entry is valid and not modified.

The test is performed in the following steps:

1. Clear all cache tag field addresses in cache tag control space.
2. Clear all cache data control space addresses.
3. For addresses within the user command specified virtual address range:
   (a) clear the cache tag/data entries for the address,
   (b) store the address in the cache tag entry with (valid, modified) bits = (1,0),
   (c) store the longword address inverted as data in cache data control space,
   (d) write the virtual address,
   (e) read by control space the cache tags and data from the cache and verify that the data is updated and the valid bit is set.

Error Descriptions

Typical error messages that may be reported during the test are:

```
With read/merge of cache block data on write miss, the memory
and cpu data was not merged correctly in cache.
cache addr 0x0000000, exp 0x00020000, obs 0x00000000

With old cache entry valid and not dirty, a block writeback of
not occur to memory
write addr: 0x00020000, mem addr 0x00040000, exp 0x00040000, obs
0x00020000

Cache entry not set valid or incorrect tag field
cache addr 0x00000000, exp 0x80000000, obs 0x00000000
```

Cache Write Miss/No Writeback (invalid) Test

This test verifies that doing a supervisor data operand write to system memory with the memory block address not in the cache will cause a replacement and update the cache entry data, tag address and set the valid and the modified bit for the new cache entry. For the old cache entry verify that the old cache entry is not written back where the old cache entry is invalid.

The test is performed in the following steps:

1. Clear all cache tag field addresses in cache tag control space.
2. Clear all cache data control space addresses.
3. For addresses within the user command specified virtual address range:
   (a) clear the cache tag/data entries for the address,
   (b) store the address in the cache tag entry with (valid, modified) bits = (1,0),
   (c) store the longword address inverted as data in cache data control space,

(d) write the virtual address,

(e) read by control space the cache tags and data from the cache and verify that the data is updated and the valid bit is set.

Error Description

Typical error messages that may be reported during the test are:

```
With read/merge of cache block data on write miss, the memory
and cpu data was not merged correctly in cache.
cache addr 0x0000000, exp 0x00020000, obs 0x00000000

With old cache entry invalid , a block writeback of it shld not occur to
memory
write addr: 0x00020000, mem addr 0x00040000, exp 0x00040000, obs 0x00020000

Cache entry not set valid or incorrect tag field
cache addr 0x00000000, exp 0x8000000 obs 0x00000000
```

Cache Write Miss/Writeback (valid & dirty) Test

The test verifies that doing a supervisor data operand write to system memory with the memory block address not in the cache will cause a replacement and update the cache entry data, tag address and set the valid and the modified bit for the new cache entry. It verifies that the old cache entry is written back where the old cache entry is valid and modified.

The test is performed in the following steps:

1. Clear all cache tag field addresses in cache tag control space.
2. Clear all cache data control space addresses.
3. For each virtual address:
   (a) clear the cache tag/data entries for the address,
   (b) store the address in the cache tag entry with (valid, modified) bits = (0,0),
   (c) store the longword address inverted as data in cache data control space,
   (d) read the virtual address and verify that the data read is the not the inverse of the virtual address, but is the address,
   (e) read by control space the cache tags and data from the cache and verify that the data is updated and the valid bit is set.

Error Description

Typical error messages during the test are:

```
With read/merge of cache block data on write miss, the memory and
cpu data was not merged correctly in cache.
cache addr 0x00000000, exp 0x00020000, obs 0x00000000

With old cache entry valid and dirty, a block writeback of it shld occur to memory
write addr: 0x00020000, mem addr 0x00020000, exp 0xFFFDFFFF, obs 0x00020000

Cache entry not set valid or incorrect tag field
cache addr 0x00000000, exp 0x80000000, obs 0x00000000
```

Exerciser Tests

Exerciser tests are tests that perform cache enabled memory write and read tests with comparing memory write and read data. One of the tests also executes the test code in the cache. They are included in the menu of tests in order to provide fast testing of cache write hit, write miss, read hit, read miss and page flush operations without detailed testing of tags and date within the cache. As a result they should be used to quickly test the cache and memory in order to quickly verify cache/memory integrity.

The menu of Exerciser Tests is as follows:

1. Cached memory write/read test
2. Cached memory write/flush/read test
3. Cached execution/cached memory write/flush/read test

Cached Memory Write/Read test

The Cached Memory Write/Read Test enables the cache and then writes all memory from the command specified base virtual address for the command specified memory size, then reads the same virtual memory space back and performs a data compare of written with read data.

The test is performed in the following step sequence:

1. Setup all test memory above the program space to be wr/rd/cache enabled.
1. Perform a write/read test of the test memory space which should cause write hits/read hits/write misses/read misses to occur.
3. After each write/read pass compare memory data and print all data compare errors.

Error Description

Typical error messages during the test are as follows:

```
mem addr 0x20000, exp 0x00020000, obs 0x00000000
```

Cached Execution/Memory Write/Flush/Read Test

The Cached Execution/Memory Write/Flush/Read Test copies the test code into the cache, then performs a write/flush/read test of all of the specified test memory space.

The test is performed in the following step sequence:

1. Copy the test into the cache. Then, perform a write/flush/read test of all of test memory above the cached code space.

2. Setup all test memory above the program space to be wr/rd/cache enabled.
3. Perform a write/read test of the test memory space which should cause write hits/read hits/write misses/read misses to occur.
4. After each write/read pass compare memory data and print all data compare errors.

Error Description

Typical error messages during the test are as follows:

```
mem addr 0x20000, exp 0x00020000, obs 0x00000000
```

Cache Block Copy Tests

Block Copy (read) and Block Copy (write) are commands which, executed in sequence, cause a block of data to be moved from one block address to another while maintaining cache data integrity and avoiding the displacement of any valid blocks from the cache. The source block for a Block Copy sequence may be either a cache block or main memory; the destination is always a block in main memory.

Block Copy (read) and (write) must maintain cache data consistency. For both Block Copy (read) and (write), the virtual address hit criteria are the same as those used in cache read and write accesses. An additional comparison is required for both Block Copy (read) and (write), if the command address "misses" the cache. In this case, the translated physical address for the command must be compared with the translated cache block address. If they match, the command address and cache block address are synonyms, and the command must be handled as a cache "hit".

The destination o the Block Copy (write) command is always main memory. If the Block Copy (write) command "hits" the cache, then the cache block must be invalidated. A protection violation for the Write command will result if the page for the destination block prohibits writes. A protection violation prevents both writing the block and updating the MMU. Otherwise, both the MMU accessed and modified bits will be updated by a valid Block Copy (write) command.

The menu of tests provided for Block Copy testing is as follows:

1. Bcopy (src & des blks invalid) Test
2. Bcopy (src valid, des invalid) Test
3. Bcopy (src invalid, des valid) Test
4. Bcopy (src valid, des valid) Test

Bcopy (src & des blks invalid) Test

The Bcopy (src & des blks invalid) Test performs a block copy operation for the case where both the source and destination blocks are invalid cache blocks within the cache. The test is performed in the following step sequence:

1. Set all cache entries to invalid state.
2. Perform a bcopy read from one memory block to another memory block.
3. Then verify the following:

(a) the corresponding cache block remains invalid.

(b) the memory block (4 longwords) was copied correctly.

(c) Repeat for all blocks for address range of interest.

Error Description

Typical error messages during the test are as follows:

```
A bcopy operation has failed to copy data correctly
src addr 0x00020000, src data 0x00020000, des addr 0x00040000,
des data 0x00040000
```

Bcopy (src valid, des invalid) Test

The Bcopy (src valid, des invalid) Test verifies that a block copy read that is a cache hit, reads the block from the cache and leaves the "from" cache block valid and unchanged. The test is performed in the following step sequence:

1. Initialize all cache blocks to be invalid.
2. Set cache block to be valid with inverse address data pattern in it.
3. Perform a bcopy read from the block (in cache and valid) to memory addresses that are not in cache.
4. Then verify the following:
   (a) the valid "from" cache block remains valid and unchanged.
   (b) the memory block (4 longwords) was copied correctly from the cache block and not from memory.
   (c) the "to" cache block (since no cache hit) is unchanged.
5. Repeat for all blocks for address range of interest.

Error Description

Typical error messages during the test are as follows:

```
A bcopy operation has failed to copy data correctly
src addr 0x00020000, src data 0x00020000, des addr 0x00040000,
des data 0x00040000

A destination cache block was set valid during a bcopy operation!
cache addr 0x00040000, exp tag data 0x00000000, obs 0x80000000
```

Bcopy (src invalid, des valid) Test

The Bcopy (src invalid, des valid) Test verifies that for a block copy read that is in cache, reads from the block from memory cause a cache write hit, and invalidate the "to" cache block. The test is is performed in the following step sequence:

1. Initialize all cache blocks to be invalid.
2. Set "to" cache block to be valid with inverse address data pattern in it.
3. Perform a bcopy read from the block in memory to the block in cache
4. Then verify the following:

(a) the valid "to" cache block is invalidated.

(b) the memory block (4 longwords) was copied correctly from the memory block to the "to" memory block.

5. Repeat for all blocks for address range of interest.

Error Description

Typical error messages during the test are as follows:

```
A bcopy operation has failed to copy data correctly
src addr 0x00020000, src data 0x00020000, des addr 0x00040000,
des data 0x00040000

One or more valid destination cache blocks were not invalidated
during bcopy operation
cacheW tag addr 0x00000000, tag data exp 0x00000000,
tag data obs 0x80000000
```

Bcopy (src valid, des valid) Test

The Bcopy (src valid, des valid) Test verifies that for a block copy read that is in cache reads from the block in cache ,causes a cache write hit, and invalidates the "to" cache block. The test is is performed in the following step sequence:

1. Store an address pattern in memory.
2. Initialize all cache blocks to be invalid.
3. Set "to" cache block to be valid with inverse address data pattern in it.
4. Set the "from" cache block to be valid with inverse address data
5. Perform a bcopy read from the block in cache to the block in cache
6. Then verify the following:
   (a) the valid "to" cache block is invalidated.
   (b) the memory block (4 longwords) was copied correctly from the cache block to the "to" memory block.
   (c) the "from" cache block is still valid.
5. Repeat for all blocks for address range of interest.

Error Description

Typical error messages during the test are as follows:

```
A bcopy operation has failed to copy data correctly
src addr 0x00020000, src data 0x00020000, des addr 0x00040000,
des data 0x00040000

One or more valid destination cache blocks were not invalidate
during bcopy operation
cache tag addr 0x00000000, tag data exp 0x00000000,
tag data obs 0x80000000
```

Cache Exerciser Tests

The cache exerciser tests are a suite of menu selectable tests consisting of the following tests:

1. Cached Memory Write/Read Test
2. Cached Memory Write/Flush/Read Test
3. Cache Fetch NOP Test
4. Cached Execution/Cached Memory Write/Read Test

Cached Memory Write/Read Test

The Cache Memory Write/Read Test maps all test memory above the program space to be write/read/cache enabled. It then performs a write/read test fo the test memory space which should cause write hits/read hits/write misses/read misses to occur. After each write/read pass, memory data is compared and all data compare errors are printed.

Error Description

Typical error messages are:

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

Cached Memory Write/Flush/Read Test

The Cache Memory Write/Flush/Read Test maps all test memory above the program space to be write/read/cache enabled. It then performs a write/flush/read test of the test memory space which should cause write hits/read hits/write misses/read misses to occur. After each write/read pass, memory data is compared and all data compare errors are printed.

Error Description

Typical error messages are:

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

Cached Fetch NOP Test

The test copies a string of `nop` instructions into the cache. It then fetches and executes the `nop`s from the cache. The test verifies the ability to fetch and execute `nop` instructions from the cache. If it fails it fails destructively.

Cached Execution Memory Write/Read Test

The test code is copied into the cache. A write/flush/read of all the test memory is then executed from the cache of test memory above the cached code space. After each write/read pass memory data is compared and all data compare errors are printed.

B.8. Test Sequences

The following Test Sequences are provided and are selectable from the main menu of the program:

1. Quick Test
2. Default Test
3. Single Pass Default Test
4. Long Test
5. Exerciser Test

6. Cache RAM Memory Test

Quick Test

The Quick Test is a short, nonexhaustive test that verifies that the basic cache functions: write/read hits, misses, and flushes work. Addressing is not exhaustively exercised.

Default Test

The Default Test is a selection of all the cache functions tests: write/read hits, misses, and flushes as well as the exerciser tests. Cache and memory addressing is varied to attempt to provide a rigorous test of the cache and memory system. The test will run until stopped by the operator, accumulating all errors which are totaled and printed at the end of each test pass.

Single Pass Default Test

The Single Pass Default Test is a single pass of the Default Test.

Long Test

The Long Test is a longer, more exhaustive version of the default test which will execute one test pass then stop.

Exerciser Test

The Exerciser Test is a selection of tests from the Exerciser Test Menu. The tests execute fast and provide a quick but exhaustive test of the cache and memory system but with limited diagnostic information in error messages.

Cache RAM Memory Test

The Cache RAM Memory Test consists of all the cache data and tag tests from the Cache Data and Cache Tag Menus and is provided to focus testing only upon testing the Cache RAM. The test will cycle forever or until stopped by the operator and will accumulate and error count which is typed at the end of each test pass.

B.9. Glossary

Bootpath
Interface and bus logic from CPU to an I/O boot device

Block
Four longwords of memory data that have tag control/status

Cache
An associative, fast RAM between the CPU and main memory

Cache Block
The smallest number of memory words copied between the cache and memory during a cache to memory transfer.

Cache Hit
A memory access in which the accessed address's data is in the cache

Cache Miss
A memory access in which the accessed address's data is not in the cache

CPU
Central Processing Unit

Dirty
Write modified within the cache

Flush
To copy and invalidate all modified(dirty) cache blocks to memory

Ethernet
Coaxial cable communication link between systems

I/O
Input and output, as for example, an input/output device

LED
Light Emitting Diode

MMU
Memory Management Unit

Page
The smallest contiguous, selectable block of memory through the MMU

PMEG
Page Map Entry Group

POR
Power-on Reset

RAM
Random Access Memory

SCC
Serial Communications Controller

Tags
Control/status bits unique for each cache block

UART
Universal, Asynchronous, Receiver/Transmitter

B.10. Standalone ECC Memory Diagnostic

This standalone diagnostic tests the error checking and correction (ECC) memory for Sun-3 workstation products. It resides on the 1.1 version of the SunDiagnostic Executive tape, but does not run under the Executive program. You must boot it separately, as described under Section B.13 , Loading And Starting .

The layout of the memory array puts 64 bits of data and 8 bits of ECC into the RAM. However, by using partial writes (byte, word, longword) it is possible to determine the memory location that has a failure, even though on reads the CPU reads a 128 bit block of memory. The MC68020 processor only gets the byte, word, or long word requested.

The ECC logic is eight bits wide and generates check bits over each 64-bit word, making the DRAM array 72 bits wide. The ECC logic always writes the check bit code into the check bit DRAMs. The ECC error logic only reports errors when the error reporting bit is set in the Memory Enable Register. These errors are CE for single bit errors (CE corrects the error), and UE for double bit errors (no correction). The ECC logic is designed in such a way as to prevent reading the actual check bit code stored in the check bit DRAMs.

Functional testing is designed to detect permanent faults that cause the memory to function incorrectly. Generally speaking, RAM can be defined as functional if it is possible to store a 0 or a 1 into every cell of the memory, to change every cell from 0 to 1 as well as from 1 to 0, and to read every cell correctly when it stores a 0 as well as when it stores a 1, regardless of the contents of the remaining cells or the previous memory access sequences. A functional test that will cover all the possible faults is impossible from the practical point of view, as such a test may take years to execute! Therefore, in order to develop any practically feasible test procedure, we restrict ourselves to a subset of faults that are most likely to occur.

The following three fault models for RAM are the most widely used:

Stuck-at Faults
: One or more logic values in the memory system cannot be changed. For example, one or more cells are "stuck at" 1 or 0.

Coupling Faults
: There exist two or more cells that are coupled. A pair of memory cells are said to be coupled if a transition in one cell if the pair changes the state of the other cell.

Pattern Sensitivity Faults
: The fault that alters the state of a memory cell as a result of certain patterns of zeros, ones, zero-to-one transitions, and/or one-to-zero transitions in the other cells of the memory is called a pattern sensitivity fault. The fault that causes a read or a write operation of one cell of the memory to fail owing to certain patterns of zeros and ones in the other cells of the memory is also called a pattern sensitivity fault.

Slow Access/Recovery Faults
: During writes, data is not recorded in the specified time. Also, a read occurring immediately after a write may sense bad data if the chip's recovery time exceeds specification.

Refresh Faults
After long inactive periods, data is lost from the memory.

B.11. Hardware Requirements

The ECC Memory Diagnostic tests any ECC memory configuration. Each memory board has an enable register with the memory size encoded in it. With a possibility of four memory boards with any combination of 8 or 32 Megabyte boards, reading the memory enable register determines the memory size of the system. The following list specifies the environmental requirements for this test:

The CPU board is assumed functional (MMU, Cache, video ram, data path to memory) as check out by the boot PROM.

The boot path (ethernet, disk, tape) is assumed checked out so that the memory diagnostic can be loaded into main memory.

The boot PROM aids in user I/O communications.

This diagnostic runs standalone, meaning that the operating system is not needed while memory is being tested.

B.12. Overview Of The Diagnostic

The test patterns are presented to allow flexibility in sequencing as well as customizing of the tests. All tests can be broken down to allow primitive actions upon the memory to be tested on command. At a higher level, a default test sequence is provided with a single command character for ease of use.

The main enhancement of the memory diagnostic is that the memory diagnostic will be copied into CACHE RAM so that all of memory can be tested. This means that the diagnostic will no longer have to relocate itself in memory.

All tests are optimized for speed. While speed is important in terms of the volumes of memory arrays to be tested, it is also important to check for slow access and recovery faults.

Memory Interface

The memory interface will be discussed here as it applies to the Sun-3/2xx implementation of the Sun-3 architecture.

Each memory board is configured with 8 or 32 Megabytes of ECC memory. In addition, each board includes an ECC Memory Enable register, a Correctable Error register (Syndrome register), and an EDC chip Diagnostic register, as defined in the Sun-3 architecture. Sun-3/2xx memory also features a self timed refresh controller that may be enabled for memory scrub.

Each 8 Megabyte memory array is implemented with 288 256k DRAM chips with 120 NS access time. An 8 Megabyte bank is organized as four banks of 2 Megabytes each, with even and odd quadwords (addressed by A3 and ~A3) in each bank, and each 4 Megabytes is divided by address A22 and ~A22.

The Sun-3/2xx memory board is interconnected with the CPU board over the 64-bit bus.

Sun-3/2xx memory supports the operations summarized below:

Block Read/Write
Read from or write into memory a block of 16 bytes, using two full 8 byte

data transfers. Block reads may result form any CPU or DVMA bus cycle to main memory. Whether a cycle is initiated depends on whether the data is cacheable and on the state of the Sun-3 cache on the CPU board. Block writes may result from cache block replacement, cache flushes, or Block Copy operations.

Partial Write
: Write, from a single data transfer of 8 bytes, up to 4 bytes into memory, as specified with an 8 bit Byte Mark field. This field is set by the CPU board to mark bytes being modified by a CPU or DVMA bus cycle. Partial writes only result from write operations into a page in main memory which is marked Don't Cache.

Register Read/Write
: Read from or write into control register on the memory board. Control registers are addressed by the CPU as Type 1 devices in Device Space. Register writes must be partial write bus cycles.

  Data is aligned according to MC68020 conventions. On partial write bus cycles, the write data from the CPU is assumed to be in the proper alignment within a quadword. Similarly, on register transfer cycles, register data for both Read and Write bus cycles is aligned by MC68020 addressing conventions within a quadword. Following the MC68020 convention, the high order byte within a quadword, D<63..56>, is byte 0; the low order byte, D<7..0>, is byte 7.

Error Checking Correction Interface

Each memory board contains four AMD 2960A ECC chips (EDC) configured to generate eight check bits for every 64 data bits. The mode of operation for these chips is controlled through bits D<12..11> of the Enable register, which corresponds to chip inputs DM1 and DM0 of the 2960A.

On the Sun-3/2xx memory board, the EDC chip and external ECC control logic are designed to correct any single bit error and to report as an uncorrectable error (UE) any double bit error. All other errors, including those triple bit errors for which there are defined syndromes, go unreported.

Refresh

Each board includes local refresh control. A data scrub operation (if enabled) may be performed during each refresh cycle.

The refresh period is once every 15.5 microseconds. Refresh requests have higher priority than new bus requests occurring in the same cycle. Once a refresh or memory cycle begins, it completes without interruption.

During data scrub cycles, any single bit error is corrected and written back into memory, while signaling a CE error and recording the CE address. An uncorrectable error detected during a scrub is not reported. No data are written back into memory.

Initialization

Bus error and parity error handling is setup, and the MMU is setup.

To set up the MMU:

(1) The first 64 Kbytes of main memory are mapped cache enable. Then the code is copied into the cache. The diagnostic will execute from here.

(2) The next pages are mapped to the 16 Megabytes of main memory. The logical addresses for the first memory board are 0x14000 to 0x7fc000. The physical address for the first memory board are 0x0 to 0x7fffff. The logical address for memory boards 2, 3, and 4 are 0x814000 to 0x1014000. The physical addresses for memory boards 2, 3, and 4 are 0x800000 to 0xffffff. The reason to execute from Cache Ram is that the ECC logic can not be tested from memory, since an error would cause program execution to stop.

Memory is then sized.

The memory is sized to determine how many memory boards are in the system. There can be a minimum of 1 memory board and a maximum of 4 memory boards. Each memory board contains 8 or 32 Megabytes of DRAM. During testing, each board is separately run through all tests before the next board is tested.

The default parameters are then initialized.

The default parameters are address size of 8 Megabytes, 1 pass count for each test.

B.13. Loading And Starting

The following steps must be followed in the order listed below to run the ECC Memory Diagnostic.

1. Turn on the system.
2. Since the ECC Memory Diagnostic is a standalone diagnostic, the operating system should not be booted. Instead, after self-tests pass, terminate the booting process with the `L1-A` sequence. That is, hold down the [L1] key while pressing the `a` key. On the other hand, if you are interacting with a dumb terminal, use the `break` key.
3. In order to reset the memory maps to their initial state, prior to pressing the [Return] key, type **`k2`**.

NOTE *The ECC Memory diagnostic cannot be booted correctly if this reset step is skipped.*

4. At this point the ECC Memory Diagnostic can be loaded and its execution started. The three ways to load the diagnostic are listed below.

Assuming that the executable version of the ECC Memory Diagnostic resides on the local disk in directory `/pub/stand` or `/stand`, the diagnostic can be loaded by typing the following command line before pressing [Return]

```
b stand/eccmem3.diag
```

If the diagnostic resides on a remote disk, it can be loaded over Ethernet. Assuming that the network file server has a partition reserved for the system

being tested (i.e. the system under test is a client of the file server) and that `eccmem3.diag` resides in the file server's directory `/pub/stand`, the ECC Memory Diagnostic can be loaded as follows:

b ec(,*file-server_host_net_number*)**stand/eccmem3.diag**
Return

Finally, the diagnostic can be loaded from local tape. Assuming that the tape contains a bootable image of `eccmem3.diag`, the three command lines listed below can be used to load the diagnostic.

1) If `eccmem3.diag` resides on a SCSI tape, use this command line:

```
b st()
```

2). If `eccmem3.diag` resides on an archive tape, use this command line:

```
b ar()
```

3) If `eccmem3.diag` resides on a tape master, use this command line:

```
b mt()
```

B.14. User Interface

There are five menus in the ECC Memory Diagnostic, a main menu and four sub-menus. The sub-menus handle commands for options, tests, and utilities.

The Command Line Language

The semicolon (;) acts as a *separator* between commands. For instance, let's assume that our goal is to run the default option from the main menu five times. In order to accomplish this, we can make use of the loop option (option `l`) as follows:

```
d ; l 5
```

The `d` specifies selection of the default test sequence. The `;` separates the `d` command from the `l` command. `l 5` indicates that the command line is to be executed five times. In short, the `;` must be used to separate commands. It is *not* used to separate a command from its argument(s) and it is *not* used to separate one argument from another; doing so causes a syntax error.

All commands, arguments and semicolons must be separated by at least one SPACE character. Thus, `d;l 5` is *not* the correct method to invoke the default test sequence five times; the lack of spaces causes a syntax error.

Main Menu

The main menu has nine options and provides access to the sub-menus. It also contains the default test command as well as a command to display the error log. These options are described following the Main menu example.

```
Sun-3 ECC Memory Diagnostic    Rev. x.x   mm/dd/yy

Main Menu

   Selections:

    o - options
    m - memory data tests
    c - ecc tests
    u - utilities
    d - default test
    l - loop
    e - display error log
    ? - help
    ^ - quit

  Command :
```

o Option o on the main menu is a navigational command. If selected, a sub-menu containing all of the various control options is presented. The Option Menu is discussed later.

m Option m is a navigational command. If chosen, a sub-menu containing all of the various memory data tests is presented. The Memory Menu is discussed later.

c Option c is a navigational command. If chosen, a sub-menu containing all of the various ECC tests is presented. The ECC Test Menu is discussed later.

u Option "u" is a navigational command. It brings up a sub-menu containing all of the various utilities that are useful in examining the memory board under test. The Utility Menu is discussed later.

d *D[addr] [size] [cmp] [passcount]*

Option d is the default test sequence. Selection of this option executes all of the tests, which are listed below. These tests test all of the memory boards present in the system or the memory boards that have been selected by `select mem bd to test` in the Option Menu.

- Memory Enable Register
- Address pattern
- Alternate Pattern
- Diagonal Pattern
- Unique Pattern
- Checker Pattern
- NTA Pattern

- Alignment Test
- Refresh Test
- EDC Diagnostic Read Test
- CE Forced Bit Test
- UE Forced Bit Test
- EDC Diagnostic Write Test
- Syndrome Register Test
- ECC Alternating Test
- ECC Diagonal Test
- ECC Checker Pattern
- Refresh Scrubbing Test

The **d** option uses a set of global parameters that are initialized at load time or that are changed upon request from the option menu. The default parameters are:

1. Memory boards to test = 1.
2. Data mode (number of bits to test in a word) 3 = long word mode.
3. Data compare "on".
4. Pass count = 1.

Option `d` has four arguments. `Daddr` and `size` must be set, while the other two can be left at the default values. The `Daddr` parameter is the starting address of memory and can be in the range of 0x0 to 0x7ffff0. The `size` parameter is the amount of memory to test and can be in the range of 0x10 to 0x7ffff0. `Daddr` and `size` added together should not be any greater then 7ffff0.

l *loop_count*

Entering `l` from the main menu provides opportunity to specify the number of times a specific sequence of tests is to be executed. For example, terminating a command line containing one or more user-specified tests with **l 5** executes all of the tests on that command line five times.

The `loop_count` argument is optional. Without it, the test(s) on the command line are performed once. The `loop_count` argument specifies, in decimal, the number of times that the command line sequence is to be executed. The range of legal numerical values for the `loop_count` is 1 to 2147483647 or (0x7fffffff). However, if an asterisk (*) is entered as the `loop_count` argument, the given test sequence will run forever.

e Option `e` from the main menu displays an error log. More specifically, at a maximum, the first 20 errors messages of each type recorded by the ECC Memory Diagnostic are printed on the screen. You may suspend and restart the error log if you wish. Pressing any key except `q` suspends the error log display until you press another key. If you press `q` at any time, the error log display is terminated.

h If option `h` is selected from the main menu, more detailed user instructions appear on the screen.

^ Finally, option ^ from the main menu terminates the ECC Memory Diagnostic.

Option Menu

The Option Menu has twelve options and contains global parameters for the tests that can be set and reset. The options can be set for one individual test or for all tests. The option menu is shown below.

```
Sun-3 ECC Memory Diagnostic    Rev. x.x  mm/dd/yy

Option Menu

  Selections:

    a - set default address and size
    m - set data mode (byte, word, long, quad word)
    b - select memory board to test (1,2,3,4)
    p - ECC enable/disable
    x - data compare on/off
    l - scopeloop on error
    s - stop on error
    e - error messages on/off
    ? - help
    ^ - return to main menu

  Command :
```

NOTE *When you change the arguments to the parameters inc,addr,size, from the default, the new values will become the default and will be used throughout all of the tests until changed again or until the diagnostic is rebooted.*

a *[addr] [size]*

Option a sets up the starting address and the size of memory to be tested. Most of the tests and utilities require address and size parameters. The address is the beginning address (low) of memory to be tested, while the size is the number of bytes in the block of memory to be tested. The default address and size are used if none other is supplied on the command line for the tests and utilities. The default address and size are initially set to include all of memory in the system at the time the diagnostic is loaded. This option allows you to change the default address size. The address and size are always given in hex. The address should be in the range of 0 to 0x7ffff0 and the size should be in the range of 0x10 to 0x7ffff0. When testing any memory board other then board one, the program adds the virtual address offset to both the address and size parameters to allow testing of these memory boards.

m *[number]*

Option m sets the data mode that tell the tests how many bytes to test. These are byte, word, long word, and quadword modes. There are some tests that are only executed in longword mode and will not look at this parameter. However most of them the do, the description for each test explains what data modes are excepted.

You may enter a number from 1 to 4 after `m`, where (1) selects byte mode, (2) selects word mode, (3) selects long word mode, and (4) selects quad word mode.

b *[number]*

Option `b` allows you to select any or all or a combination of memory boards to be tested with each test. Note that you must know how many memory boards are in the system under test. If you select a memory board to test that does not physically exist, the system will produce a watchdog time out and trap back to the PROM monitor. The number entered on the command line is a code and the numbers can be combined. The number must be a hex number from 1 to 0xf. The default is 1 memory board selected. Use this table to decide what value to enter:

Hex Value	*Description*
1	select membd 1
2	select membd 2
3	select membds 1, 2
4	select membd 3
5	select membds 1, 3
6	select membds 2, 3
7	select membds 1, 2, 3
8	select membd 4
9	select membd 1, 4
a	select membd 2, 4
b	select membds 1, 2, 4
c	select membd 3, 4
d	select membds 1, 3, 4
e	select membds 2, 3, 4
f	select membds 1, 2, 3, 4

p *[number]*

Option `p` enables or disables ECC checking for any given memory data test. For the ECC tests, the ECC is already enabled. Note that if ECC is enabled for the memory data tests, when the test fails you will have to use the Memory Error register and Syndrome register read utilities to determine if the error was caused by a CE or UE and to determine the bad bits and the address of the failure. This could be done with data compare *off* for faster execution of the memory tests. With the data compare *off*, the Memory Error and Syndrome registers should be read after each test to determine if an error occurred and if so, the location of the error. The default is ECC checking *off*. The *[number]* argument should be `0` to select Error checking *off* or `1` to select error check *on*.

x *[number]*

Option `x` turns data compare mode on or off. The sequence of most memory tests is write, read, compare. The compare step can be skipped using this option, thereby speeding up memory scanning. It must be noted that if the data compare option is turned off, the ECC circuitry must be enabled to catch any errors that may exist in the memory array. The *number* argument may be `0` to select data compare mode *off*, or `1` to select data

compare mode *on*. The default is data compare mode *on*.

l *[number]*
Option `l` enables or disables scopeloop on error. If an error occurs, a scopeloop is entered, continually repeating the circumstances causing the initial error. The loop can be broken by typing special keys that begin the test again or to continue with the next test. The *number* argument may be `0` to disable scope loop on error, or `1` to enable scope loop on error.

s *[number]*
Option `s` stops the execution of the present test if an error occurs. The wait can be interrupted by typing any key on the key board. The *number* argument may be `0` to disable stop on error or `1` to enable stop on error.

e *[number]*
Option `e` enables or disables the display of all messages to the screen. Only the first 20 error messages are stored and display later if this option is disabled. The default is error messages enabled. The *number* argument may be `0` to disable error message display during execution of each test, or `1` to enable error message display.

`?` Option `?` brings up the Option Menu's help display.

`^` Entering `^` returns you to the main menu.

Memory Data Menu

The Memory Data Menu has twelve options and displays the Memory data tests that are available to be executed:

```
Sun-3 ECC Memory Diagnostic   Rev. x.x  mm/dd/yy

Memory Data Test Menu

  Selections:

    a - mem enb register test
    b - address pattern
    c - alternating pattern
    d - diagonal pattern
    e - unique pattern
    f - checker pattern
    g - nta pattern
    h - alignment test
    i - refresh test
    l - loop
    ? - help
    ^ - return to main menu

  Command :
```

The "dots" between arguments are placeholders because the program reads the values in four fields and determines which value applies to which parameter depending on the placement of the value.

a . . . *[passcount]*

The option a checks the memory enable register on each of the memory boards in the system. This check turns on and off the bits that are write/read only. This test checks the ability of each of the enable registers to hold and report correct status, and insures that there are no shorts.

Upon exit from the test all the memory boards except memory board 1 are disabled. This is because memory boards 2, 3, and 4 are mapped to the same virtual and physical address. If one of these board is to be tested it will be enabled prior to the execution of the test and disable after the test has completed.

The bits that are checked are the base address bits (0 - 5), Board enable bit (6), scrub enable bit (7), Enable DM0 (11), Enable DM1 (12).

The option a has four arguments, one of which is used. The first three are place holders. The *passcount* argument determines how many times to execute the test.

To abort the test and return to the test Menu press the q key.

b *[addr] [size] [cmp] [passcnt] [dmode]*

Option b tests the specified block of memory using the low order bits of the address of each location, or its complement, as data. This test can be run in byte, word, long word, or quadword mode. Use the address pattern test to detect coupling faults.

The option b has four arguments. The *address* and *size* arguments have been discussed in the section containing the Option Menu. The *cmp* option is used with this test to invert the physical address and use this inverted address as data written into the DRAM chips. If you enter 1 the physical address in inverted; entering 0 does not invert the address. The *passcnt* argument sets the number of passes this test will execute before returning to the Memory Data Menu.

The *dmode* argument selects which data size to test. The choices are byte, word, long word, and quadword. Refer to option m in the Option Menu section. To abort the test and return to the test Menu, enter q.

c *[addr] [size] . [pascnt] [dmode]*

Option c is the alternating pattern test, which tests the specified block of memory using the alternating data pattern. First the memory block is filled with data, then it is read back and compared with the specified data pattern. If the data read from an address location does not match the original data pattern an error is flagged. This test can be run in byte, word, long word, or quad word mode. Use the alternating pattern test to detect stuck at faults.

The patterns used for testing are as follows. The test pattern alternates with each pass.

pass 1: a5a5a5a5 5a5a5a5a ....
pass 2: 5a5a5a5a a5a5a5a5 ....

The option c has four arguments. The *address* and *size* arguments have been discussed in the section containing the Option Menu. The third option is a place holder so that the remaining fields can be entered. The *passcount* argument sets the number of passes this test will execute before returning to the Memory Data Menu.

The *dmode* argument selects which data size to test. The choices are byte, word, long word, and quadword. Refer to option m in the Option Menu section. To abort the test and return to the test menu, press the q key.

d *[addr] [size] [complement] [pascnt] [dmode]*

Option d performs the diagonal pattern test. The diagonal test is actually a modified galpat test (also called diapat). This test requires a number of write-read scans thru each bank of memory to be tested. At the beginning of the test the memory is initialized to zeros. The test proceeds in the following sequence:

Pass 0 : a long word of 1's is written at particular locations in memory causing a diagonal of 1's in each memory chip's memory array. These locations can be determined by examining the address lines that decode RAS and CAS.

Example

```
pass 0:  00000000 00000001
         00000000 00000002
         00000000 00000004
         00000000 00000008
         00000000 00000010
            .
            .
         80000000 00000000
```

Pass N : the diagonal is shifted until it has occupied all of the diagonal positions of each memory chip's array with wrap around. In other words, each cell of the memory array has been the only 1 cell in a row and column of the array.

Example:

```
pass N:  80000000 00000000
         00000000 00000001
         00000000 00000002
         00000000 00000004
            .
            .
         40000000 00000000
```

Each read scan checks for the diagonal of 1's in a field of 0's, note that the entire memory bank is checked. This test is run in byte, word, long word, and quad word modes. The test can be run with inverted data using the `compl` parameter. Use the diagonal pattern test to detect pattern sensitive faults.

The option *d* has four arguments. The address and size arguments have been discussed in the section containing the Option Menu. The `complement` option is used with this test to invert the data pattern written into the DRAM chips. Enter 1 to invoke invert the data pattern and 0 if you do not want to invert the pattern. The *passcount* argument sets the number of passes this test will execute before returning to the Memory Data Menu. The *dmode* argument selects which data size to test. The choices are byte, word, long word, and quadword. Refer to option `m` in the Option Menu section.

To abort the test and return to the test menu, press the `q` key.

- *[addr] [size] [incr] [pascnt] [dmode]*

  Option `e` is for the address uniqueness test. The test for address uniqueness tests the specified block of memory using the sequence {incr, 2 * incr, 3 * incr, 4 * incr, ... } for the test data. This test can be run in byte, word, long word mode. Use the unique pattern test to detect couping faults.

  Option `e` has four arguments. The address and size arguments have been discussed in the section containing the Option Menu. The `incr` argument

determines the increment value that is added to the data pattern written into the memory. If no increment value is given the memory will be cleared. The *pascnt* argument specifies the number of times to execute the test before returning the menu. The *dmode* argument selects which data size to test. The choices are byte, word, long word, and quadword. Refer to option `m` in the Option Menu section.

To abort the test and return to the test menu press the `q` key on the keyboard.

f *[addr] [size] [pattern] [pascnt]*

Option `f` performs the Checker test. Checker test writes a sequence of pattern and ~pattern in a series of write/read scans as follows:

Pass 0 {pattern, ~pattern, ~pattern}
Pass 1 {~pattern, pattern, ~pattern}
Pass 2 {~pattern, ~pattern, pattern}

The checker test requires a number of write-read scans over the block of memory under test (which explains why it takes so long to run!). The data used is an alternating sequence of pattern and ~pattern (the complement of pattern), and hence the name checker (short for checker board). This test can be executed in byte, word, long word, or quad word mode. Use the checker pattern test to detect pattern sensitive faults.

Option `f` has four arguments. The *address* and *size* arguments have been discussed in the Option Menu section. The *pattern* argument determines which pattern and ~pattern that is written into memory. The *pascnt* argument specifies how many passes are executed. For time considerations the pass count is 1, for much better testing the suggested pass count should 3 or more passes.

To abort the test and return to the test menu press the `q` key on the keyboard.

g *[addr] [size] . [pascnt]*

Option `g` performs the NTA pattern test. This test detects stuck-at faults, coupling faults, and pattern sensitivity faults in the memory under test. The test executes the 8 passes in byte mode only, to verify memory as follows.

First each location of memory is initialized to 0. Pass 1 : Each 0 is read and changed to a 1 starting at the bottom of the memory array. The 1's are read back starting at the top of memory.

Pass 2 : Each 1 is read and changed to a 0 starting at the bottom of the memory array. The 0's are read back starting at the top of memory.

Pass 3 : Each 0 is read and changed to a 1 starting at the top of the memory array. The 1's are read back starting at the bottom of memory.

Pass 4 : Each 1 is read and changed to a 0 starting at the top of the memory array. The 0's are read back starting at the bottom of memory.

Pass 5 : Each 0 is read and changed to a 1 and back to a 0 starting at the bottom of the memory array. The 0's are read back starting at the top of memory.

Pass 6 : Each 0 is read and changed to a 1 and back to a 0 starting at the top of the memory array. The 0's are read back starting at the bottom of memory.

Next each location of memory is reset to 1.

Pass 7 : Each 1 is read and changed to a 0 and back to a 1 starting at the bottom of the memory array. The 1's are read back starting at the top of memory.

Pass 8 : Each 1 is read and changed to a 0 and back to a 1 starting at the top of the memory array. The 1's are read back starting at the bottom of memory.

Use the nta pattern test to detect stuck at faults, coupling faults, and pattern sensitive faults.

Option g has four arguments. The *address* and *size* arguments have already been discussed. The third argument is a place holder if the *pascnt* argument is to be used. The *pascnt* argument has also been discussed in the above test options.

To abort the test and return to the test menu press the q key on the keyboard.

h *[adr] . . [pascnt]*

Option h is the address alignment test. This test tests the byte, word, long word, and quad word alignment of the data in the memory. The hardware is designed so that all data is stored on quad word boundaries. Any write with in this boundaries causes a read-modify-write cycle if the new data is less then the quad word. This test is designed to test this read- modify-write function of the memory board to ensure that the data is written to the correct location, and read from it. This test will test a byte, word, long word, and quadword in all possible positions, including the writes across the quad word boundary.

The option h has two arguments. The "dots" between the two arguments are placeholders because the program reads the values in four fields and determines which value applies to which parameter depending on the placement of the value. If you are not going to enter a *pascnt* value, you do not need to enter the placeholder value. *addr* specifies the starting address to test. The *pascnt* argument has already been discussed for the previous tests.

To abort the test and return to the test menu press the q key.

i *[addr] [size] . [pascnt]*

Option i performs the refresh test. This tests tests the refresh logic on all of the memory boards in the system. A data pattern is written to a 256K block of memory. The test waits for the specified delay and then reads the block of memory, and checks the data for no decay. If the data did decay,

the refresh logic is not functioning, and an error is posted.

This test can not be run in any kind of DRAM on either the CPU (video RAM) or on the memory board due to the way RAS has been implemented. All efforts will be made to execute the test in the MC68020's internal cache. If the test is too big to fit in the MC68020's cache, the test can not be executed.

The `i` option has three arguments. The *address* and *size* and *pascnt* arguments have already been discussed. The "dot" is a placeholder that must be entered if you enter a *pascnt* value.

To abort the test and return to the test menu, press the *q* key.

`l` *[loop_count]*

Option `l` from the main menu provides the opportunity to specify the number of times a user-specified sequence of tests is to be executed. For example, terminating a command line that contains one or more user-specified tests with `l 5` executes all of the tests on that command line five times.

Option `l` accepts one command line argument. The *loop_count* argument is optional. Without it, the test(s) on the command line are performed once. The *loop_count* argument specifies, in decimal, the number of times that the command line sequence is to be executed. The range of legal numerical values for the *loop_count* is 1 to 2147483647 or (0x7fffffff). However, if an asterisk (*) is entered as the *loop_count* argument, the given test sequence will run forever.

Option

`t` *[addr] . [pattern] [pascnt]*

performs the `bcopy` test.

It tests the `bcopy` interface between the CPU board and the memory board by writing a block pattern into memory and then reading the block out into the 128 bit (16 byte buffer in the cache) then doing a block copy write to memory. It verifies that the data was copied correctly.

Option "t" has three arguments. The address argument have been discussed in the Option Menu section. The *pattern* argument determines which pattern that is written into memory. The *pascnt* argument specifies how many passes are executed.

`?` Option `?` brings up the Memory Data Test Menu's help display. Option `^` returns you to the Main Menu.

ECC Test Menu

The ECC test menu has twelve options and runs a variety of tests that exercise the EDC chip functionality, and test the ability of the ECC DRAM chips to correctly store and read correct data. The ECC Test Menu is shown below.

```
Sun-3 ECC Memory Diagnostic     Rev. 1.0        3/15/86

ECC Test Menu:

  Selections:

    j - EDC Diagnostic read mode test
    k - CE forced bit test
    m - UE forced bits test
    n - EDC Diagnostic write mode test
    o - Syndrome reg test
    p - ECC alternating pattern
    q - ECC diagonal pattern
    r - ECC checker pattern
    s - refresh scrub test
    l - loop
    ? - help
    ^ - return to main menu

Command:
```

The option h has two arguments. The "dots" between arguments are placeholders because the program reads the values in four fields and determines which value applies to which parameter depending on the placement of the value.

j . . . *[pascnt]*

Option j performs the EDC Diagnostic Read test which tests the EDC chips ability to correct errors. This is accomplished by writing a data pattern into memory that causes a syndrome code of all zero's. It then writes the diagnostic register of the ECC chip with a different check bit code. It enables Error Correction and reads the data from memory with the EDC in diagnostic read mode. This should cause an error. The test then reads the Syndrome register and checks the syndrome written there against the check bit code written into the diagnostic register. They should match. If an error is detected it will be reported. This tests all 256 combinations of check bit codes that the ECC chip will generate.

The option j has four arguments. The first three arguments are place holders if the *pascnt* argument is to be used. The *pascnt* argument has been discussed in the Memory Data Test section.

To abort the test and return to the test menu, press the q key.

k . . . *[pascnt]*

Option k performs the Correctable forced Error test, which checks the ability of the ECC logic to detect and correct single bit errors. To do this, the test uses the diagnostic read mode function to cause single bit errors of the data being read from the data memory locations. This causes the ECC logic to write the corrected bit back to the data memory. Next, the test turns error correction off and reads the modified data from the memory location. If the

data read back does not match the expected data, an error is reported. This test checks all of the 64-bit positions of the stored data. The initial data pattern has all zeros. The check bit code stored along with the data is never changed, allowing all of the bits to be corrected.

The k option has four arguments. The first three arguments are place holders if the *pascnt* argument is to be used. The *pascnt* argument has been discussed in the Memory Data Test section.

To abort the test and return to the test menu, enter q.

m . . . *[pascnt]*
Option m performs the Uncorrectable forced Error test, which checks the ability of the ECC logic to detect and not correct double bit errors. This is accomplished in much the same way as the previous test, except that the check bit code used forces double bit errors instead of single bit errors. The data is read back and checked to be sure that it was not corrected. If the data was corrected an error message will be reported.

The m option has four arguments. The first three arguments are place holders if the *pascnt* argument is to be used. The *pascnt* argument has been discussed in the Memory Data Test section.

To abort the test and return to the test menu, enter q.

n . . . *[pascnt]*
Option n performs the EDC diagnostic write test, which tests the EDC chip's ability to detect and correct errors. To do this, the test writes a data pattern into memory that causes a syndrome code of all zero's. It then writes the diagnostic register of the ECC chip with a different check bit code. Next, the test writes the new check bit code into the ECC DRAM chips, using the diagnostic mode 1. Now, it enables error correction and reads the data from memory, which should cause an error. It reads the Syndrome register and checks the syndrome written there against the expected syndrome code, and they should match. If an error is detected it is reported. All 255 combinations of check bit codes that the EDC chip generates are tested.

The n option has four arguments. The first three arguments are place holders if the *pascnt* argument is to be used. The *pascnt* argument has been discussed in the Memory Data Test section.

To abort the test and return to the test Menu, enter q.

o . . . *[pascnt]*
Option o performs the Syndrome register check test, created due to the fact that the ECC tests will not generate all address combinations for the Syndrome register. The address of each write/read operation is written into the Syndrome register as long as no error has occurred, thus allowing the register to be tested. This test rotates a bit through each of the address lines of the Syndrome register, checking for opens and "stuck at" shorts. Data is written to a quad word address, the syndrome register is read and the address compared. The address is then shifted by 1 and the loop continues until all 22 address lines been checked. However, if an error is found, the Syndrome

register is displayed. This test is executed once per board when testing a 32 megabyte board.

The o has four arguments. The first three arguments are place holders if the *pascnt* argument is to be used. The *pascnt* argument has been discussed in the Memory Data Test section.

To abort the test and return to the test Menu, enter q.

p *[adr] [size] . [pascnt]*

Option p performs the ECC alternating pattern test. In the alternating pattern test the specified block of ECC memory is tested, using a data pattern that causes an alternating data pattern to be written into the ECC DRAM chips. First, the memory block is filled with data, error correction is enabled, then it is read back and the Syndrome register is checked for errors. If an error occurs, the syndrome and address are reported.

The data pattern that is written into the ECC DRAM chips follow. After each pass the data that is written into the ECC DRAM chips is inverted.

Example:

pass 1: a5 5a ...

pass 2: 5a a5 ...

Use the alternating pattern test to detect stuck at faults in the ECC DRAM chips.

The option p has four arguments. The first two determine the starting address and size of memory to test, and are described in the first section of *Option Menu*. The third argument is place holder. The last argument, *pascnt* has been discussed in the section *Memory Data Tests*.

To abort the test and return to the test Menu, enter q.

q *[adr] [size] [comp] [pascnt]*

Option q performs the ECC Diagonal pattern test. In this test, the specified block of ECC memory is tested, using a data pattern that will put a diagonal pattern in the ECC DRAM chips. First, the block of memory is written with the data pattern, error correction is enabled, and the data is read from memory. The Syndrome register is checked for errors. If an error occurs, the syndrome and address of the error are reported. This test can be executed with a data pattern that inverts the data in ECC DRAM chips.

The test pattern in the ECC DRAM chips is as follows:

pass 1: 01 02 04 08 10 20 40 80

pass 2: 02 04 08 10 20 40 80 01

The option q has four arguments. The address and size arguments have already been discussed in the *Option Menu* section. The third argument, *comp*, is a flag that, when set, complements the data that is written into memory. Enter 1 in place of *comp* if you want to set invert the data; enter 0 if you do not. The forth argument, *pascnt* also has been discussed in the section titled *Memory Data Test.* If you wish to stop the test while it is running, enter q.

r *[adr] [size] . [pascnt]*

Option r performs the ECC checker pattern test, which tests the specified block of ECC DRAM chips that writes the checker pattern into them. The data pattern is written to memory, error correction is enable, and the data is read from memory. The Syndrome register is checked for errors. If an error occurred, the syndrome and address of the error is reported.

The data written into the ECC DRAM chips is as follows:

pass 1: 0x00 0xff 0xff

pass 2: 0xff 0x00 0xff

pass 3: 0xff 0xff 0x00

The option r has four arguments. The address and size arguments have already been discussed in the *Option Menu* section. The third argument is a place holder for the *pascnt* argument. The forth argument, *pascnt*, also has been discussed and is found in the section titled *Memory Data Test.* For a better test of the ECC DRAM, set the number of passes to 3 or more.

To abort the test and return to the test menu, enter the q.

s *[adr] [size] . [pascnt]*

Option s performs the Refresh scrubbing test, which tests memory cell refresh scrubbing. The test first initializes memory with ECC on, then enables the error interrupt and scrubbing bit. It waits 20 seconds, and if it times out without an interrupt and the Syndrome register has no error in it, the scrubbing is good.

Next, the test disables interrupts and scrubbing, writes a bad ECC code to a memory location and then enables the interrupts and scrubbing. After a 20 second wait, if an interrupt occurred, and the syndrome register code is what is expected, the scrubbing did find the error. The test now checks the data location to see if it was corrected. If no interrupt occurred, an error is reported.

This test is executed once per board when testing a 32 Megabyte board. The time delay has been increased slightly to allow for scrubbing of 32 Megabyte memory boards.

The r option has four arguments. The *address* and *size* arguments have already been discussed in the *Option Menu* section. The third argument is a place holder for the *pascnt* argument. The forth argument, *pascnt*, also has

been discussed and is found in the section titled *Memory Data Test.*

To abort the test and return to the test menu, enter `q`.

`l` *[loop_count]*

Entering `l` from the main menu provides opportunity to specify the number of times a specific sequence of tests is to be executed. For example, terminating a command line containing one or more user-specified tests with **`l 5`** executes all of the tests on that command line five times.

Option `l` accepts one command line argument. The *loop_count* argument is optional. Without it the test(s) on the command line are performed once. The *loop_count* argument specifies, in decimal, the number of times that the command line sequence is to be executed. The range of legal numerical values for the *loop_count* is 1 to 2147483647 or (0x7fffffff). However if an asterisk (*) is entered as the *loop_count* argument, the given test sequence will run forever.

`?` Option `?` displays the Memory Data Test Menu's help display.

`^` Option `^` returns the you back to the Main Menu.

Utility Menu

The Utility Menu has six options and shows utility functions that will aid in determining the functionality of the memory board(s) in the system. The tools are: fill memory, display a section of memory, read the CPU Memory Error Register, and Read the memory board Memory Enable Register and Syndrome register. The Utility Menu is shown below.

```
Sun-3 ECC Memory Diagnostic   Rev. x.x   mm/dd/yy

Utility Menu

  Selections:

    f - fill
    d - display
    e - memory error register (cpu)
    s - read syndrome register
    ? - help
    ^ - return to main menu

  Command :
```

`f` *[adr] [size] [pattern]*

Option `f` is the Fill Memory utility, which allows you to fill a specified block of memory with a specified pattern. Memory can be filled with bytes, words, long words of the given data pattern depending on the data mode set.

The option `f` has three arguments. The *address* and *size* arguments have been discussed in the section, *Option Menu.* The *pattern* argument is the data pattern that is to be written in DRAM.

`d` *adr size*

Option `d` displays a specified section of memory on the console or terminal.

Data is always (no matter what the data mode) read and displayed a byte at a time. Each line of the display contains the hexadecimal address (always a multiple of 0x10 except for the first line if the specified block doesn't begin on a multiple of 0x10 boundary) followed by 16 bytes of data grouped in long words (by 4s).

In order to display memory on any board other than board "0", you should first go to the Option Menu and, using the m option, select the board from which you want to display the memory.

The option d has two arguments. The *adr* argument is the starting address of the section of memory to be displayed. This address should be between 0 and 0x7ffff0. The *size* argument is length of data to be displayed and has the same range as the address argument.

The following is an example of the display.

```
100000:  00 01 02 03  04 05 06 07  08 09 0a 0b  0c 0d 0e 0f
100010:  10 11 12 13  14 15 16 17  18 19 1a 1b  1c 1d 1e 1f
```

- **e** Option e displays the Memory Error Register on the CPU board. The data that is displayed CE and UE. This is useful when the compare routine is disabled and ECC is turned on to speed up execution of the tests.

  When the Memory Error register is displayed, it is displayed as a hex value.

- **s** Option s displays the ECC Memory Enable and Syndrome registers when executed. It sets the last address of the syndrome code if no error occurred (or, in case of error, the address of the error), and it sets the CE bit. The syndrome code is either the last syndrome latched before the register was enabled or the first error occurrence after the register was enabled.

  The ECC Enable Register and Syndrome register are displayed as hex digits. Following is an example of the display.

  ```
  ECC Enable register = 0x0240ffff
  Syndrome register, 0xc001200
  ```

  Where the contents of the Enable register = board size 8 megabytes, board enabled.

  The contents of the Syndrome register = syndrome code 0xc, address 0x2400, CE bit is zero.

- **?** Option ? displays the Utility help menu.

B.15. Error Handling

ECC Errors

ECC errors are handled separately from bus errors. ECC is now a non-maskable interrupt at level 7, rather than a bus error as before in the older architecture. The ECC handling in the hardware has been vastly improved in the current architecture.

The hardware latches ECC errors synchronously as they occur (no cycle delays) and the processor is immediately notified. In addition, the virtual address containing the ECC error is latched in hardware as well as the syndrome code in which bit of the contents of the latched address is in error.

This error occurs only when ECC is enabled during memory data tests:

```
*** ECC error! : memory error reg= 0x<memory_error_reg>
                 syndrome reg= 0x<syndrome reg>
```

The first line flags the error as an ECC error and displays the contents of the memory error register. The next line displays the syndrome register's contents. The last line displays the information saved on the stack by the MC68020 as it processes exceptions (an interrupt in this case).

ECC Test Error Messages

The following messages are displayed when an error has occurred during the ECC tests.

ECC Data Compare Error

The following message may be displayed during execution of the ECC Alternate, ECC Diagonal, and ECC Checker tests. The failing address is displayed along with the address that was read from the syndrome register (these should match), and the contents of the syndrome code. To determine what the code means use the table at the end of this section.

This message is displayed when, during a read from a memory cycle where the syndrome register code is non-zero (error condition).

test name `failed @` *addr>*
`syndrome addr (`*saddr*`) syndrome (`*syndrome*`)`

EDC Forced Error

The following message may be displayed during execution of the EDC Diag Read, CE forced, UE forced, and EDC Diag Write tests. The failing address is displayed along with the expected syndrome code, the read syndrome code.

test name `failed @` *addr*
`exp (`*wrdata*`) obs (`*rddata*`) xor (`*Wrdata ^ rddata*`)`

The message shown above occurs when the expected, forced syndrome code does not equal the syndrome code read form the syndrome register.

Refresh Scrub Errors

These errors occur during the Refresh Scrubbing test. The first one, `No Interrupt` means that an error condition was forced with the (level 7) interrupt turned on and the interrupt did not happen. The second one, `No CE Mem Cntrl Reg`, occurs during the same forced error condition where the CE bit in the Memory Error Control register was not set. The third error, `No CE Syndrome reg`, occurs when the CE bit is not present in the syndrome register during the same forced error. The error messages look something like this:

test name No Interrupt failed @ *addr*
exp*expdata* obs *rddata* xor *expdata* ^ *rddata*.

test name No CE Mem Cntrl reg failed @ *addr*
exp*expdata* obs *rddata* xor *expdata* ^ *rddata*.

test name No CE in Syndrome reg failed @ *addr*
exp*expdata* obs *rddata* xor *expdata* ^ *rddata*.

Bus Errors

Bus errors are trapped and a message is displayed as follows:

*** bus error! : 0x*bus_error_reg*
sr=*statusreg* pc=*progctr* vss=*vectoff&specstat* @*addr*

The first line flags the error as a bus error and displays the contents of the bus error register. The second line displays the information saved on the stack by the MC68020 as it processes exceptions (a bus error in this case).

Data Compare Errors

This error message is displayed for all of the Memory Data Tests. It is displayed when an compare error is found. That is, during a read compare operation the data read does not equal the data written. The address of the failure is displayed along the data written and the data observed. The message is displayed as shown below.

test name failed @ *addr*
exp (*wrdata*) obs (*rddata*) xor (*wrdata* ^ *rddata*)

B.16. Special Problems

At the time of this writing, when exiting to the PROM monitor from this diagnostic, using a `k1` command causes problems. It is recommended that, in order to boot the operating system or any other standalone diagnostic, you first execute a `k2` reset instead of a `k1` reset.

B.17. Replacing the Memory Board

If the ECC Memory Diagnostic is being executed in the field you determine that the memory board under test should be replaced, look at the first message after the menu display, `Testing Memory Board` *X*, where *X* is a number from 1 to 4. The jumper on the edge of the board determines what the board number is. The following table shows the jumper positions and what they mean.

Memory Board	Jumper Position
	o o
1	o o
	o o
	0--0
	o o
2	o o
	0--0
	o o
	o o
3	0--0
	o o
	o o
	0--0
4	o o
	o o
	o o

B.18. Recommended Test Procedure

The recommended test procedure for minimal testing of the Sun-3/2xx memory boards is that you execute the following tests:

Checker Pattern Test - Memory Data Test Menu

EDC Diag Read Test - ECC Test Menu

ECC Checker Pattern Test - ECC Test Menu

These tests will give a brief confidence level of the memory data RAM (DRAM) chips and the addressing to the chips. They test the functionality of the Error Detection and Correction chips, and finally, the ECC DRAM chip's ability to store and read data from them. The tests listed above can be executed on all boards if they are in the system and if the board number has been selected (see Option Menu).

If you want to exhaustively test all the memory boards in the system, execute the default test from the Main Menu. This test has been described in the *Main Menu Tests* section. This option is useful for burn-in of the memory boards and exhaustive field testing.

B.19. Glossary

AMD
Advanced Micro Devices

Bootpath
Interface and bus logic from CPU to an I/O boot device

Cache
An associative, fast RAM between the CPU and main memory

CE
Correctable error

CPU
Central Processing Unit

DM0
Diagnostic mode 1 for EDC chip (write function)

DM1
Diagnostic mode 2 for EDC chip (read function)

DRAM
Dynamic Random Access Memory

EDC
AMD's 16 bit Error Detection and Correction unit

ECC
Error Checking and Correction, on main memory

I/O
Input and output, as for example, an input/output device

UE
Uncorrectable Error

RAM
Random Access Memory

Refresh scrub
Correction of single bit errors that are found during a refresh cycle

Video RAM
Memory that holds the video information that is display on the screen

B.20. Syndrome Decode Table

The following table defines which bit is in error or if the error was caused by a double bit error (UE) or multi-bit error. The table is read left to right. For example if the syndrome code in the syndrome register was CE, read down the first column until you find C0 then go across until you find 0E. This tells us that bit 0 is bad.

Syndrome Bits 7-4	*Syndrome Bits 3 - 0*															
	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	*	cx	c0	t	c1	t	t	m	c2	t	t	17	t	m	16	t
10	c4	t	t	18	t	19	20	t	t	21	22	t	23	t	t	m
20	cs	t	t	08	t	9	10	t	t	11	12	t	13	t	t	m
30	t	14	m	t	15	t	t	m	m	t	t	m	t	m	m	t
40	c16	t	t	m	t	m	m	t	t	m	33	t	m	t	t	32
50	t	m	34	t	35	t	t	36	37	t	t	38	t	39	m	t
60	t	m	56	t	57	t	t	58	59	t	t	60	t	61	m	t
70	62	t	t	m	t	63	m	t	t	m	m	t	m	t	t	m
80	c32	t	t	m	t	m	m	t	t	m	49	t	m	t	t	48
90	t	m	50	t	51	t	t	52	53	t	t	54	t	55	m	t
a0	t	m	40	t	41	t	t	42	43	t	t	44	t	45	m	t
b0	46	t	t	m	t	47	m	t	t	m	m	t	m	t	t	m
c0	t	m	m	t	m	t	t	m	m	t	t	1	t	m	0	t
d0	m	t	t	2	t	3	4	t	t	5	6	t	7	t	t	m
e0	m	t	t	24	t	25	26	t	t	27	28	t	29	t	t	m
f0	t	30	m	t	31	t	t	m	m	t	t	m	t	m	m	t

How to decode this table

* denotes no error detected

A number indicates bit number of the single bit in error

A t means that a two-bit error was detected

An m means that more than two errors were detected

Revision History

Dash Number	Date	Comments
10	November 16, 1987	FCS for 1.0 Release (Internal Only)
11	December 21, 1987	First Draft for Release 1.1
12	December 29, 1987	Beta Draft for Release 1.1
13	May 20, 1988	FCS for Release 1.1