sun
microsystems

SunDiagnostic Executive User's Guide
For MC68020, MC68030, and SF9010IU (SPARC) Based Sytems

Part Number: 800-3273-10
Revision: A of 30 June 1989

SunDiagnostic Executive User's Guide For MC68020, MC68030, and SF9010IU (SPARC) Based Systems

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

Part No: 800-3273-10
Revision A of 30 June 1989

The Sun Graphical User Interface was developed by Sun Microsystems Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees.

Contents

Tables

Figures

1

Introduction

1

Introduction

This manual describes the programs on the SunDiagnostic 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. Audience

This manual is intended for use by personnel in field service, repair depots, manufacturing, and "self-maintaining" customer sites. We assume that the reader is familiar with Sun systems and with diagnostic practices.

1.2. Common Terms

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, `cpu4.exec` tests the CPU board, and `mem4.exec` tests the memory.

Executive Tape — A 1/4" or 1/2" 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 — describes the SunDiagnostic Executive, an operating system that provides a platform for Sun diagnostics and runs independent of the SunOS™.

1.3. Conventions

Fonts

In this manual, different fonts are used to make things clearer. The most common fonts are Roman, `Courier`, **`Courier bold`**, *Italic*, and **Roman 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.

Courier
: `Courier 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.

Courier bold

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

1.4. References

See the following documents for further information:

- The Field Service Manuals for your system for a CPU board overview and hardware assembly procedures.
- The Hardware Installation, CPU board configuration, and slot assignment documents for your system, which provide information on cabling as well as board and system configuration.
- The System Administration Manual for your version of Sun software, which describes various system operations, including the use of the disk formatting facility, `format`.
- The *PROM User's Manual*, and subsequent addenda in the SunOS Release Notes, which contain additional information on EEPROM programming.

1.5. The Exec Tape

This manual contains a chapter describing every diagnostic on the tape version released as of the date of this writing. This manual references the 1/4-inch or 1/2-inch SunDiagnostic Executive tapes, Part Numbers 700-2180 and 700-2181 or 700-2182 and 700-2183. One set of tapes supports Sun-3 systems, while the other supports Sun-4 (SPARC™) systems. The diagnostic programs may either be booted directly off this tape, or copied onto a disk and booted from it. Chapter 2 provides a table of contents for each tape set, as well as a matrix that shows which tape files are intended to support which Sun architectures.

1.6. Organization of This Manual

The first two chapters cover the Exec in general and the operating system, or underlying "Executive" platform of this diagnostic. The names of subsequent chapters reflect the hardware that each diagnostic tests. Diagnostics that test similar hardware are grouped together, so that the chapters describing graphics accelerator boards, for example, would be found in the "Graphics Diagnostics" section. Each chapter includes menu and test descriptions. The appendices cover standalone diagnostics that are included on the tape, pin designations for loopback connectors, and a "bug report" form.

1.7. Required Equipment

The SunDiagnostic Executive requires this functional hardware:

- Memory — 1Mbyte minimum
- Sun-3 or Sun-4 CPU (MC68020, MC68030 or SF9010IU)
- The MMU
- Real-time clock
- The system must have at least one of the following:

  Console and Keyboard
  A terminal connected to Serial Port A
  A modem connected to Serial Port B

- A boot path to a storage device:

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

- Serial Port Loopback connectors, required for CPU board diagnostics.
- Additional equipment requirements are listed where necessary. Refer to *Appendix B* for CPU board loopback connector pinouts.

1.8. Software Requirements

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

- The SunDiagnostic Executive
- A set of Diagnostic Programs (in the `diags` file) 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, although there are four such programs included on the tape that can be individually booted or copied from tape.*

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

Configuring a Terminal

You may execute the Exec from a "dumb" 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

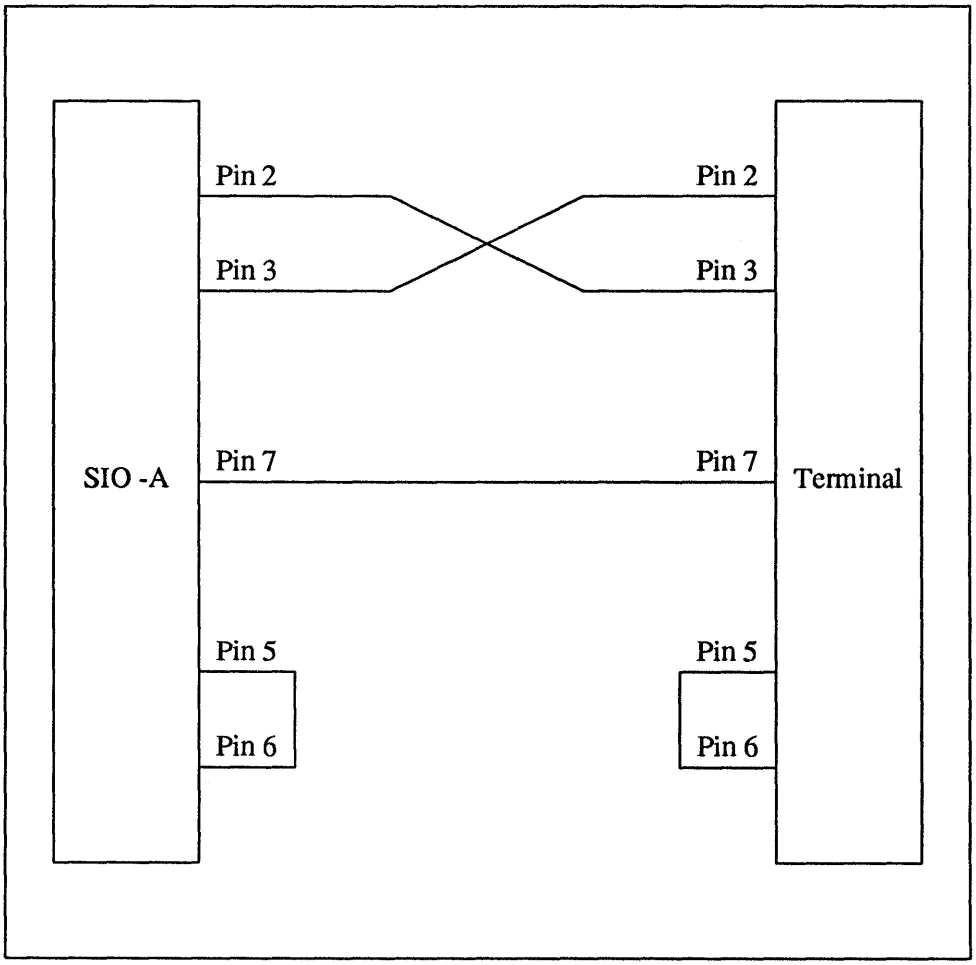
Connect the terminal to the connector labeled Serial Port A at the rear of the system.

The cable you use should be wired as follows:

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 1-1 *RS-232 Connections*



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

2

Using the SunDiagnostic Executive

2

Using the SunDiagnostic Executive

2.1. What This Chapter Contains

This chapter contains an overview of the SunDiagnostic Executive, a description of the release tape contents, loading and booting instructions, and a description of the Exec menu selections and their required command syntax. Subsequent chapters discuss the individual diagnostics that run under the Exec.

2.2. History

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

This arrangement had two significant limitations. First, only one diagnostic program could run at a time. This made it hard to load down the system for a thorough test, and 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.

2.3. Overview

The SunDiagnostic Executive provides a single, unified diagnostic environment with only one user interface. The diagnostic tests still reside in individual programs within `/stand`, but they all work through the SunDiagnostic Executive (hereafter called the Exec) platform. The platform provides a consistent interface and multitasking capabilities. In addition, the Exec provides the functionality listed below, which is available to any diagnostic. Some Exec features are:

- Multiple consoles: You can configure the Exec to run from any control terminal. This device may be the Sun monitor, Serial Port A, Serial Port B, or a remote console. Multiple terminals can also be connected to a system, and each used for input and output.
- Error Logging: The Exec provides the ability to capture error messages into a logfile. This logfile may reside either on a local disk or on the server.
- Script Execution: You may write command scripts to automatically drive the menus without further user input.
- Remote Execution: One of the special features of the Exec is the ability to perform remote diagnostics over a modem connected to a Serial Port, or by way of Ethernet. Any diagnostic that runs under the Exec execution platform can be used remotely.

The Exec has one Main Menu that appears when the you boot. Commands in this main menu call up the `Environment Menu`, `Options Menu`, `Diagnostics Menu`, `Status Menu`, and a `Log Menu`. Each diagnostic has its own menu, called up from the `Diagnostics` menu.

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

2.4. The Exec Tape

The Exec release tape contains boot programs, the Exec program itself and the diagnostics that currently run under the Exec. The following pages show Sun-3 and Sun-4 (SPARC™) tape contents, along with a table that shows which tape files support which Sun architectures. Two "standalone" diagnostics, `eccmem.diag, cache.diag`, are included on the tapes for each architecture. They are designed to test Sun-3/200 and Sun-4/200 series workstation memory. These standalone tests do not function as part of the SunDiagnostic Executive. They may be extracted from tape and executed separately. Documentation for these tests is in *Appendix A*.

NOTE *The user interface for the standalone tests is not the same as that described here for Diagnostics Menu choices.*

The table on the following page lists the contents of the 1.2 SunDiagnostic Executive Sun-3 tape, for MC68020 and MC68030-based workstations. The part number for the 1/4-inch tape is 700-2180; the part number for the 1/2-inch tape is 700-2181.

Table 2-1 *MC68020, MC68030 Tape Contents—see next page*

Name	*File Number (decimal)*	*Sun-3 Tape Contents Description*
tpboot020	0	enables Sun-3 boot from tape
tpboot030	1	enables Sun-3/400 Series, Sun-3/80 boot from tape
toc	2	Contains list of contents of this tape
extract_exec	3	Script to copy diagnostics to disk
Copyright	4	Textfile containing copyright notice
exec	5	The SunDiagnostic Executive for Sun-3 architecture
exec3x	6	The SunDiagnostic Executive for Sun-3/400 Series and 3/80
diags	7	Diagnostics Menu and File names (used by Exec)
cg6.exec	8	P4 Low-End Graphics Accelerator
cg8.exec	9	P4 24-Bit Frame Buffer Board Diagnostic
cg9.exec	10	VME 24-Bit Frame Buffer Board Diagnostic
color.exec	11	Generic VME Color Board Diagnostic
cpcache.exec	12	Sun-3/400 Series Cache Diagnostic
cpu.exec	13	Sun CPU Diagnostic
eeptool.exec	14	EEPROM programming tool
ether.exec	15	Sun Ethernet Diagnostic
ether2.exec	16	Ethernet II Diagnostic
exectest.exec	17	Exec Verification Suite
fdc.exec	18	On-Board Floppy Disk Controller Diagnostic
fddi.exec	19	FDDI Board Diagnostic
fpa.exec	20	Sun-3 Floating Point Accelerator Diagnostic
fpa_plus.exec	21	Sun-3/80 Floating Point Accelerator Plus Diagnostic
gp1.exec	22	Graphics Processor1/Graphics Buffer Diagnostic
gp2.exec	23	Graphics Processor2 Diagnostic
hss.exec	24	High Speed Serial Interface Diagnostic
iocache.exec	25	Sun-3/400 Series I/O Cache Diagnostic
ipi.exec	26	IPI Disk Subsystem Diagnostic
kb.exec	27	Sun Keyboard Diagnostic
mcp.exec	28	Sun ALM2/MCP Board Diagnostic
mem.exec	29	Sun Memory Diagnostic
mempar.exec	30	Sun-3/80 Parity Memory Diagnostic
mouse.exec	31	Sun Mouse Diagnostic
mti.exec	32	Sun MTI/ALM Board Diagnostic
scsisub.exec	33	SCSI Subsystem Diagnostic
espscsi.exec	34	ESP SCSI diagnostic
smd.exec	35	Sun SMD Diagnostic
taac.exec	36	TAAC-1 Accelerator Diagnostic
tape.exec	37	Pertec 1/2-inch Tape Diagnostic
video.exec	38	Video Circuitry Diagnostic
vidmon.exec	39	Sun Video Monitor Diagnostic
vme3.exec	40	Sun VME Diagnostic
netcon	41	Network Console Program
logfile	42	Error Log File
eccmem3.diag	43	Standalone ECC Memory Diagnostic
cache3.diag	44	Standalone Cache Memory Diagnostic

The table below lists the contents of the 1.2 SunDiagnostic Executive Sun-4 tape, for SPARC-based workstations. The part number for the 1/4-inch tape is 700-2182; the part number for the 1/2-inch tape is 700-2183.

Table 2-2 *SPARC Tape Contents*

Name	*File Number (decimal)*	*Decription-Comments*
tpboot.sun4	0	enables boot from tape
Copyright	1	Textfile containing copyright notice
toc	2	Contains list of contents of tape
extract_exec	3	Script to copy diagnostics to disk
Copyright	4	Textfile containing copyright notice
exec4	5	The SunDiagnostic Executive for SPARC architecture
diags4	6	Diagnostic Menu and File names (used by Exec)
cg6.4.exec	7	P4 Low-End Graphics Accelerator
cg8.4.exec	8	P4 24-Bit Frame Buffer Board Diagnostic
cg9.4.exec	9	VME 24-Bit Frame Buffer Board Diagnostic
color4.exec	10	Generic VME Color Board Diagnostic
cpcache4.exec	11	CPU Cache for SPARCsystem330
cpu4.exec	12	Sun CPU Board Diagnostic
eeptool4.exec	13	EEPROM programming tool
ether4.exec	14	Sun Ethernet Diagnostic
ether2.4.exec	15	Ethernet II Board Diagnostic
exectest4.exec	16	Exec Verification Suite
fddi4.exec	17	FDDI Board Diagnostic
fpu4.exec	18	Sun-4 Floating Point Unit Diagnostic
gp1.4.exec	19	Graphics Processor1/Graphics Buffer Diagnostic
gp2.4.exec	20	Graphics Processor2 Diagnostic
hss4.exec	21	High Speed Serial Interface Diagnostic
ipi4.exec	22	IPI Disk Subsystem Diagnostic
kb4.exec	23	Sun Keyboard Diagnostic
mcp4.exec	24	Sun ALM2/MCP Board Diagnostic
mem4.exec	25	Sun Memory Diagnostic
mouse4.exec	26	Sun Mouse Diagnostic
mti4.exec	27	Sun MTI/ALM Board Diagnostic
scsisub4.exec	28	Sun SCSI Subsystem Diagnostic
espscsi4.exec	29	SPARCsystem330™ ESP SCSI Diagnostic
smd4.exec	30	Sun SMD Diagnostic
tape4.exec	31	Pertec 1/2-inch Tape Diagnostic
video4.exec	32	Sun Video Circuit Diagnostic
vidmon4.exec	33	Sun Video Monitor Diagnostic
vme4.exec	34	Sun VME Diagnostic
netcon4	35	Network Console Program
logfile	36	Error Log File
eccmem4.diag	37	Standalone ECC Memory Diagnostic
cache4.diag	38	Standalone Cache Memory Diagnostic

The table below shows all the files found on both the Sun-3 and Sun-4 tapes, and indicates with an "x" which architecture is supported by that file.

Table 2-3 *Tape Contents/System Architecture Matrix*

File	***System Architecture Supported***				
	Sun-3	***Sun-3E***	***Sun-3/80 or 3/400***	***Sun-4***	***SPARCsystem 330***
exec	x	x			
exec3x			x		
exec4				x	x
cg6	3/60 only		3/400	x	x
cg8	x		x	x	x
cg9	x		x	x	x
color	x		x	x	x
cpcache			3/400 only	x	x
cpu	x	x	x	x	x
eeptool	x		x	x	x
ether	x		x	x	x
ether2	x		x	x	x
fdc			x		
fddi	x		x	x	x
fpa	x		x		
fpa_plus			x		
fpu				x	x
gp1	x		x	x	x
gp2	x		x	x	x
hss	x		x	x	x
iocache			3/400 only		
ipi	x		x	x	x
kb	x	x	x	x	x
mcp	x		x	x	x
mem	x	x	x	x	x
mempar			3/80 only		
mouse	x	x	x	x	x
mti	x		x	x	x
scsisub	x		x	x	
espscsi			3/80 only		x
espscsi4					x
smd	x		x	x	x

Table 2-4 *Tape Contents/System Architecture Matrix, Continued*

File	*System Architecture Supported*				
	Sun-3	*Sun-3E*	*Sun-3/80 or 3/400*	*Sun-4*	*SPARCsystem 330*
taac	x		x	x	x
tape	x		x	x	x
video	x	x	x	x	x
vidmon	x	x	x	x	x
vme	x		x	x	
eccmem3.diag	x				
cache3.diag	x				
eccmem4.diag				x	
cache4.diag				x	

The files on the tape are stored in `tar` format, on either a 1/4-inch, QIC24 format tape cartridge, or a 1/2-inch, 6250 BPI Storage Module Device tape.

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

2.5. Loading and Booting the Exec

Unless you boot from tape, the Exec loads its programs from SunOS 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 an its diagnostic programs are all in `/stand`.

Before Booting

The Exec cannot function unless the system has a minimum amount of functional hardware. The Exec depends on the system self-tests built into the boot PROM s to ensure minimum functionality. If you power-up the workstation in DIAG mode and the PROM selftest prints out error messages on the attached terminal, the system 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 SunOS `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 halting the operating 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. (The Break key on a terminal.) 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 load or boot the Exec. The Exec can be booted directly from tape or installed onto a local or remote disk. Be forewarned, loading from tape may take a considerable amount of time.

NOTE *The Exec relies explicitly on the value in location 0x1f of the CPU board EEPROM to specify the I/O device. An invalid value in this location will force I/O to Serial Port A. The setting of the system diagnostic switch has no affect on this interpretation. The Exec is set up this way because system hardware is assumed to be questionable, and the serial port requires a minimum of functional CPU board circuitry. Refer to the PROM User's Manual for information on using the PROM monitor* **q** *command to change EEPROM programming. The Video Monitor Diagnostic chapter of this manual also contains information on the EEPROM setting in location 0x1f, and the EEPROM Editing Tool chapter explains how to use the Exec tool to change EEPROM parameters. Valid values are:*

00	on-board frame buffer
10	Serial Port A (ttya)
11	Serial Port B (ttyb)
12	VME
20	P4 frame buffer

Installing the Exec Onto Disk

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

Servers vs Local Disk

There are two places where the Exec may be installed: 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. The best place is dependent on your particular configuration and which diagnostics you wish to run.

The local disk should be used for any standalone system that is not connected to a network, or when the complete Ethernet Diagnostic is to be executed. The guidelines that follow are command examples; consult your System Administration manual for more information.

Server

Installing a server with the Exec is a little more involved than installing on a local disk (see *Local Disk* on the next page). The booting procedure for the Exec has been targeted for Version 4.x of the SunOS operating system. This OS discourages the use of nd. However, you can place the Exec on a server that is running a 3.x version of the operating system by understanding how nd is handled by the server.

On a SunOS 3.x server, the bootpath is determined by the system architecture:

/pub.MC68020	*for the Sun-3 series*
/pub.MC68030	*for the Sun-3/400 series*
/pub.SPARC	*for the Sun-4 and SPARCsystem 300 series*

For Sun-4 (SPARC) systems, when you execute the following command from the monitor,

```
b ie()stand/exec4
```

the boot program searches for the file in /pub.SPARC/stand on the server.

Therefore, the Exec and all its diagnostics should be placed into the /pub.SPARC/stand directory.

However, on a server that is running SunOS Version 4.x, the Exec must be placed in the /export/root partition. We recommend that you place all the Exec files in /export/root/stand, hard-linked to each of the client root directories. This process eliminates the need to place the Exec in each individual client root directory. To create a hard link:

```
%su   enter password
#cd /export/root
#mkdir stand
#cd stand
#mt -f /dev/nrdevice0 rew
#mt -f /dev/nrdevice0 fsf 3
#tar xvf /dev/nrdevice0
#extract_exec device0
#cd /export/root/client/
#ln -f /export/root/stand stand
```

You must then create a symbolic link:

```
cd /
ln -s /export/root/stand stand
```

When that is complete, you can boot with

```
>b ie()/stand/exec4
```

The / before `stand` notifies the Exec that the server is running SunOS 4.x and not to prepend `pub.SPARC` to the bootpath.

Finally, you must disable the security restrictions from the server to permit the Exec to download the files it needs. To do this, you edit the `/etc/inetd.conf` file.

NOTE *To complete this set-up, you must first kill the* `inetd` *daemon, and then, after editing the file, restart the daemon.*

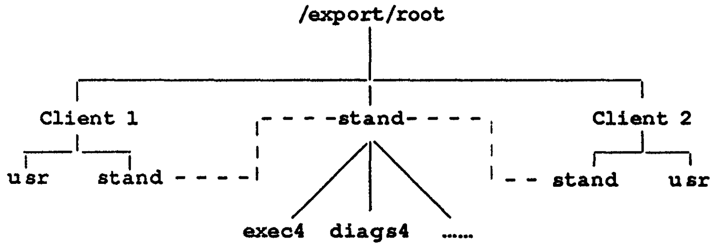
A typical file looks like this:

```
# @(#)inetd.conf 1.16 87/11/13 SMI
ftp     stream  tcp     nowait  root    /usr/etc/in.ftpd        in.ftpd
telnet  stream  tcp     nowait  root    /usr/etc/in.telnetd     in.telnetd
shell   stream  tcp     nowait  root    /usr/etc/in.rshd        in.rshd
login   stream  tcp     nowait  root    /usr/etc/in.rlogind     in.rlogind
exec    stream  tcp     nowait  root    /usr/etc/in.rexecd      in.rexecd
finger  stream  tcp     nowait  root    /usr/etc/in.fingerd     in.fingerd
tftp    dgram   udp     wait    root    /usr/etc/in.tftpd       in.tftpd -s /tftpboot
comsat  dgram   udp     wait    root    /usr/etc/in.comsat      in.comsat
talk    dgram   udp     wait    root    /usr/etc/in.talkd       in.talkd
name    dgram   udp     wait    root    /usr/etc/in.tnamed      in.tnamed
```

You must edit the line that contains `tftpd`, removing the `-s` option, so that it looks like this:

```
tftp    dgram   udp     wait    root    /usr/etc/in.tftpd    in.tftpd /tftpboot
```

Here is a diagram of the relationship you have just created. The dotted lines represent the hard links between `/export/root/stand` and the client `stand` directory.



Local Disk

To install the Exec on a local disk, select or create a target directory and perform the installing step described below.

You must first 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 **st8** for **mt8** in the examples below. Do the following:

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

example# cd /stand
example# mt -f /dev/nrmt8 rewind
example# mt -f /dev/nrmt8 fsf 3
example# tar xvf /dev/nrmt8
```

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

```
example# extract_exec mt8
you will see a number of messages here..
example# exit
example%
```

All of the files on the tape should now be copied onto the target directory.

Other Partitions

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, on a system running SunOS 3.x, the Exec may be installed in a larger partition. On a system running SunOS 3.x, we suggest that you install the Exec in `/usr/stand`, because `/usr` is usually a large partition. To do this, follow the installation procedure above, but use the directory `/usr/stand` instead of `/stand`.

If the system is also the server for the network and already has the Exec files installed in `/pub.SPARC` (SunOS 3.x) or `/export/stand` (SunOS 4.x), it would be logical to boot directly from these partitions.

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).

You may use this sequence to perform the remote tape installation. Make sure that you have a user ID on both the remote tape server and the target system.

```
%su
Password: enter super-user root passwd
#rsh remote_host mt -f /dev/nrdevice8 rew
#rsh remote_host mt -f /dev/nrdevice8 fsf file_number
#rsh remote_host -n dd if=/dev/nrdevice8 bs=blocksizeb | tar xvfpB -
```

For *device*,use `st` for SCSI tape or `mt` for 1/2-inch tape, and replace *remote_host* with the name of the system that has the tape drive from which you want to load the SunDiagnostic Executive.

Installing a Boot Block

Procedures for installing a boot block differ, depending on which version of the SunOS you are running.

For SunOS 3.x:

If you want to boot the Exec directly from `/usr/stand`, bootblocks must be installed on the `/usr` partition.

To install a bootblock in `/usr/stand`, use the following sequence:

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

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
```

Once the bootblock is installed, you can boot directly from that partition.

For SunOS 4.x:

On a standalone workstation that is running SunOS 4.x, it is not necessary to run `installboot` to make a partition other than partition a bootable. Use the procedures given previously to extract the diagnostic from the tape, and then boot from the PROM monitor, using the `-a` argument, which prompts you to specify the device and name of the file you want to boot. The example below shows what you would enter to boot the Exec if you had loaded it in `/usr/stand`:

```
>b -a
root filesystem type (4.2 nfs ): 4.2
root device ( xy%d[a-h] sd%d[a-h] xd%d[a-h] ): sd0g
Boot: /stand/exec
```

After entering the device and path to the place where the Exec files are stored, the Exec will boot and you will see its main menu and this message:

```
Can't open file 'sd(0,0,0)diags'.
```

You must then change the "load" variable in the Exec's Environment menu to match the device and path you have just booted. To do so, enter **e** (for Environment Menu) from the Exec's main menu, and then type a command such as this:

```
load=sd(0,0,6)/stand/
```

`sd` is for SCSI disk. The "6" in `(0,0,6)` stands for the partition "g", where the Exec resides in this example.

Now the Exec can read the `diags4` file and any tests residing in `/usr/stand` will be available for use under the Exec.

Booting from Disk

The Exec is booted from the PROM monitor prompt. It uses the monitor command line to determine the pathname (target directory) to its remaining files and diagnostics. The Exec first reads the `diags` file, which is an ASCII text file containing a list of all the diagnostics. Once this is read, the `autoexec` file is searched for along this same bootpath. The Exec automatically searches for this script file upon startup. The requested diagnostics are loaded from this bootpath and the logfiles are written to this target directory. Therefore, to boot the Exec from disk — either locally or across the Ethernet — copies of the Exec and the diagnostic programs must be available in the target directory from which they will be booted.

Autobooting the Exec

Sun systems can be configured so the Exec will boot automatically when the system is powered-on or reset with the CPU board diagnostic switch in the ON position. Setting this configuration requires programming the system's EEPROM to use the diagnostic bootpath. The EEPROM editing tool described in this manual may be used to perform this programming. Once you have programmed the diagnostics bootpath, do the following:

1. Halt the system (see the *Halting your System* section).
2. Set the normal/diagnostic switch on the system from NORM to `DIAG`.
3. Power-cycle the system or type **`k2`** from the monitor.

You can also start the Exec by booting the program `/stand/exec4` from the PROM monitor manually, as follows:

Booting from Local Disk

To boot from the root partition, simply:

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
```

If the Exec has been installed on a non-root partition, the parameters of the boot command line will depend on whether a bootblock has been installed on that partition, to make it a bootable partition.

If a bootblock has been installed and the Exec placed in `stand`, the boot command line will be:

```
>bdevice(0,0,partition)stand/exec4
```

device is the type of disk the Exec is booted from.

partition is the partition number.

If bootblocks are *not* installed, the Exec can still be booted with:

```
>b -a
Boot:device(0,0,0)vmunix
Load:device(0,0,0)vmunix
Boot:device(,,6)stand/exec4
```

Booting from Server/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 in order 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 ie(0,X,0)stand/exec4     (for 3.x servers)
or
>b ie(0,X,0)/stand/exec4    (for 4.x servers)
```

Replace *X* with the hexadecimal host number of the server that has the Exec on disk. (If your server has the Exec on it, see *shortcut*, on the next page)

NOTE *Absence of the " /" in front of* `stand` *causes* `/pub.SPARC` *to be prepended to the bootpath.*

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 system being booted
```

Shortcut

If the disk server of your test system 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/exec4
```

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

Invoking a Script File

If you have written a script file (described under the "Writing Script Files" at the end of this chapter), 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 the script file.

Disabling Cache

By default, the Exec will come up on a Sun-4/260 with the cache turned on. You may disable the cache by requesting it on the boot command line, like this:

```
>bdevice (ctlr,unit,part) bootpath source=filename cache=disable
```

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 after the PROM monitor prompt:

```
> b st()
boot: st(0,0,5)
```

or, for a 1/2-inch tape:

```
> b mt()
boot: mt(0,0,5)
```

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.

When booting the standalone diagnostics, such as `eccmemx.diag` from SCSI tape, use this sequence:

```
> b st()
boot: st(0,0,hex_file_number)
```

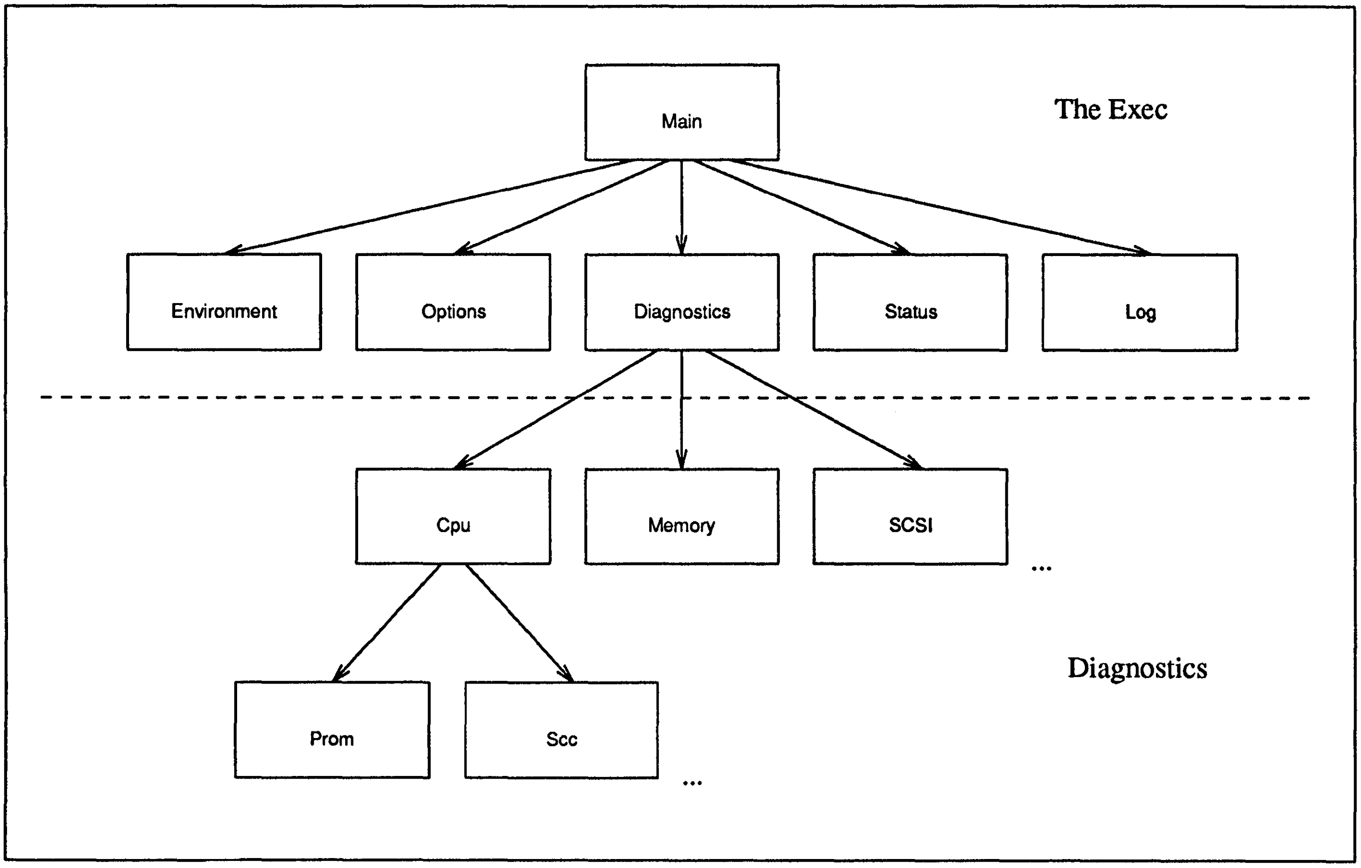
Substitute `mt` for `st` when booting from 1/2-inch tape. Consult the tape table of contents and enter the hexadecimal equivalent of the test file number in place of *hex_file_number*

2.6. 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 are selected from 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. 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 "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 from 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; therefore 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. For most applications, this is sufficient. 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 that permits execution of different diagnostics in a time-sliced environment. Once you have entered a diagnostic through the Diagnostics Menu, you may type the at-sign command (`@`) to place the diagnostic in the background. The diagnostic will run as a background process and the Diagnostic Menu will be displayed. The information that the diagnostic was sending to the screen will also be placed in the background. You are now free to start another diagnostic. You can have up to ten different diagnostics running at once, nine of them running in the background. Note that multiple copies of the same diagnostics may not be run simultaneously.

Once you have placed a diagnostic into the background, you may use the Status Menu to return it to the foreground. The Status Menu lists all the diagnostic programs currently running. Simply select the process number of diagnostic you want. That diagnostic will now take over the screen. You can now run other tests, move around the menus in that diagnostic, or do anything else you could normally do. To return the diagnostic to the background again, press the [@] key; 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.*

The discussion above applies straightforwardly when a single diagnostic is started from the Diagnostics Menu. However, the Exec also permits sequential diagnostic requests to be placed on one command line when you are in the Diagnostics Menu. A full discussion of this syntax can be found later in this chapter. The Exec waits for each diagnostic to complete before beginning the next diagnostic specified on the command line. During this processing, forcing the current diagnostic into background is interpreted as a completion by the command line processor. Consequently, the next diagnostic will begin to execute immediately. In turn, if this diagnostic is placed in the background, the next diagnostic on the command line will be to execute. When no more diagnostics are found on the command line, control passes to the Diagnostics Menu.

Whether they are running in the foreground or the background, 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.

2.7. User Interface

The user sees the Exec as a series of menus. The diagnostic programs 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 sub-menu "branch".

Menu Structure

Each menu contains a header line, which identifies the menu. The main body of the menu is a list of menu lines. Each menu line shows a menu command, followed by a short description of the command. At the bottom of the menu is the command prompt. The Main Menu is a typical example:

```
     Revision    Date     Sample Menu

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

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. 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 letter(s).

To make things easier, you need only enter enough of a command name for the Exec to distinguish between it and another command. This means you only have to type the letters capitalized in the command shown on the menu. You are free to type in more of the word if you want. In some cases this will be the entire word, if necessary to make the command recognizable to the Exec. 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.*

Also, you can type the commands in any mixture of upper or lower case; the Exec is not case sensitive.

Global Options

The Exec contains a set of global options that may be invoked from any menu in the menu tree. These commands are not listed on any menu. The global options provide general use features. They make it possible to move around the menu tree quickly, obtain help information, or re-execute a previous command. These options are described in the following text.

Esc *beep*
Pressing the [Esc] key 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 @ places the current diagnostic in background processing and returns you to the Exec from a diagnostic. 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.

NOTE *This option is the only command that does not require a carriage return.*
For more information, see *The Operating System Perspective.*

history
This option brings up the History Menu, from which you may execute any of the previous five commands.

! Entering an exclamation point is the same as entering the `history` command.

help
This option prints any available help messages for a particular command. Enter the word help, followed by the command name.

? Entering a question mark is the same as entering `help`.

alias
With this option you may create and save a command line. For example, if you wanted to simply type `test1` in order to run all the memory tests, from the CPU diagnostic menu, you would use the `alias` option this way:

```
alias test1="mem all addr=0 size=100000"
```

do This option invokes a previously defined alias. For example, if you had set up the "test1" alias as shown in above, you would enter

```
do test1
```

in order to execute the command line assigned to the alias named `test1`.

repeat
This option repeats a command line the number of times you specify. For example, you might enter something like:

```
repeat=10 "read_test;write_test"
```

Command Line Syntax

A command line is composed 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
```

Each command is composed of the command name and a list of command parameters, if applicable.

Command Parameters

Some commands may need parameters; they are listed in the help line for each particular command name. There are two types of parameters, which are differentiated by the presence of the = sign. The "equals" sign signifies an assignment operation and has the syntax:

command_name=value
or
parameter=value

Note that there are no spaces on either side of the equals sign. For example:

```
Command ==>mem all pattern=0x11
        (mem  has no equals sign)
        (all  has no equals sign)
        (pattern  has an equals sign)

Command ==>script=testscript        (script= has an equals sign)
```

Passing Commands

There is a way to pass a command string to a diagnostic from the Diagnostic Menu or its sub-menus. To accomplish this, use the cmd parameter. For example, if you entered cpu from the Diagnostics Menu, and did not add an argument, it would bring up a menu of sub-commands that run various CPU tests. Among the tests listed in this submenu are the scc and clock commands. If you wanted to run just these tests without going to the cpu menu, enter the following command from the Diagnostics Menu:

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

Note that the scc and clock commands are separated by a semicolon, since the two commands are to be run sequentially.

The spaces before and after the semi-colon require that you use quotation marks around the command string.

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

```
Command ==>cpu cmd="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. The correct syntax would 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 wherever the Exec expects a parameter. For example, when you enter a * for the number of times the test will run (the *pass=* parameter), 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 all of them 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
```

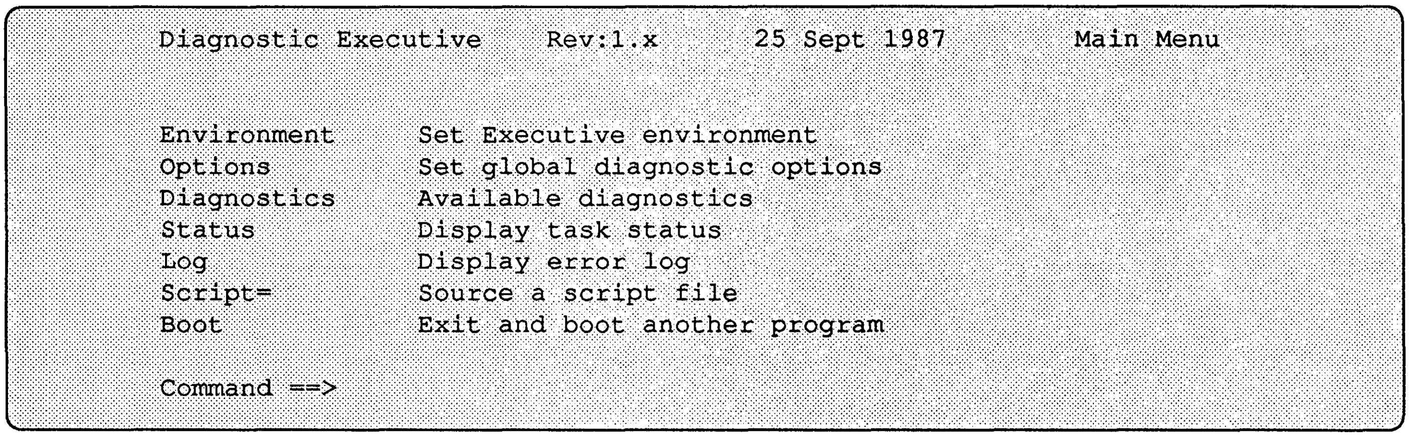
would be interpreted as three separate commands, not a command with two options. The spaces between items are optional.

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.



```
Diagnostic Executive    Rev:1.x      25 Sept 1987        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 **e** from the main menu moves you into the Environment Menu. Go to this menu to configure the Exec to its operating environment. See The Environment Menu section for details.

`options`

Selecting **o** from the main menu 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 **d** from the main menu 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 **s** from the main menu 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 **`l`** from the main menu 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*

Entering **script=** and a file name causes the Exec to read commands from the file you name instead of reading commands from the console. The *scriptfile* contains a sequence of command lines, arranged in a script, to be run by the Exec. This command allows you to run a pre-defined sequence of tests automatically. When the Exec runs to the end of the file, control returns to you.

Refer to *Writing Script Files* later in this chapter, for more information.

boot *boot-line*

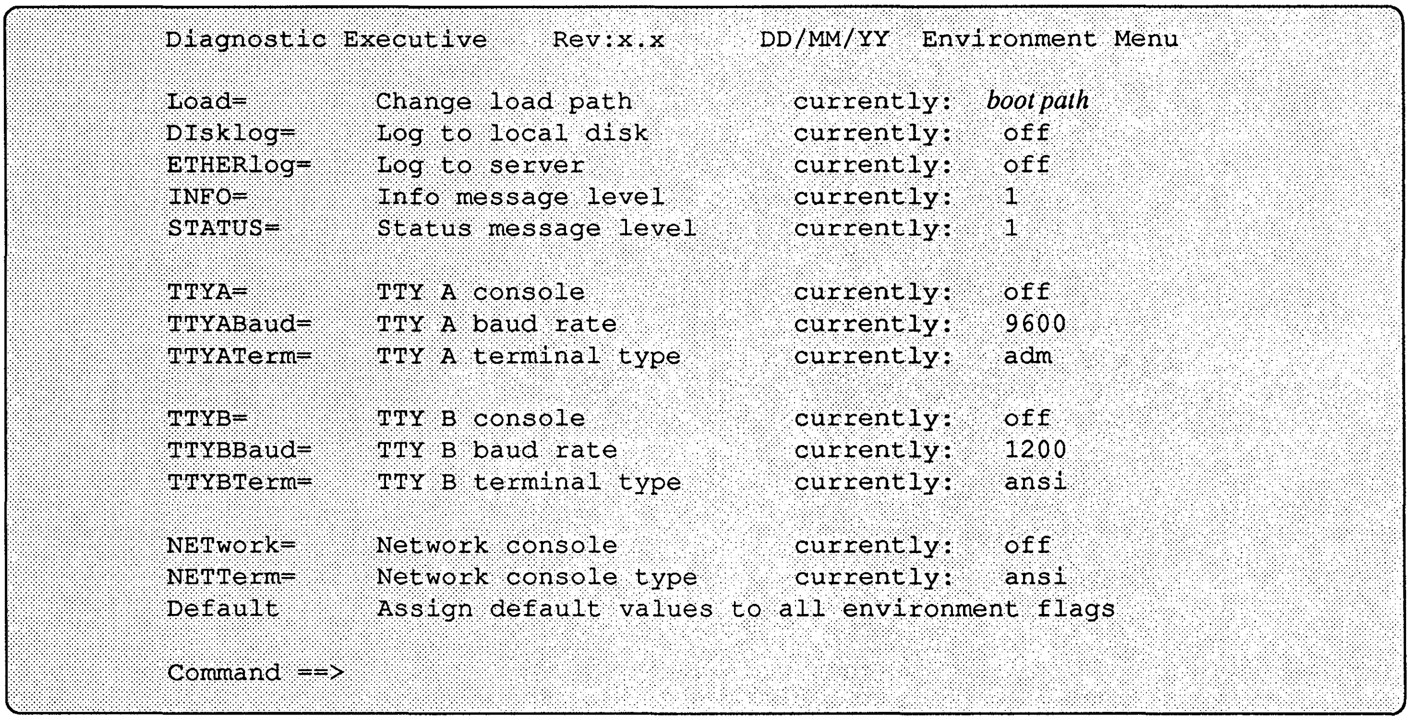
Selecting this command causes the Exec to exit and reboot using the boot-line. To boot `vmunix`, type from disk:

```
boot sd(0,0,0)vmunix
```

Running `boot` without any arguments restarts the 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 system 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.



```
Diagnostic Executive    Rev:x.x      DD/MM/YY  Environment Menu

Load=         Change load path           currently:  bootpath
DIsklog=      Log to local disk          currently:   off
ETHERlog=     Log to server              currently:   off
INFO=         Info message level         currently:   1
STATUS=       Status message level       currently:   1

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 ==>
```

Following are descriptions of each Environment Menu 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()	st()	ie()
xy()	xt()	le()
xd()		
fd()		

Combine the device name with the directory path on the device to make *loadpath*. End the loadpath with a diagonal (/). If the boot device is, for example, SCSI disk zero (sd0) and the directory is `/stand`, then enter **loadpath=sd()/stand/**

disklog=*enableflag*

This parameter enables or disables logging the disklog onto local disk. To enable logging, *enableflag* should be replaced with `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 disklog. To disable logging, enter **disklog=off**. If the Exec is already logging messages, that procedure stops, and any new messages are not sent to the disklog.

The disklog is saved in a file called `logfile` in the directory indicated by the `loadpath` variable on the test system's **local** disk. The file must already exist on the local disk, with at least 32K bytes of data in it, **before** you run the Exec.

etherlog=*enableflag*

NOTE *Do not enable remote logging to a server when* `netcon` *is also enabled. Doing so causes Ethernet chip contention problems.*

This parameter enables or disables the logging process to the server, when booting as a diskless client. To enable logging, the *enableflag* should be replaced withon. 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 remote logfile. To disable logging, enter **etherlog=off**. If the Exec is already logging messages, that procedure stops, and any new messages are discarded.

On a server, the logfile is placed in

loadpath/`etherlog`/*ethernet_number*.

The file must already exist on the server, **before** you run the Exec. To obtain the *ethernet_number*.

1. On the Exec console, type in **`set`**. Enter that number *exactly* as you see it, including zeroes and capital letters, as the directory name you are creating:

   loadpath/**`etherlog`**/*ethernet_number*

2. On the server, change directories to the *ethernet_number* directory and type in **`touch logfile`** to create an Ethernet logfile with the current date.

3. Make sure the file and directory can be written by the group or the public. Use `chmod 666 logfile` to make the file accessible to the public.

`info=`*info_level*

This parameter sets the level of detail you want in information messages. Possible level entries are 0, 1, 2 and 3. Setting the level to zero suppresses all informational messages; setting it to three gives the most detailed messages.

`status=`

This parameter sets the level of detail you want in status messages. Possible level entries are 0, 1, 2 and 3. Setting the level to zero suppresses all informational messages; setting it to three gives the most detailed messages.

`ttya=`*enableflag*

This command tells the Exec whether you are using a terminal connected to Serial Port A on the test system. If you are, *enableflag* should be replaced with the string `on`. If you're not, use the **`ttya=off`**. The Exec is controlled from one or more sources, called consoles. The console default is the Sun keyboard and monitor. 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 CPU board's Serial Port A. 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 or not you are using a modem connected to the Serial Port B on the test system. If you are, *enableflag* should replaced with be **on**. If you're not, enter **ttyb=off**. The Exec is controlled from one or more sources, called consoles. The console default is the Sun keyboard and monitor. 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 telephone line.

ttybbaud=*baudrate*

This command sets the baud rate for the Serial Port B 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 1200.

ttybterm=*termtype*

This command sets the terminal type the Exec expects to be connected to Serial Port B. The legal values are `ansi`, `adm`, or `tty`. The table under the `ttyaterm=` heading shows when to use each. The `tty` setting contains no escape sequences, so it will work on nearly any terminal, display or printer.

network=*enableflag*

This command notifies the Exec to connect with a **netcon** session on some remote system on the net. To start the remote session, enter **network=on**. To disconnect the remote session, enter **network=off**.

netterm=*termtype*

This command sets the terminal type the Exec expects to be connected to 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 below shows all of the terminal selections and when to use them:

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.

`default`

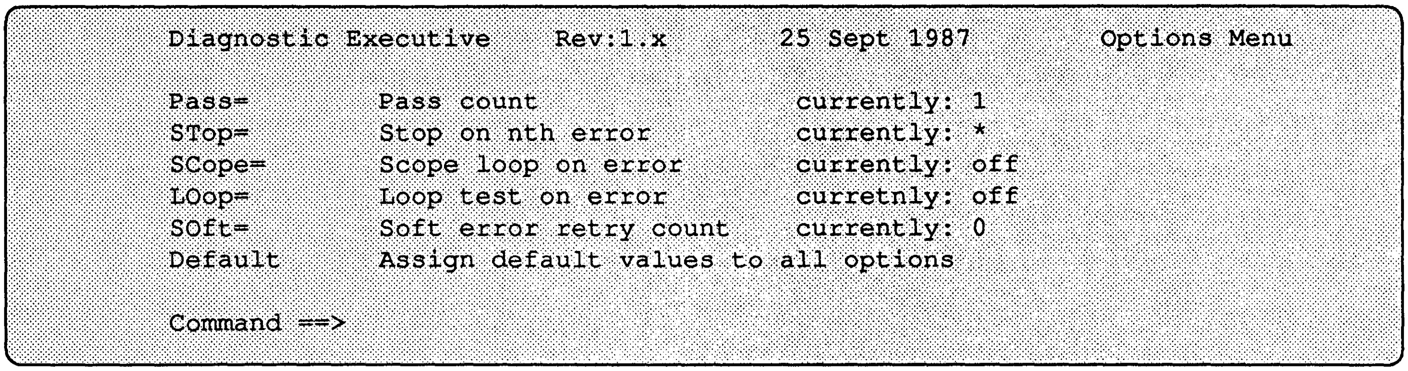
This command sets all of the values in this menu to their defaults. The default values are listed in the table below:

Command	*Default*
`Load=`	*boot path*
`disklog=`	off
`etherlog=`	off
`info=`	1
`status=`	1
`ttyb=`	off
`ttybbaud=`	1200
`ttybterm=`	ansi
`ttya=`	off
`ttyabaud=`	9600
`ttyaterm=`	adm
`network=`	off
`netterm=`	ansi

The Options Menu

The commands in the Options Menu control how the diagnostic tests will react to errors. You can simultaneously set certain characteristics for all the diagnostics.

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.



```
Diagnostic Executive    Rev:1.x        25 Sept 1987        Options Menu

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

Command ==>
```

`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 menu command is entered with a `pass=` argument.

`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 supercedes the `stop=` setting; 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. This setting only affects diagnostics that have a scopeloop test in their menus.

`loop=`

This parameter is intended for future Exec enhancement; it is not currently used.

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 could be in RAM, disk, registers, and so on.) 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 judgement. 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.

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
`loop=`	off		

Diagnostics 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 systems have various configurations and therefore need specific 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 Line 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.*

If you bring up the Diagnostics Menu from a Sun-3 system, it will look something like this:

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

CG6.......... P4 Low-End Graphics Accelerator Diagnostic
CG8.......... P4 24-Bit Frame Buffer Diagnostic
CG9.......... VME 24-Bit Frame Buffer Diagnostic
COlor ....... Generic VME Color Board (CG2,CG3,CG5) Diagnostic
CPCache...... CPU Cache (Sun-3/400) Diagnostic
CPU ......... CPU Board Diagnostic
EEprom ...... EEPROM Editing Tool
Ether ....... Ethernet Diagnostic
ETHer2 ...... Ethernet II Board Diagnostic
FDC ......... Floppy Disk Drive Diagnostic
FDDi......... FDDI Board Diagnostic
FPA ......... Sun-3 FPA Board Diagnostic
FPA_Plus .... Sun-3/400 FPA+ Diagnostic
Gp .......... Graphics Processor 1 and Graphics Buffer Diagnostic
GP2 ......... Graphics Processor 2 Diagnostic
MORE

Command==>
```

If you enter **MORE**, the remainder of the Sun-3 menu is displayed:

```
HSS ......... High Speed Serial Board Diagnostic
IOCache ..... IO Cache (Sun-3/400) Diagnostic
IPi ......... IPI Board Diagnostic
Kb .......... Keyboard Diagnostic
MCp ......... MCP and ALM-2 Board Diagnostic
Mem ......... Memory Diagnostic
MOuse ....... Mouse Diagnostic
MTi ......... MTI/ALM Board Diagnostic
Parity ...... Parity Memory (Sun-3/80) Diagnostic
SMD ......... SMD Controller/Disk Diagnostic
SUBsystem ... SCSI Sub-system Diagnostic
ESPscsi...... ESP SCSI (Sun-3/80) Diagnostic
TAAc ........ TAAC Accelerator Board Diagnostics
Tape ........ Pertec 1/2" Tape Diagnostic
Video ....... Video Circuit Diagnostic
VIDMon ...... Video Monitor Diagnostic
VME ......... VME Interface Diagnostic

Command==>
```

If you bring up the diagnostics menu from a Sun-4 or SPARC system, this sort of menu comes up:

```
Diagnostic Executive    Rev:1.x    dd mm yy     Diagnostics Menu
CG6.......... P4 Low-End Graphics Accelerator Diagnostic
CG8........... P4 24-Bit Frame Buffer Diagnostic
CG9........... VME 24 Bit Frame Buffer Diagnostic
COlor ........ Generic VME Color Board (CG2,CG3,CG5) Diagnostic
CPCache ...... CPU Cache (SPARCstation 330) Diagnostic
CPu .......... CPU Board Diagnostic
EEprom ....... EEPROM Editing Tool
Ether ........ Ethernet Diagnostic
ETHER2 ....... Ethernet II Board Diagnostic
FDDi.......... FDDI Board Diagnostic
FPU .......... FPU Diagnostic
Gp ........... Graphics Processor 1 and Graphics Buffer Diagnostic
GP2 .......... Graphics Processor 2 Diagnostic
HSS .......... High Speed Serial Board Diagnostic
IPi .......... IPI Board Diagnostic
Kb ........... Keyboard Diagnostic
MCp .......... MCP and ALM-2 Board Diagnostic
Mem .......... Memory Diagnostic
MORE

Command==>
```

If you enter **MORE**, the remainder of the Sun-4 menu is displayed:

```
MOuse ........ Mouse Diagnostic
MTi .......... MTI/ALM Board Diagnostic
SMD .......... SMD Controller/Disk Diagnostic
SUBsystem .... SCSI Sub-system Diagnostic
ESPscsi ...... ESP SCSI (SPARCstation 330) Diagnostic
TAAc ......... TAAC Accelerator Board Diagnostics
Tape ......... Pertec 1/2" Tape Diagnostic
Video ........ Video Circuit Diagnostic
VIDMon ....... Video Monitor Diagnostic
VME .......... VME Interface Diagnostic

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 Diagnostics Menu.*

Starting A Diagnostic

Entering the name of a diagnostic (or just the letters shown in upper case on the menu) puts you in that program's main menu. All the diagnostics are designed to have a similar user interface. Each top menu will probably have:

- A "Run All Tests" command that runs every test in the diagnostic.
- A "Run Quick Tests" command that runs a subset of all the tests, taking 2 minutes or less.
- A "Run Default Tests" command that 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 given a specific 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

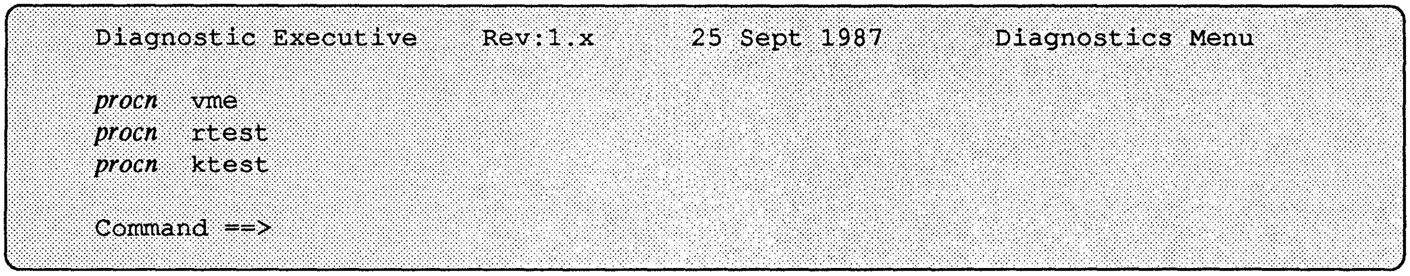
```
cpu cmd=all bg
```

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. The content of this menu depend 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:



```
Diagnostic 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.

`@`

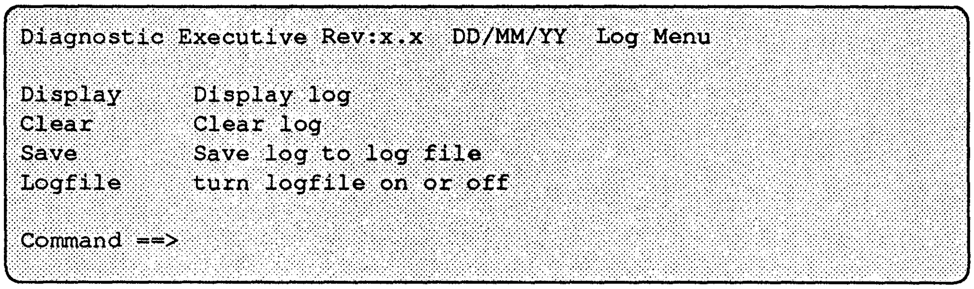
Entering an at-sign character (`@`) will stop a test from displaying on the screen and start it running in the background. The current status menu is redisplayed. This command has no effect when there are no tests running in the foreground.

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. *error_location* describes what part of the test failed. The *error_info* section provides more information about the particular error.

Here is an example of the Log Menu:



```
Diagnostic 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 disklog is saved in a file called* `logfile` *in the directory indicated by the loadpath variable on the test system's* **local** *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.

2.9. Writing Script Files

Once you are familiar with the menu structure of the Exec and its diagnostics, you may write script files for the Exec to automatically execute.

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 as though you were running all the diagnostics from the Diagnostics 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 Diagnostics 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 no provisions for dealing with continuation lines; therefore, the second command line must exist as a single input line. The following is an example of a script that runs three diagnostic sequences consecutively.

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

The example above first selects the Diagnostics Menu from the Exec's Main Menu. The next line selects the CPU ''quick'' test sequence, then escapes back to the Diagnostics Menu. After the second semicolon, the VIDEO default test is selected from the Diagnostics Menu. Control returns to the previous menu (the main menu) before finally selecting Memory quick test. After this last test the script escapes 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 command that is passed to a diagnostic can be as complex as necessary. The following is an example of a script to run specific diagnostics from the diagnostic submenus:

```
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, and another quicktest sequence is invoked from that menu. Finally, the two escapes bring you up through the CPU diagnostic main menu to the Diagnostics Menu.

When the Exec reads 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 Diagnostics Menu section in this chapter for details.

2.10. Remote Execution and the Network Console

The Exec can be controlled by any console on a local network. This remote console is treated as a second console. The combination of this feature with the SunOS `tip` command provides for remote diagnostic execution through any modem connection by way of the server for the local net.

To run the Exec remotely over Ethernet, the `netcon` program must be installed on a system that is running the SunOS operating system. `netcon` is included on the Exec release tape, and is initially installed in the same directory as the Exec and its diagnostics. This program can be copied to wherever you normally install SunOS executables.

Once this program is is installed, the Exec can run remotely over Ethernet. To do this, first enter 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. Do not enable remote logging to a server when netcon is also enabled.

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 go to the Exec's Environment Menu. First you must set the terminal type by assigning the `netterm=` variable. Finally, to acquire the network console, type **network=on**. The Exec will now attempt to connect to the remote console. Once under netcon, the remote screen will act like an *ansi* terminal.

CAUTION **The remote console should be running on the network before network= is enabled from the Environment Menu. If you enable network= when a network console is not 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.**

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
```

3

Sun-3/400 Series Central Cache Diagnostic

3

Sun-3/400 Series Central Cache Diagnostic

3.1. General Description

The central cache is an area of fast RAM used to contain 16-byte blocks of data from main memory. Central cache must be consistent with main memory, to guarantee no loss of data from memory. The Central Cache Diagnostic is designed to verify the correct operation of the central cache. The diagnostic also tests the blockcopy command, a function code 4 control space command in Sun-3 systems.

3.2. What This Chapter Contains

Following an overview of the diagnostic and a list of required hardware and firmware, the Main Menu and submenus are discussed. Menu selections and optional parameters are listed, along with brief descriptions. The end of the chapter contains a description of test messages and a glossary.

3.3. Overview of the Diagnostic

The Central Cache Diagnostic allows you to test all aspects of the central cache by running one continuous sequence of tests. It also lets you run individual tests, for specific debugging purposes.

The Central Cache Diagnostic tests all cache operations that can be performed and all the elements of the central cache, including Tag RAM, Data RAM, cache logic, and blockcopy circuitry.

The parameters of each test are given default values upon execution. Online help is provided. The Central Cache Diagnostic also generates and logs error messages for later retrieval.

3.4. Hardware Requirements

The following hardware is required to run the Central Cache Diagnostic:

- A Sun-3/400 series cardcage capable of supporting a Sun-3/400 series CPU board and a memory board
- A Sun-3/400 series CPU board
- A Sun-3/400 series memory board
- A monitor
- A keyboard
- A boot device (local disk, local tape, or remote disk over Ethernet)

3.5. Firmware Requirements

The Sun-3/400 series power-up boot PROM is required to run the Central Cache Diagnostic. Execution of the power-up selftest is necessary to verify the functionality of the Sun-3/400 series CPU board before the diagnostic is run.

3.6. Additional Requirements

A few additional conditions apply when running the Central Cache Diagnostic:

- The diagnostic program loads into memory pages marked "non-cacheable."
- When the Central Cache Diagnostic is started, it automatically disables all other caches that can cache data in the system, to ensure that the diagnostic code does not find its way into the central cache when the cache is under test.
- Another diagnostic, the I/O Cache Diagnostic, supplements the testing of the central cache. Because it tests the consistency between the central cache and the I/O cache, it is recommended that the I/O Cache Diagnostic be run also, for complete testing of the cache system.

3.7. User Interface

The user interface of the Central Cache Diagnostic adheres to the standards of the Exec menu system. Each test may be selected from a menu by typing the letter or letters displayed in uppercase in the column on the left side of the menu. Additional parameters and options may be specified on the command line.

A Main Menu and five submenus are provided. The commands on the submenus provide for testing the general functions of the central cache, the blockcopy command, and operations which cross cache block boundaries. A submenu also provides access to options which control and display environment variables and the error log. All, Default, and Quick Test sequences are provided on the Main Menu and submenus.

Press ESC then RETURN to return to the Main Menu from a submenu. To exit the diagnostic, press ESC while the Main Menu is displayed.

To display online Help for any menu option, enter the following on the command line:

help *option_name*

3.8. Starting the Diagnostic

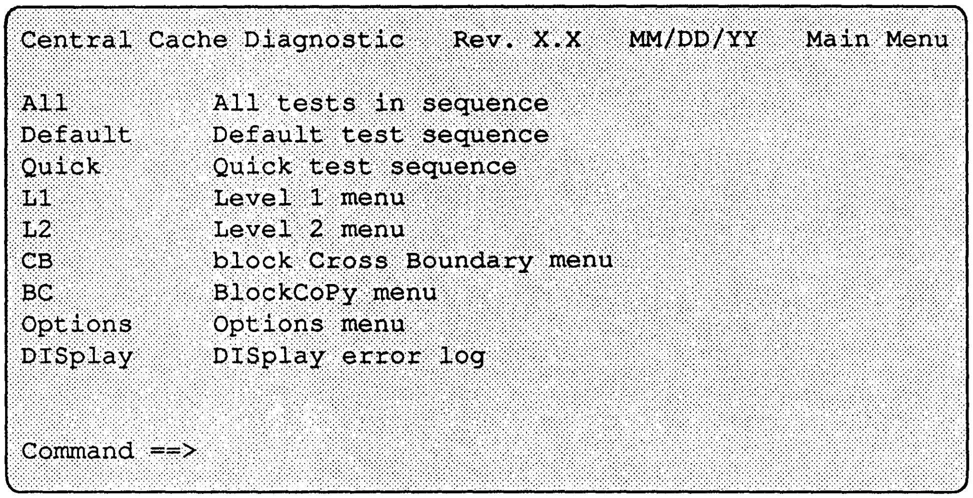
For information on starting the Exec, refer to Chapter 2, "Using the SunDiagnostic Executive." After you have started the Exec, choose the Central Cache Diagnostic from the Diagnostics Menu.

3.9. The Diagnostic Menus

This section of the chapter provides a modular description of the Central Cache Diagnostic, beginning with the Main Menu and working down through the options available on each of the submenus.

Main Menu

The Main Menu, which displays when you start the Central Cache Diagnostic, provides access to the submenus of the individual tests:



```
Central Cache Diagnostic   Rev. X.X   MM/DD/YY   Main Menu

All         All tests in sequence
Default     Default test sequence
Quick       Quick test sequence
L1          Level 1 menu
L2          Level 2 menu
CB          block Cross Boundary menu
BC          BlockCoPy menu
Options     Options menu
DISplay     DISplay error log


Command ==>
```

Command Parameters

The test selections on the submenus, as well as the All, Default, and Quick selections on the Main Menu, take optional parameters. You specify the parameters by entering them on the command line when you run individual tests. If no parameters are specified, the default values are used. The meanings of the parameters, and their default values, are shown in the following table.

NOTE *When parameter values are described in a table as a range, the values may be combined. For example, a range of parameter values specified as* 1–7 *means that the values 1 through 7, inclusive, are allowed, and the values may be combined. Therefore, if the description of the parameter range lists values such as* 1=walking zero, 2=walking 1, *or* 4=constant, *you may combine values to select multiple options. A value of 5 (1+4) would select* walking zero *and* constant *options. If parameter values are specified separately — not as a range — they may not be combined to select multiple options.*

Parameter	*Description*	*Default*
`Begin`	Beginning hexadecimal memory address (1–Memory size).	1
`End`	Ending hexadecimal memory address (1–Memory size).	Memory size
`COmplement`	Use of inverted or non-inverted data. 0 (non-inverted) 1 (inverted)	0
`INCr`	Increment value.	1
`ITeration`	Number of times each subtest is run per pass.	1
`MOde`	Data size used in test. 0 (byte) 1 (word — two bytes) 2 (long word — four bytes)	0
`PATtern`	Hexadecimal data pattern used in constant pattern test (0–FFFFFFFF).	55555555
`SEed`	Hexadecimal seed used in random pattern test (0–FFFFFFFF).	5
`SUBtest`	Subtests to be run.	Varies by test

Certain other parameters are specific to individual tests. These parameters, and their default values, are described together with the tests to which they apply.

Environmental Parameters

The choices on the Options Menu allow you to set global environmental parameters that control how the Central Cache Diagnostic tests perform. You can, for example, specify the number of passes, limit the number of retries on error, and set the level of verbosity of informational and status messages. All of the global environmental variables may be used with all of the tests on the Main Menu and submenus, with the exception of the commands on the Options Menu itself.

The current values of the environmental parameters are shown on the Options Menu. You can specify different values in two ways:

- You can select the parameters on the Options Menu and reset the current values.
- You can enter new values on the command line for a *subset* of the parameters on the Options Menu.

A *local* environmental parameter that you enter on the command line of a specific test is in effect for the duration of the test only. If you do not enter a local environmental parameter, the current value on the Options Menu is used. The local environmental parameters that you may enter on the command line are summarized in the following table:

Parameter	*Description*	*Default*
`Pass`	Pass count.	1
`SCope`	Scope on error. 0 (off) 1 (on)	0
`SOft`	Soft error retry limit.	0
`STop`	Stop on error count.	0
`WAit`	Set wait flag. 0 (off) 1 (on)	0
`INFO`	Set informational message level. 0 (off) 1 (terse) 2 (verbose 3 (more verbose)	1
`STATUS`	Set status message level. 0 (off) 1 (terse) 2 (verbose 3 (more verbose)	1

All Tests in Sequence

A

The *All Tests in Sequence* option on the Main Menu executes a sequence of all tests in all the submenus of the diagnostic, in the order that they appear on the submenus.

Default Test Sequence

D

The *Default Test Sequence* option on the Main Menu executes the Default Test Sequences on all the submenus in the order that they appear on the submenus.

Quick Test Sequence

Q

The *Quick Test Sequence* option on the Main Menu executes the Quick Test Sequences on all the submenus in the order that they appear on the submenus.

Level 1 Menu

When you choose **L1** from the Main Menu, the Level 1 Menu is displayed:

```
Central Cache Diagnostic   Rev. X.X   MM/DD/YY   Level 1 Menu

All          All tests in sequence
Default      Default test sequence
Quick        Quick test sequence
CD           cpu Cache Disabled (bypass)
DR           Data Ram test
TR           Tag Ram test
RNC          Read from non-cacheable page
WNC          Write to a non-cacheable page
RH           Read Hit
RM           Read Miss
RMWB         Read Miss with WB
WH           Write Hit
FWH          First Write Hit
DISplay      DISplay error log


Command ==>
```

The options on the Level 1 Menu are designed to test the general functionality of the central cache: cache disbaling, reading and writing to cache blocks and to a non-cacheable page, and the functionality of the Data and Tag RAM.

A

The *All Tests in Sequence* option on the Level 1 Menu executes a sequence of all of the tests on that menu.

D

The *Default Test Sequence* option on the Level 1 Menu executes a series of all the tests on the menu.

Q

The *Quick Test Sequence* option on the Level 1 Menu executes a sequence of all the tests on the menu.

CD *b= e= inc=*

The *CPU Cache Disabled* option on the Level 1 Menu tests the cache disable function. The test verifies that if the central cache is disabled, all memory accesses to main memory bypass the central cache and go directly to main memory.

DR *co= inc= it= mo= pat= se= sub=*

The *Data RAM Test* option on the Level 1 Menu verifies the correct operation of the Data RAM. For each pass of this test, each subtest is run the number of times specified by the value of `it=`. The subtests specific to this test are summarized in the following table:

Parameter	*Description*	*Default*
`SUBtest`	Subtests to be run (1–7F). 1 (address pattern test) 2 (constant pattern test) 4 (NTA pattern test) 8 (unique pattern test) 0x10 (checker pattern test) 0x20 (mats pattern test) 0x40 (random pattern test)	7F

TR *co= pat= sub=*

The *Tag RAM Test* option on the Level 1 Menu verifies the correct operation of Tag RAM. For each pass of this test, each subtest is run. The subtests specific to this test are summarized in the following table:

Parameter	*Description*	*Default*
`SUBtest`	Subtests to be run (1–1F). 1 (write/read test) 2 (inverse tag test) 4 (pattern write/read test) 8 (walking zeros/walking ones test) 0x10 (three-pattern test)	1F

RNC

The *Read from Non-cacheable Page* option on the Level 1 Menu verifies correct operation when reading from a memory page that is not cacheable. The test attempts to cache a page that is not cacheable. The read operation should bypass the cache and read directly from main memory.

WNC

The *Write to a Non-cacheable Page* option on the Level 1 Menu verifies correct operation when writing to a memory page that is not cacheable. The test attempts to cache a page that is not cacheable. The write operation should bypass the cache and write directly to main memory.

RH

The *Read Hit* option on the Level 1 Menu verifies the correct function of a cache read hit operation. The data of the accessed location is in the cache, and the cache block is not modified. Data should be read from the cache block, not from main memory. In the second pass of this test, the read operation is repeated with the cache block modified. The state of the M bit should have no effect on the results of this test.

RM

The *Read Miss* option on the Level 1 Menu verifies the correct function of a cache read miss operation. In this test, data is not in the cache, and the cache block is not modified. Data for location *x* can be loaded into the data cache from main memory. The read should proceed by reading from the data cache [(x(15:4)].

RMWB

The *Read Miss with WB* option on the Level 1 Menu verifies the correct function of a cache read miss with writeback operation. In this test, data for location *x* is not in the cache, and the cache block is modified. The modified cache block must be written back to main memory at location *y* before new data for location *x* is loaded into the data cache from main memory.

WH

The *Write Hit* option on the Level 1 Menu verifies the correct function of a cache write hit operation. In this test, data for the accessed location is in the cache, and the cache block is not modified. The cache block should be updated with the M bit set.

FWH

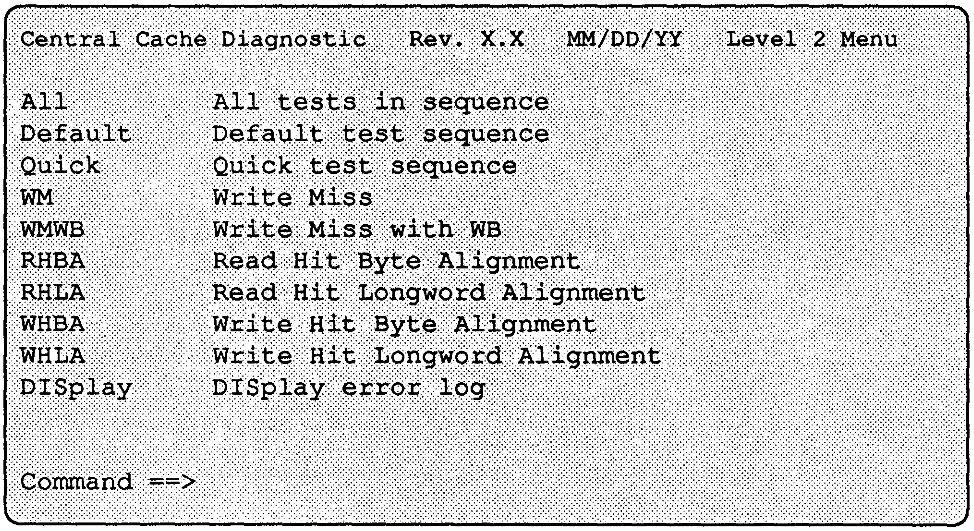
The *First Write Hit* option on the Level 1 Menu verifies the correct function of a first write hit operation.

DIS

The *Display Error Log* option on the Level 1 Menu displays the Central Cache Diagnostic error log.

Level 2 Menu

When you choose **L2** from the Main Menu, the Level 2 Menu is displayed:



```
Central Cache Diagnostic   Rev. X.X   MM/DD/YY   Level 2 Menu

All          All tests in sequence
Default      Default test sequence
Quick        Quick test sequence
WM           Write Miss
WMWB         Write Miss with WB
RHBA         Read Hit Byte Alignment
RHLA         Read Hit Longword Alignment
WHBA         Write Hit Byte Alignment
WHLA         Write Hit Longword Alignment
DISplay      DISplay error log


Command ==>
```

The options on the Level 2 Menu are designed to test the functionality of the central cache slave interface.

A

The *All Tests in Sequence* option on the Level 2 Menu executes a sequence of all of the tests on that menu.

D

The *Default Test Sequence* option on the Level 2 Menu executes a sequence of all tests on the menu.

Q

The *Quick Test Sequence* option on the Level 2 Menu executes a sequence of all tests on the menu.

WM

The *Write Miss* option on the Level 2 Menu verifies the correct function of a cache write miss operation. In this test, data is not in the data cache, and the data cache is not modified. Data for location *x* can be loaded into the data cache from main memory. The write should proceed by writing into the data cache at (x(15:4) and setting the M bit of cache tags.

WMWB

The *Write Miss with WB* option on the Level 2 Menu verifies the correct function of a cache write miss with writeback operation. With the central cache enabled, this test writes to a location *x* with the data for *x* not in the cache and the cache block modified. The modified cache block must be written back to main memory at location *y* before data for location *x* is loaded into the data cache. The write operation modifies the new cache block at location x(15:4).

RHBA

The *Read Hit Byte Alignment* option on the Level 2 Menu verifies the correct function of a read hit operation within a block with byte-aligned operands. The test confirms that byte alignment is correct for all byte move/read instructions that are read hits. The same data is read with the cache enabled then disabled, and the results are compared.

RHLA

The *Read Hit Longword Alignment* option on the Level 2 Menu verifies the correct function of a read hit operation within a block with longword-aligned operands. The test confirms that byte alignment is correct for all longword move/read instructions that are read hits. The same data is read with the cache enabled then disabled, and the results are compared.

WHBA

The *Write Hit Byte Alignment* option on the Level 2 Menu verifies the correct function of a write hit operation within a block with byte-aligned operands. The test confirms that byte alignment is correct for all byte move/write instructions that are write hits. The same data is written with the cache enabled then disabled, and the results are compared.

WHLA

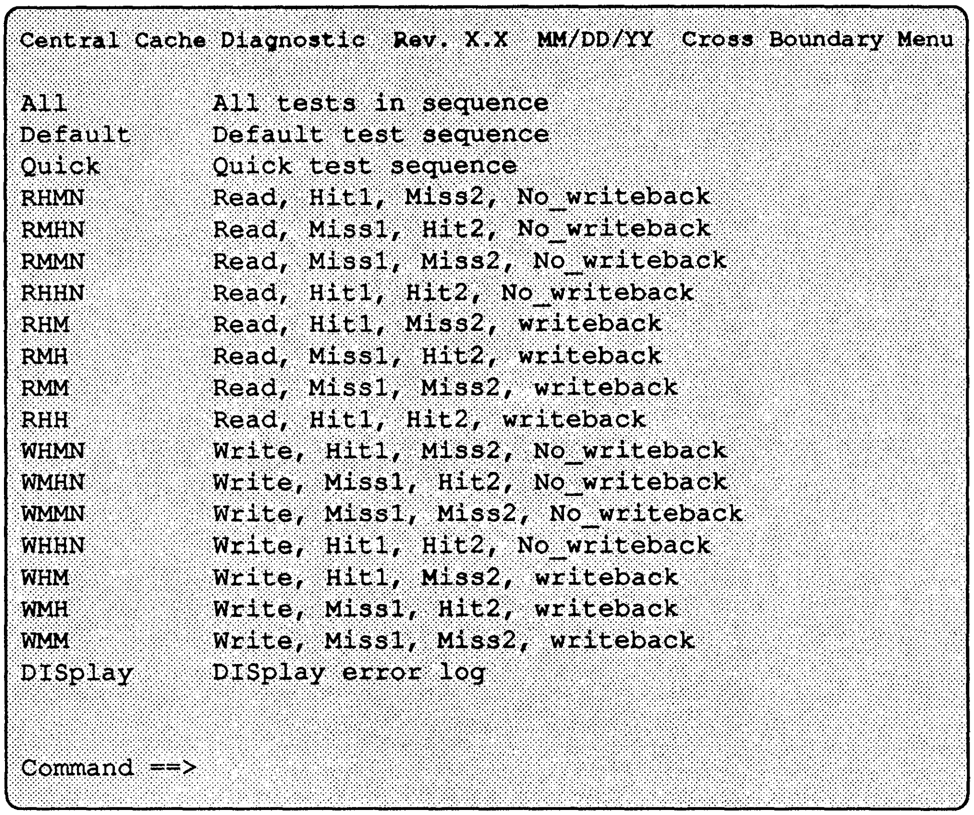
The *Write Hit Longword Alignment* option on the Level 2 Menu verifies the correct function of a write hit operation within a block with longword-aligned operands. The test confirms that byte alignment is correct for all longword move/write instructions that are write hits. The same data is read with the cache enabled then disabled, and the results are compared.

DIS

The *Display Error Log* option on the Level 2 Menu displays the Central Cache Diagnostic error log. For details, see the description of this option in the section on the Level 1 Menu earlier in this chapter.

Cross Boundary Menu

When you choose **CB** from the Main Menu, the Cross Boundary Menu is displayed:



```
Central Cache Diagnostic  Rev. X.X  MM/DD/YY  Cross Boundary Menu

All           All tests in sequence
Default       Default test sequence
Quick         Quick test sequence
RHMN          Read, Hit1, Miss2, No_writeback
RMHN          Read, Miss1, Hit2, No_writeback
RMMN          Read, Miss1, Miss2, No_writeback
RHHN          Read, Hit1, Hit2, No_writeback
RHM           Read, Hit1, Miss2, writeback
RMH           Read, Miss1, Hit2, writeback
RMM           Read, Miss1, Miss2, writeback
RHH           Read, Hit1, Hit2, writeback
WHMN          Write, Hit1, Miss2, No_writeback
WMHN          Write, Miss1, Hit2, No_writeback
WMMN          Write, Miss1, Miss2, No_writeback
WHHN          Write, Hit1, Hit2, No_writeback
WHM           Write, Hit1, Miss2, writeback
WMH           Write, Miss1, Hit2, writeback
WMM           Write, Miss1, Miss2, writeback
DISplay       DISplay error log


Command ==>
```

The options on the Cross Boundary Menu are designed to test correct cache operation for memory read/write operations with longword operands that cross cache block boundaries. The menu offers a variety of read and write operations to verify that a word access to a location that crosses the block boundary of adjacent cache blocks will generate correct combinations of hits, misses, and writebacks of cache data blocks.

A

The *All Tests in Sequence* option on the Cross Boundary Menu executes a sequence of all of the tests on that menu.

D

The *Default Test Sequence* option on the Cross Boundary Menu executes a sequence of all tests on the menu.

Q

The *Quick Test Sequence* option on the Cross Boundary Menu executes a sequence of all tests on the menu.

RHMN

The *Read, Hit1, Miss2, No Writeback* option on the Cross Boundary Menu verifies the correct read/hit/miss operation with no writeback.

RMHN

The *Read, Miss1, Hit2, No Writeback* option on the Cross Boundary Menu verifies the correct read/miss/hit operation with no writeback.

RMMN

The *Read, Miss1, Miss2, No Writeback* option on the Cross Boundary Menu verifies the correct read/miss/miss operation with no writeback.

RHHN

The *Read, Hit1, Hit2, No Writeback* option on the Cross Boundary Menu verifies the correct read/hit/hit operation with no writeback.

RHM

The *Read, Hit1, Miss2, Writeback* option on the Cross Boundary Menu verifies the correct read/hit/miss operation with writeback.

RMH

The *Read, Miss1, Hit2, Writeback* option on the Cross Boundary Menu verifies the correct read/miss/hit operation with writeback.

RMM

The *Read, Miss1, Miss2, Writeback* option on the Cross Boundary Menu verifies the correct read/miss/miss operation with writeback.

RHH

The *Read, Hit1, Hit2, Writeback* option on the Cross Boundary Menu verifies the correct read/hit/hit operation with writeback.

WHMN

The *Write, Hit1, Miss2, No Writeback* option on the Cross Boundary Menu verifies the correct write/hit/miss operation with no writeback.

WMHN

The *Write, Miss1, Hit2, No Writeback* option on the Cross Boundary Menu verifies the correct write/miss/hit operation with no writeback.

WMMN

The *Write, Miss1, Miss2, No Writeback* option on the Cross Boundary Menu verifies the correct write/miss/miss operation with no writeback.

WHHN

The *Write, Hit1, Hit2, No Writeback* option on the Cross Boundary Menu verifies the correct write/hit/hit operation with no writeback.

WHM

The *Write, Hit1, Miss2, Writeback* option on the Cross Boundary Menu verifies the correct write/hit/miss operation with writeback.

WMH

The *Write, Miss1, Hit2, Writeback* option on the Cross Boundary Menu verifies the correct write/miss/hit operation with writeback.

WMM

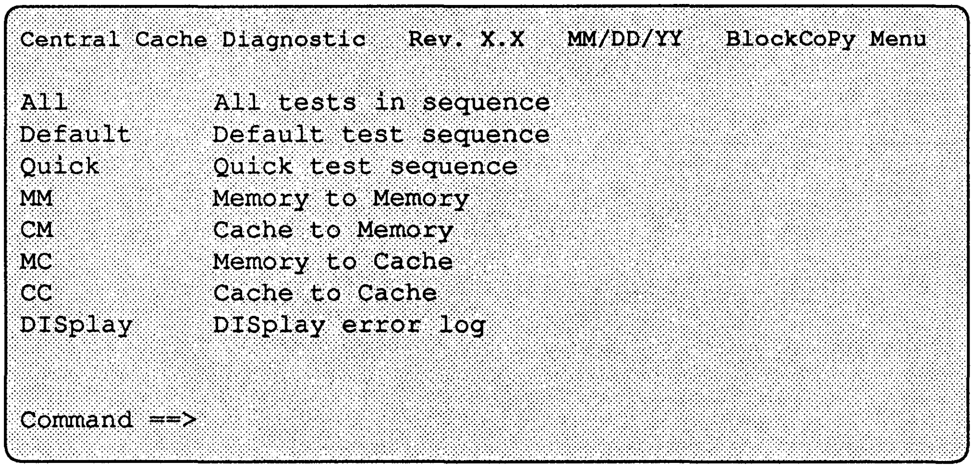
The *Write, Miss1, Miss2, Writeback* option on the Cross Boundary Menu verifies the correct write/miss/miss operation with writeback.

DIS

The *Display Error Log* option on the Cross Boundary Menu displays the Central Cache Diagnostic error log.

Blockcopy Menu

When you choose **BC** from the Main Menu, the Blockcopy Menu is displayed:



```
Central Cache Diagnostic   Rev. X.X   MM/DD/YY   BlockCoPy Menu

All          All tests in sequence
Default      Default test sequence
Quick        Quick test sequence
MM           Memory to Memory
CM           Cache to Memory
MC           Memory to Cache
CC           Cache to Cache
DISplay      DISplay error log


Command ==>
```

The options on the Blockcopy Menu are designed to verify the correct operation of the blockcopy command, a function code 4 control space command in Sun-3 architectures. The blockcopy is equivalent to a blockmove instruction which moves sixteen bytes of data from source to destination. The destination can only be main memory. Any blockcopy command that attempts to write to the central cache should invalidate the cache block and write directly to main memory.

A

The *All Tests in Sequence* option on the Blockcopy Menu executes a sequence of all of the tests on that menu.

D

The *Default Test Sequence* option on the Blockcopy Menu executes a sequence of all tests on the menu.

Q

The *Quick Test Sequence* option on the Blockcopy Menu executes a sequence of all tests on the menu.

MM

The *Memory to Memory* option on the Blockcopy Menu verifies the correct blockcopy operation for memory-to-memory operands. With cache blocks invalid, this test moves a block of data from source memory to destination memory, leaving the cache blocks unchanged.

CM

The *Cache to Memory* option on the Blockcopy Menu tests the blockcopy operation for cache-to-memory operands. This test verifies that a blockcopy read that is a cache hit first reads the block from the cache, leaving the cache block valid and unchanged, then writes the block to main memory successfully.

MC

The *Memory to Cache* option on the Blockcopy Menu tests the blockcopy operation for memory-to-cache operands. This test verifies that moving a block of data from main memory to another block of main memory that falls in the cache first invalidates the cache block, then transfers the data to the main memory destination.

CC

The *Cache to Cache* option on the Blockcopy Menu tests the blockcopy operation for cache-to-cache operands. This test performs a blockcopy read operation on data in the cache then writes the same cache block to main memory which also resides in the cache. The cache block should be invalidated (the destination for a blockcopy write can only be main memory, not the cache), and the data should be written successfully to main memory.

DIS

The *Display Error Log* option on the Blockcopy Menu displays the Central Cache Diagnostic error log.

Options Menu

When you choose O from the Main Menu, the Options Menu is displayed:

```
Central Cache Diagnostic   Rev. X.X   MM/DD/YY   Options Menu

Pass=        Pass limit                              [value]
SQPass=      SeQuence Pass limit                     [value]
SCope=       SCopeloop on error                      [value]

SOft=        SOft error retry limit                  [value]
STop=        STop on nth error                       [value]
WAit=        WAit on error for message viewing       [value]

INFO=        INFOrmation message level               [value]
STATUS=      STATUS message level                    [value]
MORE=        set MORE menu display option            [value]

Def          set Default values
DISplay      DISplay error log


Command ==>
```

As discussed at the beginning of this chapter, the choices on the Options Menu allow you to set environmental parameters that control how the Central Cache

Diagnostic tests perform. All of the tests execute according to the current values of these parameters. The current values of the local environmental parameters are shown in the column on the right side of the Options Menu.

You can specify different values for the environmental parameters by entering new values on the command line when you issue a specific command (in effect for that command only) or by choosing options from the Options Menu and entering new values.

P

The *Pass Limit* option on the Options Menu sets the pass count limit. This limit allows a test to be run until the number of iterations exceeds the pass count limit or the `st=` limit is exceeded first.

Parameter	*Description*	*Default*
`Pass`	Pass count.	1

SQP

The *Sequence Pass Limit* option on the Options Menu determines the number of times that each test runs in succession within a single test sequence. The number of times that the sequence executes is determined by the value of `Pass=`. For example, assume that a menu contains only Test A, Test B, and an All Test sequence. If `SQPass`=2 and `Pass`=3, each test is executed twice in succession within each All Test sequence, and the sequence is executed three times:

Test A, Test A, Test B, Test B
Test A, Test A, Test B, Test B
Test A, Test A, Test B, Test B

Parameter	*Description*	*Default*
`SQPass`	Pass count for successive execution of tests within a sequence.	1

SC

The *Set Scopeloop on Error* option on the Options Menu sets the scope loop on error flag to the value specified. The scope loop on error function displays a message indicating the type of operation, the address, and the data used while looping. The DIAGNOSTIC bit in the Enable Register is set to 1 before the operation and set to 0 after the operation while looping. This resetting is intended to be used as a SYNC trigger for scope/analyzer debugging. To keep the loop as tight as possible, no messages are displayed while looping is in progress.

Parameter	*Description*	*Default*
`SCope`	Scope on error. 0 (off) 1 (on)	0

SO

The *Soft Error Retry Limit* option on the Options Menu sets the count limit for soft error retries. Tests for which this limit is applicable are retried when an error occurs until the number of retries exceeds the soft error retry limit.

Parameter	*Description*	*Default*
`SOft`	Soft error retry limit.	0

ST

The *Stop on Nth Error* option on the Options Menu causes testing to stop when the number of errors exceeds the *n*th error limit specified. A value of 0 causes testing to continue when an error is encountered.

Parameter	*Description*	*Default*
`STop`	Stop on error count.	0

WA

The *Wait on Error for Message Viewing* option on the Options Menu determines whether or not testing is interrupted when an error is encountered. This option, when set, causes testing to stop temporarily for 30 to 60 seconds when an error occurs so that messages on the screen may be read before testing continues or before the menu is redisplayed. A value of 0 causes testing to continue without interruption when an error is encountered.

Parameter	*Description*	*Default*
`WAit`	Set wait flag. 0 (off) 1 (on)	0

INFO

The *Information Message Level* option on the Options Menu sets the level of verbosity of informational messages to the value entered.

Parameter	*Description*	*Default*
INFO	Set informational message level. 0 (off) 1 (terse) 2 (verbose 3 (more verbose)	1

STATUS

The *Status Message Level* option on the Options Menu sets the level of verbosity of status messages to the value entered.

Parameter	*Description*	*Default*
STATUS	Set status message level. 0 (off) 1 (terse) 2 (verbose 3 (more verbose)	1

MORE

The *Set More Menu Display* option on the Options Menu enables or disables the `more` feature. This feature causes the diagnostic to display a screenful of messages then pause for your input. To display the next screen of information, press [RETURN].

Parameter	*Description*	*Default*
MORE	Enable or disable `more` command. DISable (off) ENAble (on)	DISable

D

The *Set Default Values* option on the Options Menu resets all environmental variables to their default values.

DIS

The *Display Error Log* option on the Options Menu displays the Central Cache Diagnostic error log.

Display Error Log

When you choose **DIS** from the Main Menu, the *Display Error Log* option allows you to display the Central Cache Diagnostic error log.

3.10. Messages

The Central Cache Diagnostic displays messages at the beginning and end of each test and test sequence. As a test completes, a status message is displayed, indicating whether the test passed or failed. In the case of failure, the message is both displayed and written to the error log. You can view messages associated with all tests that failed by choosing the Display Error Log option from any menu.

You control the amount of information displayed in the test messages by specifying the level of verbosity (1–3) of the `INFO` and `STATUS` environmental parameters. The higher the level, the more information displayed about an error. If you want to see only whether a test passed or failed, set this level to 1. For detailed information about a failure, such as the memory address where a failure occurred or the data expected and received at a location under test, set the level to 3.

3.11. Glossary

Blockcopy	Memory-to-memory block moves.
"Cacheable" Page	A memory page that is not marked "non-cacheable."
Cache Block	Sixteen bytes of data in the central cache.
Cache Bypass	An access in which the cache is ignored.
Cache Enable/Disable	Instructions that turn the central cache on or off.
Cache Hit	An access in which the data is in the cache.
Cache Miss	An access in which the data is not in the cache.
Cache Tags	Control/status bits which represent the status of a block in the central cache.
Central Cache	A small block of fast memory between the microprocessor and main memory used to hold the most recently used data and instructions. This temporary storage reduces the number of accesses to slower, main memory.
CPU	Central Processing Unit. In this chapter, CPU refers to the Sun-3/400 series board containing the Motorola MC68030 microprocessor chip.
Ethernet	A communication link between systems, using coaxial cable.
Modified Cache Block	A cache block containing data that has been modified (modified bit = 1).
Valid Cache Block	A cache block containing valid data (valid bit = 1).

Writeback

The cache writes back a cache block to main memory if it is marked valid and modified when a cache miss on that block occurs.

4

SPARCsystem 330 Cache Diagnostic

4

SPARCsystem 330 Cache Diagnostic

4.1. General Description

The SPARCsystem 330 Cache consists of 128 Kbytes of high-speed Data RAM (32-bit words), 32 Kbytes of high-speed Tag RAM (32-bit words), and a cache controller.

Data RAM, which is organized as cache lines or blocks of four words, is used to hold a copy of the most recently used blocks of main memory. Tag RAM holds the control, status, and address bits for the cache lines. Each Tag RAM location corresponds to a four-word cache line of Data RAM. If a cache line holds a copy of four words in main memory, the corresponding tag is called a *valid* tag.

The cache controller chip controls the interfaces between Tag and Data RAM and the rest of the system, including the memory controller, the DVMA gate array, the I/O gate array, the IU, (Integer Unit), and the virtual buses.

means that writing to a location in memory that has its copy in the cache updates both the main memory and the copy in the cache at the same time. It also means that a cache line is allocated only on a read access, not on a write access.

4.2. What This Chapter Contains

Following an overview of the diagnostic and a list of required hardware, the SPARCsystem 330 Cache Diagnostic Main Menu and submenus are discussed. All menu options are described. The end of the chapter contains a list of error messages, arranged by test, and a glossary.

4.3. Overview of the Diagnostic

The SPARCsystem 330 Cache Diagnostic runs under the Exec and conforms to the interface standards of the Exec. All, Default, and Quick test sequences are provided. The diagnostic generates and logs error messages for later retrieval. Online help is also available.

The SPARCsystem 330 Cache Diagnostic tests the following areas of the cache subsystem:

- Data RAM — used to hold the cached data. This memory can be accessed as 32-bit words in ASI=2, starting from location 0x90000000.
- Tag RAM — used to hold address and control/status bits of the cached data. This memory can be accessed as 32-bit words in ASI=2, starting from location 0x80000000.
- Reading — allocation of a cache line, bringing the data to the cache, and updating the data and tag bits on a read access.

- Writing — updating capability of the cache subsystem on a write access.
- Flushing — ability of the cache subsystem to invalidate a cache line selectively.

4.4. Hardware Requirements

In order to run the SPARCsystem 330 Cache Diagnostic, the system under test must have the following:

- A SPARCsystem 330 CPU board
- A cardcage, power supply, and related cables
- A monitor
- A keyboard
- A boot device (local disk, local tape, or remote disk over Ethernet)

4.5. User Interface

The user interface of the SPARCsystem 330 Cache Diagnostic adheres to the menu standards of the Exec. A Main Menu and submenus are provided. Each option may be selected from a menu by typing the letter or letters displayed in upper case in the column on the left side of the menu.

The submenus provide access to the individual tests. Each submenu tests a functional area of the cache subsystem. A separate submenu provides tools for selective debugging.

To return to the Main Menu from a submenu, press [ESC] then press [RETURN]. To exit the diagnostic and return to the Exec, press [ESC] when the Main Menu is displayed.

To display online Help for a menu option, enter the following on the command line:

```
? option_name
```

4.6. Starting the Diagnostic

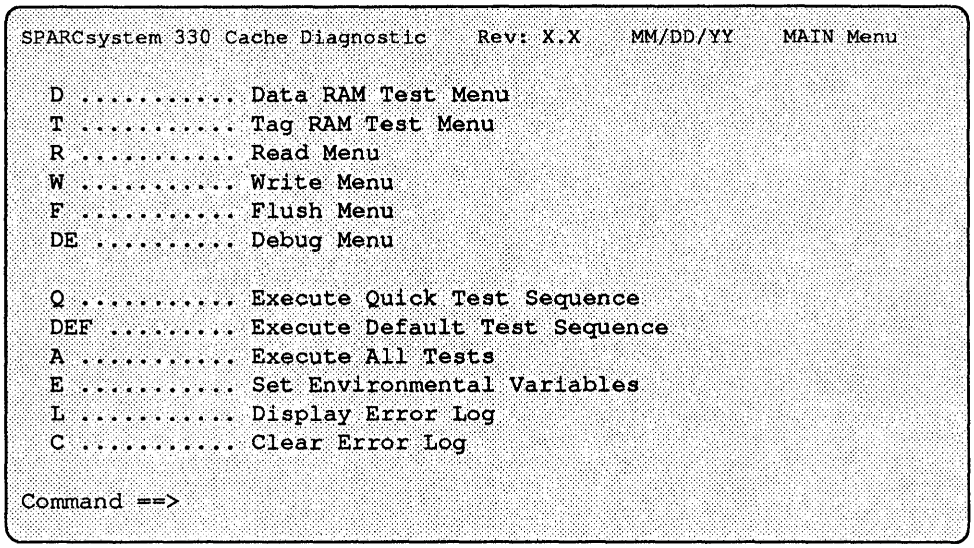
For information on starting the Exec, refer to Chapter 2, "Using the SunDiagnostic Executive." After you have started the Exec, choose the SPARCsystem 330 Cache Diagnostic from the Diagnostics Menu.

4.7. The Diagnostic Menus

This section of the chapter provides a modular description of the SPARCsystem 330 Cache Diagnostic, beginning with the Main Menu and working down through the options available on each of the submenus. A list of messages generated by each test in the diagnostic is given in the section entitled "Error Messages."

Main Menu

The FDC Main Menu, which displays when you start the FDC Diagnostic, provides access to the submenus of the individual tests:



```
SPARCsystem 330 Cache Diagnostic    Rev: X.X    MM/DD/YY    MAIN Menu

  D ........... Data RAM Test Menu
  T ........... Tag RAM Test Menu
  R ........... Read Menu
  W ........... Write Menu
  F ........... Flush Menu
  DE .......... Debug Menu

  Q ........... Execute Quick Test Sequence
  DEF ......... Execute Default Test Sequence
  A ........... Execute All Tests
  E ........... Set Environmental Variables
  L ........... Display Error Log
  C ........... Clear Error Log

Command ==>
```

Parameters

Several of the options on the Main Menu take additional parameters that you may specfy on the command line. The parameters, together with their descriptions and default values, are shown in the following table:

Parameter	*Description*	*Default*
`Info`	Informational message level. More detailed information (1). Less detailed information (0).	0
`PAss`	Number of times a test is executed. If specified with the All option, all tests on the menu are executed this number of times.	1
`Repeat`	Number of times a test is executed. This parameter overrides the value of `pa=`.	1
`SError`	Stop on *n*th error.	???
`STATus`	Display and logging of status messages. Status messages displayed and logged (1). Status messages neither displayed nor logged (0).	1

Data RAM Test Menu

D

The *Data RAM Test Menu* option on the Main Menu provides access to the tests on the Data RAM Test Menu. These tests are described in detail later in this chapter.

Tag RAM Test Menu

T

The *Tag RAM Test Menu* option on the Main Menu provides access to the tests on the Tag RAM Test Menu, described later in this chapter.

Read Menu

R

The *Read Menu* option on the Main Menu provides access to the tests on the Read Menu, described later in this chapter.

Write Menu

W

The *Write Menu* option on the Main Menu provides access to the tests on the Write Menu, described later in this chapter.

Flush Menu

F

The *Flush Menu* option on the Main Menu provides access to the tests on the Flush Menu, described later in this chapter.

Debug Menu

DE

The *Debug Menu* option on the Main Menu provides access to the debugging tools on the Debug Menu, described later in this chapter.

Execute Quick Test Sequence

Q *r=*

The *Execute Quick Test Sequence* option on the Main Menu executes a brief series of tests which affords a basic level of confidence in the cache subsystem.

Execute Default Test Sequence

DEF *r=*

The *Execute Default Test Sequence* option on the Main Menu executes all of the tests on all of the submenus, excluding any time-consuming tests. This sequence affords a high level of confidence in the cache subsystem.

Execute All Tests

A *r=*

The *Execute All Tests* option on the Main Menu executes all of the tests, in the order listed on the submenus.

Set Environmental Variables

E *i= se= pa= stat=*

The *Set Environmental Variables* option on the Main Menu sets or displays the environmental variables. If you enter this command with no parameter specified on the command line, the current settings of the environmental variables are displayed.

Display Error Log

L

The *Display Error Log* option on the Main Menu displays all of the messages entered into the error log by the diagnostic.

Clear Error Log

C

The *Clear Error Log* option on the Main Menu removes all messages from the error log.

Data RAM Test Menu

When you choose **D** from the Main Menu, the Data RAM Test Menu is displayed:

```
SPARCsystem 330 Cache Diagnostic  Rev: X.X  MM/DD/YY  DATA RAM Test Menu

   W .......... Data RAM Write/Write Next Pattern Test
   AD ......... Data RAM Address Test
   P .......... Data RAM 3-Pattern Test
   WA ......... Data RAM Walking Ones Test
   M .......... Data RAM March Test
   A .......... Execute All Cache Data Tests

Command ==>
```

The options on the Data RAM Test Menu provide access to the tests available for verifying the 128 Kbytes of Data RAM. In addition to the specific tests, an All test sequence is provided for comprehensive testing.

You may specify a parameter on the command line with any of the options on the menu. A description of the parameter and its default are shown in the following table:

Parameter	*Description*	*Default*
`Repeat`	Number of times a test is executed. This parameter overrides the value of `pa=`.	1

Data RAM Write/Write Next Pattern Test

W *r=*

The *Data RAM Write/Write Next Pattern Test* option on the Data RAM Test Menu provides a test of the write/read data integrity of the Data RAM. The test writes a pattern to each address in the Data RAM space. It then inverts the pattern and writes it to the next address. Lastly, the original address is read back and compared to the noninverted pattern.

Data RAM Address Test

AD *r=*

The *Data RAM Address Test* option on the Data RAM Test Menu verifies the address uniqueness of the cache Data RAM space. The test writes the address of each location as data to that location, over the entire Data RAM space. It then reads back every location and compares the contents of the location to its address.

Data RAM 3-Pattern Test

P *r=*

The *Data RAM 3-Pattern Test* option on the Data RAM Test Menu performs three write and compare operations to the entire Data RAM. On the first pass, the pattern 0xA5972C5A, 0x5AA5972C, 0x2C5AA597 is repeated throughout the cache data space, then the entire Data RAM is read and compared. On the second pass, the following pattern is used: 0x5AA5972C, 0x2C5AA597, 0x972C5AA5. On the third pass, this pattern is used: 0x2C5AA597, 0xA5972C5A, 0x5AA5972C.

Data RAM Walking Ones Test

WA *r=*

The *Data RAM Walking Ones Test* option on the Data RAM Test Menu writes the entire Data RAM space with a bit-rotating data pattern of ones in a field of zeros. The test writes each address with a pattern then writes the complement of the pattern to the next address. Finally the contents of the original address are read back and compared to the noninverted pattern.

Data RAM March Test

M *r=*

The *Data RAM March Test* option on the Data RAM Test Menu writes the entire Data RAM space with hexadecimal zeros from lowest to highest address. The zero pattern is read, and 0xFFFFFFFF pattern is written from highest to lowest address. Finally the 0xFFFFFFFF pattern is read, and zeros are written from lowest to highest address.

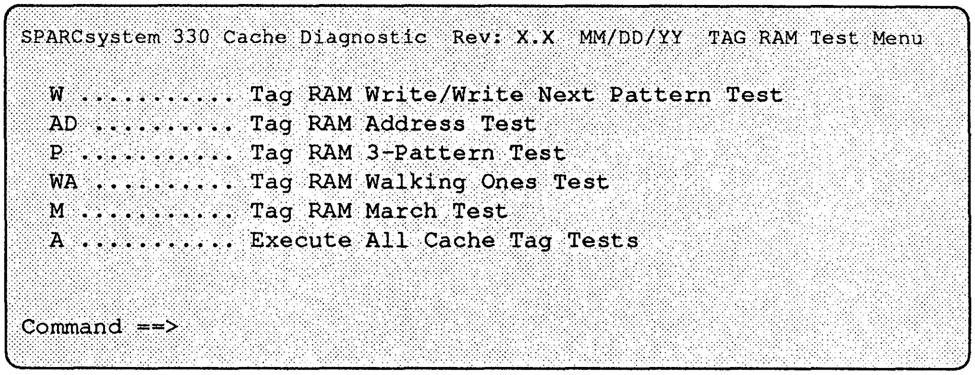
Execute All Cache Data Tests

A *r=*

The *Execute All Cache Data Tests* option on the Data RAM Test Menu executes all of the tests on the menu in order.

Tag RAM Test Menu

When you choose **T** from the Main Menu, the Tag RAM Test Menu is displayed:

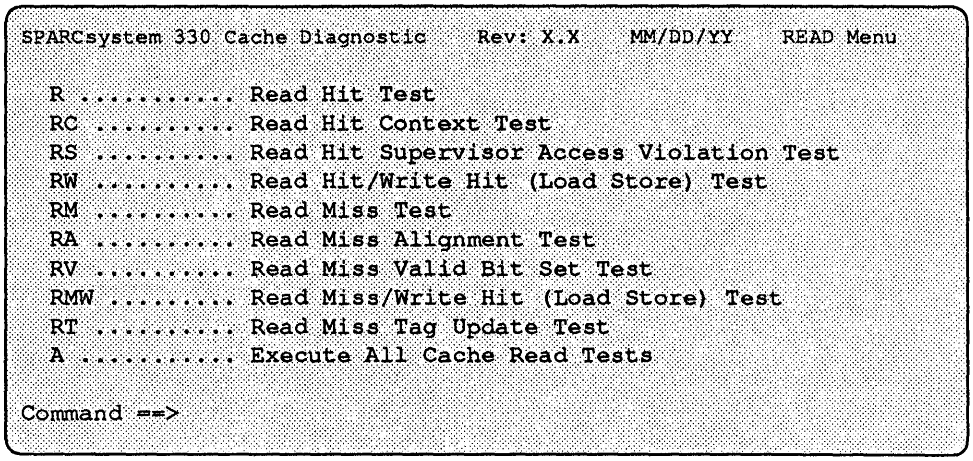


```
SPARCsystem 330 Cache Diagnostic  Rev: X.X  MM/DD/YY  TAG RAM Test Menu

  W .......... Tag RAM Write/Write Next Pattern Test
  AD ......... Tag RAM Address Test
  P .......... Tag RAM 3-Pattern Test
  WA ......... Tag RAM Walking Ones Test
  M .......... Tag RAM March Test
  A .......... Execute All Cache Tag Tests


Command ==>
```

The options on the Tag RAM Test Menu perform the same tests on the 32 Kbytes of Tag RAM that the options on the Data RAM Test Menu, just described, perform on the Data RAM. These options also take the *r=* parameter on the command line.

Read Menu

When you choose **R** from the Main Menu, the Read Menu is displayed:



```
SPARCsystem 330 Cache Diagnostic    Rev: X.X    MM/DD/YY    READ Menu

   R .......... Read Hit Test
   RC ......... Read Hit Context Test
   RS ......... Read Hit Supervisor Access Violation Test
   RW ......... Read Hit/Write Hit (Load Store) Test
   RM ......... Read Miss Test
   RA ......... Read Miss Alignment Test
   RV ......... Read Miss Valid Bit Set Test
   RMW ........ Read Miss/Write Hit (Load Store) Test
   RT ......... Read Miss Tag Update Test
   A .......... Execute All Cache Read Tests

Command ==>
```

The options on the Read Menu allow you to validate read operations on the cache, both when the cache line does and does not contain data.

You may specify a parameter on the command line with any of the options on the menu. A description of the parameter and its default are shown in the following table:

Parameter	*Description*	*Default*
`Repeat`	Number of times a test is executed. This parameter overrides the value of `pa=`.	1

Read Hit Test

R *r=*

The *Read Hit Test* option on the Read Menu verifies that reading from a location in main memory that has its copy in the cache results in data being read from Data RAM, not from main memory. The test performs the following steps for each cache line in the cache space:

1. Initializes 16 bytes of main memory, starting from address X, by writing the inverse of the address of each location to that location.
2. Initializes a cache line corresponding to address X by writing the address of each Data RAM location to that location.
3. Sets the Tag entry corresponding to address X to valid, writable, in context 0.
4. Turns the cache ON.
5. Reads four words, starting from location X.
6. Turns the cache OFF.
7. Verifies that the data read are the same as the contents of Data RAM (read hit), not the contents of main memory, and that the Data RAM remains unchanged.

Read Hit Context Test

RC *r=*

The *Read Hit Context Test* option on the Read Menu verifies that if a block of memory mapped as supervisor-only access is in the cache, reading this block in supervisor mode causes a cache read hit, regardless of the context in which it is being read. The test executes the following steps for all contexts (0–15).

1. Initializes 16 bytes of main memory, starting from address X, by writing the inverse of the address of each location to that location.
2. Initializes a cache line corresponding to address X by writing the address of each Data RAM location to that location.
3. Sets the Tag entry corresponding to address X to valid, writable, supervisor-only, in one of the 16 contexts.
4. Turns the cache ON.
5. Reads four words in supervisor mode, context 0, starting from location X.
6. Turns the cache OFF.
7. Verifies that the data read are from Data RAM (read hit), not from main memory, and that the Data RAM and Tag RAM remain unchanged.

Read Hit Supervisor Access Violation Test

RS *r=*

The *Read Hit Supervisor Access Violation Test* option on the Read Menu verifies that, while in User Mode, an attempt to access a location that is in the cache and mapped as supervisor-only access generates a bus error exception and causes the access to be denied. The test performs the following steps:

1. Initializes the main memory under test by writing the inverse of the address of each location to that location.
2. Maps a page of memory as supervisor-only access.
3. Turns the cache ON.
4. Reads a word, to allocate a cache line and bring data to the cache.
5. Turns the cache OFF.
6. Verifies that Tag RAM contains the correct bits.
7. Turns the cache ON.
8. Accesses the supervisor-only area in User Mode, which should cause a bus error.
9. Turns the cache OFF.
10. Verifies that the bus error exception has occurred and that the protection bit in the Bus Error Register is set.

Read Hit/Write Hit (Load Store) Test

RW *r=*

The *Read Hit/Write Hit (Load Store) Test* option on the Read Menu verifies the execution of the load store instruction. The load store instruction, `LDSTUB`, includes both a read and a write operation. This test verifies the execution of `LDSTUB` when the read operation is a cache hit and the write operation is a cache hit. The test performs the following steps for each byte of a cache line:

1. Sets main memory corresponding to a cache line to an incrementing pattern (1–16).
2. Sets Data RAM locations to a decrementing pattern (16–1) for the cache line being tested.
3. Sets Tag RAM to valid, writable, in context 0.
4. Turns the cache ON.
5. Performs a load store operation.
6. Turns the cache OFF.
7. Verifies that the data read is the same as the initialized value of Data RAM, that the main memory and the corresponding Data RAM are set to 0xFF, and that no other location in the cache line is affected.

Read Miss Test

RM *r=*

The *Read Miss Test* option on the Read Menu verifies that if a location in memory does not have its contents in the cache, reading that location will allocate a cache line and bring 16 bytes of memory, starting from that location, into the cache. The tag corresponding to that location should also be updated. The test executes the following steps for each cache line in the cache space:

1. Initializes 16 bytes of main memory, starting from address X, by writing the inverse of the address of each location to that location.
2. For the cache line corresponding to address X, writes the address of each Data RAM location to that location.
3. Sets the Tag entry corresponding to address X to invalid, writable, in context 15.
4. Turns the cache ON.
5. Reads a word from location X.
6. Turns the cache OFF.
7. Verifies that the Tag entry is valid and that it contains the correct address bits. The Data RAM and data read during the read access are verified to contain the contents of the initialized memory.

Read Miss Alignment Test

RA *r=*

The *Read Miss Alignment Test* option on the Read Menu verifies byte, half-word, and doubleword alignment within a cache line, on a read miss. The test executes the following steps for each byte, halfword, and doubleword:

1. Sets 16 bytes of memory, starting from address X, to an incrementing pattern (1–16).
2. For the cache line corresponding to address X, writes the address of each Data RAM location to that location.
3. Sets the Tag entry corresponding to address X to invalid, writable, in context 0.
4. Turns the cache ON.
5. Reads 16 bytes, 8 halfwords, or two doublewords, starting from location X.
6. Turns the cache OFF.
7. Verifies that the Tag entry is valid and that it contains the correct address bits. The Data RAM and data read during the read access are verified to contain the incrementing pattern.

Read Miss Valid Bit Set Test

RV *r=*

The *Read Miss Valid Bit Set Test* option on the Read Menu verifies that if location X has its contents in the cache and its corresponding tag entry set to valid, then a read access to a location Y whose address is equal to the address of location X plus 128 Kbytes causes a read miss and brings the contents of location Y into the cache. The test executes the following steps:

1. Sets 16 bytes of memory, starting from address X, to a known pattern.
2. Sets 16 bytes of memory, starting from address Y (Y=X+128 Kbytes), to another pattern.
3. For the cache line corresponding to address X, writes the address of each Data RAM location to that location.
4. Sets the Tag entry corresponding to address X to valid, writable, in context 0, with address bits corresponding to address X.
5. Turns the cache ON.
6. Reads four 32-bit words, starting from address Y.
7. Turns the cache OFF.
8. Verifies that the Tag entry is valid and that it contains the address bits for address Y. The Data RAM and data read during the read access are verified to contain the contents of address Y.

Read Miss/Write Hit (Load Store) Test

RMW *r=*

The *Read Miss/Write Hit (Load Store) Test* option on the Read Menu verifies the execution of the load store instruction. The load store instruction, `LDSTUB`, includes both a read and a write operation. This test verifies the execution of `LDSTUB` when the read operation is a cache miss and the write operation is a cache hit. The test performs the following steps for each byte of a cache line:

1. Sets the main memory corresponding to a cache line to an incrementing pattern (1–16).
2. Sets Data RAM corresponding to the cache line to 0.
3. Sets Tag RAM associated with the cache line to invalid, writable, in context 0.
4. Turns the cache ON.
5. Performs a load store operation.
6. Turns the cache OFF.
7. Verifies that the data read is the same as the data in main memory, that the accessed byte in main memory is set to 0xFF, that the Data RAM corresponding to that byte is set to 0xFF, and that no other location in the cache line is affected.

Read Miss Tag Update Test

RT *r=*

The *Read Miss Tag Update Test* option on the Read Menu checks the updating of Tag RAM bits on a read miss. The test performs the following steps for each possible 128 Kbyte block of virtual memory space:

1. Sets Tag RAM associated with the area being tested to invalid, writable, in context 10.
2. Maps the memory to be tested as readable, writable, cacheable, supervisor-only access.
3. Turns the cache ON.
4. Reads a byte for each cache line in the cache space, causing a read miss.
5. Turns the cache OFF.
6. Verifies that all tags are valid and contain the correct bit pattern.

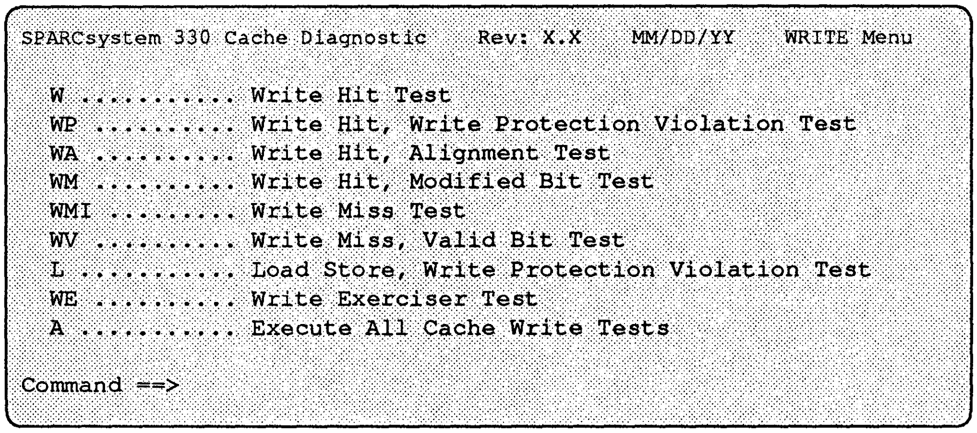
Execute All Cache Read Tests

A *r=*

The *Execute All Cache Read Tests* option on the Read Menu executes all of the tests on the menu in order.

Write Menu

When you choose **W** from the Main Menu, the Write Menu is displayed:



```
SPARCsystem 330 Cache Diagnostic    Rev: X.X    MM/DD/YY    WRITE Menu

  W ........... Write Hit Test
  WP .......... Write Hit, Write Protection Violation Test
  WA .......... Write Hit, Alignment Test
  WM .......... Write Hit, Modified Bit Test
  WMI ......... Write Miss Test
  WV .......... Write Miss, Valid Bit Test
  L ........... Load Store, Write Protection Violation Test
  WE .......... Write Exerciser Test
  A ........... Execute All Cache Write Tests

Command ==>
```

The options on the Write Menu allow you to check the validity of the cache system by performing write operations.

You may specify a parameter on the command line with any of the options on the menu. A description of the parameter and its default are shown in the following table:

Parameter	*Description*	*Default*
`Repeat`	Number of times a test is executed. This parameter overrides the value of `pa=`.	1

Write Hit Test

W *r=*

The *Write Hit Test* option on the Write Menu verifies that writing to a location in main memory that has its copy in the cache will update the main memory and its corresponding location in the Data RAM. The test performs the following steps for each cache line in the cache space:

1. Initializes 16 bytes of main memory, starting from address X, by writing the inverse of the address of each location to that location.
2. Initializes a cache line corresponding to address X by writing the address of each Data RAM location to that location.
3. Sets the Tag entry corresponding to address X to valid, writable, in context 0.
4. Turns the cache ON.
5. Writes a known pattern to four locations, starting from location X.
6. Turns the cache OFF.
7. Verifies that the contents of main memory and of Data RAM are the same as the pattern that was written, and that the tag for the cache line remains unchanged.

Write Hit, Write Protection Violation Test

WP *r=*

The *Write Hit, Write Protection Violation Test* option on the Write Menu verifies that attempting to write to a location that is mapped as write-protected and that is in the cache with the write-allowed (WA) bit of Tag RAM set to 0 will generate a bus error exception and cause the access to be denied. The test performs the following steps:

1. Initializes the main memory under test by writing the inverse of the address of each location to that location.
2. Maps a page of memory as read-only (write protected).
3. Sets the Tag RAM corresponding to the write-protected page to valid, in context 0.
4. Initializes the Data RAM corresponding to the write-protected page by writing the address of each Data RAM location to that location.
5. Turns the cache ON.
6. Reads a word from the read-only page of memory, to allocate a cache line and bring data to the cache.
7. Turns the cache OFF.
8. Verifies that Tag RAM is valid and contains the correct bits.
9. Turns the cache ON.
10. Attempts to write to a write-protected location, which should generate a bus error.
11. Turns the cache OFF.
12. Verifies that the bus error exception has occurred and that the protection bit in the Bus Error Register is set.

Write Hit, Alignment Test

WA *r=*

The *Write Hit, Alignment Test* option on the Write Menu verifies byte, halfword, and doubleword alignment within a cache line, on a read/write hit. The test executes the following steps for each byte, halfword, and doubleword:

1. Sets 16 bytes of memory, starting from address X, to 0.
2. For the cache line corresponding to address X, sets Data RAM to 0.
3. Sets the Tag entry corresponding to address X to valid, writable, in context 0.
4. Turns the cache ON.
5. Writes an incrementing pattern (1–16) to 16 bytes, 8 halfwords, or two doublewords, starting from location X.
6. Turns the cache OFF.

7. Verifies that the Tag RAM is valid and unchanged and that memory and Data RAM have been updated and contain the incrementing pattern.

Write Hit, Modified Bit Test

WM *r*=

The *Write Hit, Modified Bit Test* option on the Write Menu verifies that writing to a location in main memory that has its copy in the cache and the modified bit (M bit) of the MMU's page map entry for the location set to 0 will update main memory, its corresponding location in the Data RAM, and the MMU's M-bit in the corresponding page map entry. The test performs the following steps for each page in the cache:

1. Initializes 128 Kbytes of main memory, starting from address X, by writing the inverse of the address of each location to that location.
2. Initializes The Data RAM by writing the address of each Data RAM location to that location.
3. Sets the Tag RAM to valid, writable, in context 0.
4. Turns the cache ON.
5. Writes a known pattern to the first location of each page of data that is in the cache.
6. Turns the cache OFF.
7. Verifies that the contents of main memory and of the Data RAM are the same as the pattern that was written, that the tag for the cache line is valid and remains unchanged, and that the M-bit of the MMU's page map entries that were accessed are set to 1.

Write Miss Test

WMI *r*=

The *Write Miss Test* option on the Write Menu verifies that writing to a location in main memory that is not in the cache updates only the main memory and does not allocate a cache line for the location that was accessed. The test executes the following steps for each cache line in the cache space:

1. Initializes 16 bytes of main memory, starting from address X, by writing the inverse of the address of each location to that location.
2. Initializes the cache line corresponding to address X by writing the address of each Data RAM location to that location.
3. Sets all tags to 0.
4. Turns the cache ON.
5. For the cache line, writes the address of each location as data to that location, starting from location X.
6. Turns the cache OFF.
7. Verifies that the data was written to main memory and that the Tag RAM still contains 0.

Write Miss, Valid Bit Test

WV *r=*

The *Write Miss, Valid Bit Test* option on the Write Menu verifies that if location X has its contents in the cache and its corresponding tag entry set to valid, then a write access to a location Y whose address is equal to the address of location X plus 128 Kbytes causes a write miss, writes the data to location Y, and invalidates the cache line. The test executes the following steps:

1. Sets 16 bytes of memory, starting from address X, to a known pattern.
2. Sets 16 bytes of memory, starting from address Y (Y=X+128 Kbytes), to another pattern.
3. For the cache line corresponding to address X, writes the address of each Data RAM location to that location.
4. Sets the Tag entry corresponding to address X to valid, writable, in context 0, with address bits corresponding to address X.
5. Turns the cache ON.
6. For four locations, starting from location Y, writes the address of each location as data to that location.
7. Turns the cache OFF.
8. Verifies that the Tag entry is invalid, that the contents of main memory for location Y are updated, and that the contents of main memory for location X remain unchanged.

Load Store, Write Protection Violation Test

L *r=*

The *Load Store, Write Protection Violation Test* option on the Write Menu verifies that an attempt to execute a load store instruction, when the read portion of the instruction is a cache hit, the write portion is a cache hit, and the location being accessed is mapped as read-only, will generate a bus error exception and cause the access to be denied. The test performs the following steps for each byte of a cache line:

1. Initializes 16 bytes of main memory by writing an incrementing pattern (1–16).
2. Maps a page of memory as read-only (write-protected).
3. Sets the Tag RAM corresponding to the mapped memory to valid, read-only (WA bit = 0), in context 0.
4. Sets four locations of Data RAM corresponding to the initialized memory to a decrementing pattern (16–1).
5. Turns the cache ON.
6. Reads a word from the read-only page of memory, to allocate a cache line and bring data to the cache.

7. Turns the cache OFF.

8. Verifies that Tag RAM is valid and contains the correct bits.

9. Turns the cache ON.

10. Attempts a load store operation, which should cause a bus error.

11. Turns the cache OFF.

12. Verifies the following:
a) The bus error exception has occurred.
b) The protection bit in the bus error register is set to 1.
c) The main memory, Data RAM, and Tag RAM remain unchanged.
d) The read portion of the load store operation was denied.

Write Exerciser Test

`WE` *r=*

The *Write Exerciser Test* option on the Write Menu verifies consecutive write hits without any read access of memory for data or instruction. The routine that performs the consecutive write hits resides in the cache while the routine is being executed. The test performs the following steps for a block of data:

1. Initializes 128 Kbytes of cacheable memory by writing the inverse of the address of each location to that location.

2. Sets all the tags in the Tag RAM space to 0.

3. Copies the WRITE routine to the cacheable area of memory.

4. Turns the cache ON.

5. Brings the initialized area of memory, including the WRITE routine, into the cache.

6. Turns the cache OFF.

7. Clears the 128 Kbytes of initialized memory to 0.

8. Turns the cache ON.

9. Copies a block of memory from a cacheable area of memory to a non-cacheable area, then back to a different cacheable area. The copying includes the following sequence:
a) Four doubleword reads followed by four doubleword writes.
b) Four word reads followed by four word writes.
c) Eight halfword reads followed by eight halfword writes.
d) Eight byte reads followed by eight byte writes.
e) Eight byte reads followed by eight byte writes.

10. Turns the cache OFF.

11. Verifies that the memory and Data RAM were updated properly.

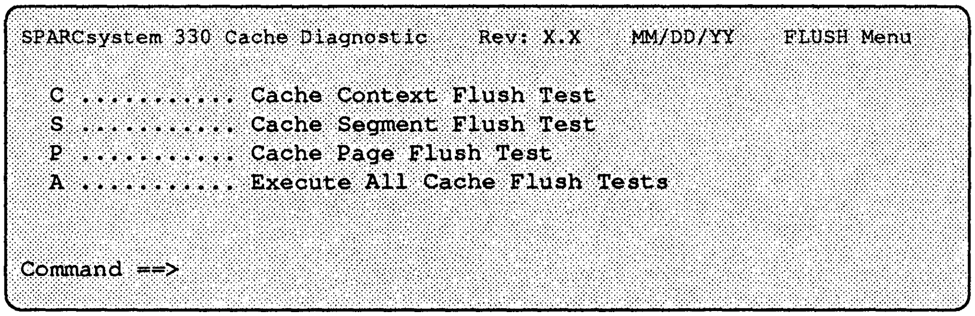
Execute All Cache Write Tests

A *r=*

The *Execute All Cache Write Tests* option on the Write Menu executes all of the tests on the menu in order.

Flush Menu

When you choose **F** from the Main Menu, the Flush Menu is displayed:



```
SPARCsystem 330 Cache Diagnostic    Rev: X.X    MM/DD/YY    FLUSH Menu

   C .......... Cache Context Flush Test
   S .......... Cache Segment Flush Test
   P .......... Cache Page Flush Test
   A .......... Execute All Cache Flush Tests


Command ==>
```

The options on the Flush Menu allow you to verify the functioning of the cache system during a page, segment, and context flush.

You may specify a parameter on the command line with any of the options on the menu. A description of the parameter and its default are shown in the following table:

Parameter	*Description*	*Default*
`Repeat`	Number of times a test is executed. This parameter overrides the value of `pa=`.	1

Cache Context Flush Test

C *r=*

The *Cache Context Flush Test* option on the Flush Menu verifies that on a context flush, only the tag entries corresponding to the flushed context are invalidated. The test performs the following steps for each context (1–15):

1. Sets the first half of the tags in the Tag RAM space to valid, context 0.
2. Sets the second half of the tags in the Tag RAM space to valid, a different context.
3. Turns the cache ON.
4. Performs a context flush in context 0.
5. Turns the cache OFF.
6. Verifies that only the tags corresponding to context 0 are invalidated and that the rest of the tags remain unchanged.

Cache Segment Flush Test

s *r=*

The *Cache Segment Flush Test* option on the Flush Menu verifies that on a segment flush, only the tag entries corresponding to the flushed segment are invalidated, and if the segment of memory is mapped as supervisor-only access, the segment is flushed, regardless of the context in which the segment flush is performed. The test performs the following steps:

1. Sets the first half of the tags in the Tag RAM space to valid, segment 0, context 8, supervisor-only access.
2. Sets the second half of the tags in the Tag RAM space to valid, segment 1, context 8, supervisor-only access.
3. Turns the cache ON.
4. Performs a segment flush in context 0, for segment 0.
5. Turns the cache OFF.
6. Verifies that only the tags corresponding to segment 0 are invalidated and that the rest of the tags remain unchanged.
7. Sets the first half of the tags in the Tag RAM space to valid, segment 0, context 0.
8. Sets the second half of the tags in the Tag RAM space to valid, context 0, segments 0–0xFFF consecutively (tag X set to segment X, tag X+1 set to segment X+1, and so on).
9. Turns the cache ON.
10. Performs a segment flush in context 0, for segment 0.
11. Turns the cache OFF.
12. Verifies that only the tags corresponding to segment 0 are invalidated and that the rest of the tags remain unchanged.

Cache Page Flush Test

p *r=*

The *Cache Page Flush Test* option on the Flush Menu verifies that on a page flush, only the tag entries corresponding to the flushed page are invalidated, and if the page of memory is mapped as supervisor-only access, the page is flushed, regardless of the context in which the page flush is performed. The test performs the following steps:

1. Sets the tag entries corresponding to page 0 in a segment to valid, segment 0, supervisor-only access, context 2.
2. Sets the tag entries corresponding to pages 1–15 to valid, context 0.
3. Turns the cache ON.
4. Performs a page flush in context 0, for page 0.
5. Turns the cache OFF.

6. Verifies that only the tags corresponding to page 0 are invalidated and that the rest of the tags remain unchanged.
7. Turns the cache ON.
8. Performs a segment flush in context 0, for page 1.
9. Turns the cache OFF.
10. Verifies that only the tags corresponding to page 1 are invalidated and that the rest of the tags remain unchanged.

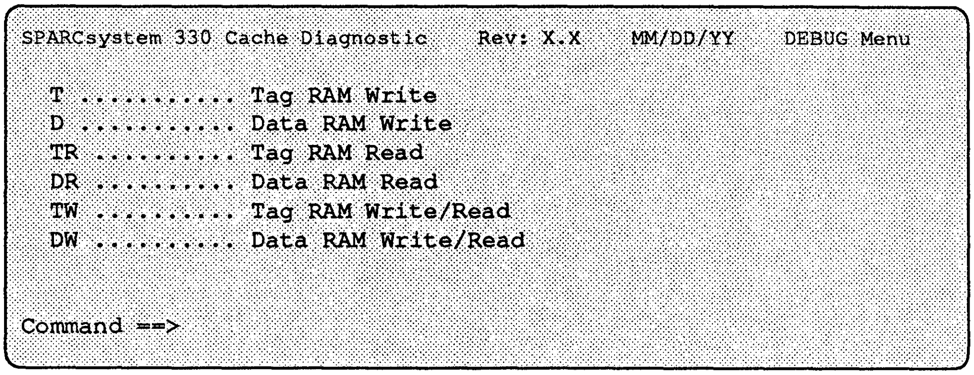
Execute All Cache Flush Tests

A *r=*

The *Execute All Cache Flush Tests* option on the Flush Menu executes all of the tests on the menu in order.

Debug Menu

When you choose **DE** from the Main Menu, the Debug Menu is displayed:



```
SPARCsystem 330 Cache Diagnostic    Rev: X.X    MM/DD/YY    DEBUG Menu

  T .......... Tag RAM Write
  D .......... Data RAM Write
  TR ......... Tag RAM Read
  DR ......... Data RAM Read
  TW ......... Tag RAM Write/Read
  DW ......... Data RAM Write/Read


Command ==>
```

The options on the Debug Menu allow you to initiate scope loop write and read accesses of the Data and Tag RAM, for selective debugging purposes. All of the options execute continuously; to interrupt an option, press any key.

You may specify additional parameters on the command line. The parameters, together with their descriptions and default values, are shown in the following table:

Parameter	*Description*	*Default*
`Count`	The number of times a location or range of locations is to be accessed before you can interrupt by pressing a key.	1000
`End`	The address in Data or Tag RAM at which writing or reading stops. This address is not accessed.	4 (Data RAM: location 0x90000004 in ASI=2) (Tag RAM: location 0x80000004 in ASI=2)
`Pattern`	The data pattern to be written.	0
`Start`	The first address in Data or Tag RAM to which to write or from which to read.	0 (Data RAM: location 0x90000000 in ASI=2) (Tag RAM: location 0x80000000 in ASI=2)

Tag RAM Write

T *s= e= p= c=*

The *Tag RAM Write* option on the Debug Menu executes a continuous scope loop write access of the Tag RAM, from the specified starting address to the specified ending address. The data pattern specified by p= is written the number of times specified by `c=`. If you do not press a key to interrupt, the looping continues.

Data RAM Write

D *s= e= p= c=*

The *Data RAM Write* option on the Debug Menu executes a continuous scope loop write access of the Data RAM, from the specified starting address to the specified ending address. The data pattern specified by p= is written the number of times specified by `c=`. If you do not press a key to interrupt, the looping continues.

Tag RAM Read

T *s= e= c=*

The *Tag RAM Read* option on the Debug Menu executes a continuous scope loop read access of the Tag RAM, from the specified starting address to the specified ending address. The read loop is executed the number of times specified by `c=`. If you do not press a key to interrupt, the looping continues.

Data RAM Read

DR *s= e= c=*

The *Data RAM Read* option on the Debug Menu executes a continuous scope loop read access of the Data RAM, from the specified starting address to the specified ending address. The read loop is executed the number of times specified by c=. If you do not press a key to interrupt, the looping continues.

Tag RAM Write/Read

TW *s= e= p= c=*

The *Tag RAM Write/Read* option on the Debug Menu executes a scope loop write followed by a read access of the Tag RAM. It writes then reads back a location or range of locations from the specified starting address to the specified ending address. The data pattern specified by p= is written then read the number of times specified by c=. If you do not press a key to interrupt, the looping continues.

Data RAM Write/Read

DW *s= e= p= c=*

The *Data RAM Write/Read* option on the Debug Menu executes a scope loop write followed by a read access of the Data RAM. It writes then reads back a location or range of locations from the specified starting address to the specified ending address. The data pattern specified by p= is written then read the number of times specified by c=. If you do not press a key to interrupt, the looping continues.

4.8. Error Messages

Main Menu Error Messages

The following error messages are displayed when an error is encountered before the main menu is displayed:

```
Can Not Test The Cache. Wrong machine Type.

ERROR : Boot command line error.
 Usage : cache4.sa cache=dis

Unable to allocate memory as READABLE, WRITABLE, CACHEABLE
```

Data RAM Test Menu Error Messages

The following error messages are displayed when an error is encountered while testing the Data RAM:

Data RAM Write/Write Next Pattern Test

```
Error (D1): Compare error
 addr=00000000 exp=00000000 obs=00000000 XOR=00000000.
```

Data RAM Address Test

```
Error (D2): Compare error
 addr=00000000 exp=00000000 obs=00000000 XOR=00000000.
```

Data RAM 3-Pattern Test

```
Error (D3): Compare error
 addr=00000000 exp=00000000 obs=00000000 XOR=00000000.
```

Data RAM Walking Ones Test

```
Error (D4): Compare error
 addr=00000000 exp=00000000 obs=00000000 XOR=00000000.
```

Data RAM March Test

```
Error (D5): Compare error
 addr=00000000 exp=00000000 obs=00000000 XOR=00000000.
```

Tag RAM Test Menu Error Messages

The following error messages are displayed when an error is encountered while testing the Tag RAM:

Tag RAM Write/Write Next Pattern Test

```
Error (T1): Compare error
 addr=00000000 exp=00000000 obs=00000000 XOR=00000000.
```

Tag RAM Address Test

```
Error (T2): Compare error
 addr=00000000 exp=00000000 obs=00000000 XOR=00000000.
```

Tag RAM 3-Pattern Test

```
Error (T3): Compare error
```

```
 addr=00000000 exp=00000000 obs=00000000 XOR=00000000.
```

Tag RAM Walking Ones Test

```
Error (T4): Compare error
 addr=00000000 exp=00000000 obs=00000000 XOR=00000000.
```

Tag RAM March Test

```
Error (T5): Compare error
 addr=00000000 exp=00000000 obs=00000000 XOR=00000000.
```

Read Menu Error Messages

The following error messages are displayed when an error is encountered while running the Read tests:

Read Hit Test

```
Error (R1): In supervisor mode, Read Hit data,
       different from content of DATA RAM.
 Addr=00000000 Exp=00000000 Obs=00000000

Error (R1): In user mode, Read Hit data,
       different from content of DATA RAM.
 Addr=00000000 Exp=00000000 Obs=00000000

Error (R1): Unexpected Data RAM update.
 Addr=00000000 Exp=00000000 Obs=00000000
```

Read Hit Context Test

```
Error (R2): Read Hit data, different from content of DATA RAM.
 Addr=00000000 Exp=00000000 Obs=00000000

Error (R2): On Read Hit, Tag RAM updated.
 Addr=00000000 Exp=00000000 Obs=00000000

Error (R2): Unexpected Data RAM update.
 Addr=00000000 Exp=00000000 Obs=00000000
```

Read Hit Supervisor Access Violation Test

```
Error (R3):
 TAG bits incorrectly set.
 addr=00000000 exp=00000000  obs=00000000, xor=00000000

Error (R3): Access to Supervisor-access-only data
       in user mode, did not cause a trap.
 addr=00000000 obs=00000000

Error (R3):
 Protection Error bit, in BUS ERROR register is not set.
       bus errorreg=00
```

Read Hit/Write Hit (Load Store) Test

```
Error (R4): On a load_store Hit,
       Read Hit data, is different from content of DATA RAM.
 addr=00000000 exp=00000000  obs=00000000

Error (R4): On a load_store Hit,
               main memory not updated properly.
 addr=00000000 exp=00000000  obs=00000000

Error (R4): On a load_store Hit, more than one location in
       memory was updated.
 Exp Addr=00000000, Obs Addr=00000000
```

Read Miss Test

```
Error (R5): After clearing TAG Valid bit, it is still set.
 addr=00000000 obs=00000000

Error (R5): On a Read Miss, TAG bits incorrectly updated.
 Addr=00000000 Exp=00000000 Obs=00000000

Error (R5): On a Read Miss, DATA RAM incorrectly updated.
 Addr=00000000 Exp=00000000 Obs=00000000

Error (R5): Read Miss data,
                  different from content of main memory.
 Addr=00000000 Exp=00000000 Obs=00000000
```

Read Miss Alignment Test

```
Error (R6): On a Read Miss, TAG bits incorrectly updated.
 Addr=00000000 obs=00000000

Error (R6): Read Miss data,
          different from content of main memory.
 Addr=00000000 Exp=00000000 Obs=00000000

Error (R6): On a Read Miss, DATA RAM incorrectly updated.
 addr=00000000 exp=00000000 obs=00000000
```

Read Miss Valid Bit Set Test

```
Error (R7): After setting TAG Valid bit, it is still 0.
 addr=00000000 obs=00000000

Error (R7): On a Read Miss, TAG bits incorrectly updated.
 Addr=00000000 obs=00000000

Error (R7): On a Read Miss, DATA RAM incorrectly updated.
 addr=00000000 exp=00000000 obs=00000000

Error (R7): Read Miss data,
         different from content of main memory.
 Addr=00000000 Exp=00000000 Obs=00000000
```

Read Miss/Write Hit (Load Store) Test

```
Error (R8): On a load_store Miss,
       data read, is different from content of Memory.
 addr=00000000 exp=00000000  obs=00000000

Error (R8): On a load_store Miss,
              main memory incorrectly updated.
 addr=00000000 exp=00000000  obs=00000000

Error (R8): On a load_store Miss,
             DATA RAM incorrectly updated.
 addr=00000000 exp=00000000  obs=00000000
```

Read Miss Tag Update Test

```
Error (R9):
 Unable to allocate memory as READABLE, WRITABLE, CACHEABLE
 addr=00000000

Error (R9):
 Unable to deallocate memory
 addr=00000000

Error (R9):
 TAG bits incorrectly set.
 addr=00000000 exp=00000000  obs=00000000, xor=00000000
```

Write Menu Error Messages

The following error messages are displayed when an error is encountered while running the Write tests:

Write Hit Test

```
Error (W1): On a Write Hit, TAG bits updated.
 Addr=00000000 Exp=00000000  Obs=00000000 XOR=00000000

Error (W1): On a Write Hit, DATA RAM incorrectly updated.
 Addr=00000000 Exp=00000000  Obs=00000000

Error (W1): On a Write Hit, main memory incorrectly updated.
 Addr=00000000 Exp=00000000  Obs=00000000
```

Write Hit, Write-Protect Violation Test

```
Error (W2): Tag bits incorrectly updated.
 Addr=00000000 Exp=00000000 Obs=00000000, Xor=00000000

Error (W2): Writing to read-only memory, did not cause a trap.

Error (W2): Protection Error bit,
    in BUS ERROR register is not set. bus errorreg=00

Error (W2): On a Write Hit access to a Read-only area of memory,
       Data RAM was modified.
 Addr=00000000 Exp=00000000 Obs=00000000

Error (W2): On a Write Hit access to a Read-only area of memory,
       Memory was modified.
 Addr=00000000 Exp=00000000 Obs=00000000
```

Write Hit, Alignment Test

```
Error (W3): On a Write Hit, TAG bits updated.
 Addr=00000000 Exp=00000000  Obs=00000000

Error (W3): On a Write Hit, Data RAM incorrectly updated.
 Addr=00000000 Exp=00000000  Obs=00000000

Error (W3): On a Write Hit, main memory incorrectly updated.
 Addr=00000000 Exp=00000000  Obs=00000000
```

Write Hit, Modified Bit Test

```
Error (W4): On a Write Hit, TAG Valid bit not set.
 addr=00000000 obs=00000000

Error (W4): On a Write Hit, DATA RAM incorrectly updated.
 Addr=00000000 Exp=00000000 Obs=00000000

Error (W4): On a Write Hit, main memory incorrectly updated.
 Addr=00000000 Exp=00000000 Obs=00000000

Error (W4): On a Write Hit, MMU's Modified bit is not set.
 Addr=00000000 Page map entry =000000000
```

Write Miss Test

```
Error (W5): On a Write Miss, TAG Valid bit is set.
 Addr=00000000 Data=00000000

Error (W5): On a Write Miss, main memory incorrectly updated.
 Addr=00000000 Exp=00000000  Obs=00000000
```

Write Miss, Valid Bit Test

```
Error (W6): On a Write Miss, TAG Valid bit is set.
 Addr=00000000 Obs=00000000

Error (W6): On a Write Miss, main memory not updated correctly.
 Addr=00000000 Exp=00000000  Obs=00000000

Error (W6): On a Write Miss,
              data written to an invalid address.
 Exp Addr=00000000 Obs Addr=00000000
```

Load Store, Write Protection Violation Test

```
Error (W7): Unexpected Read Hit Occurred.
 Addr=00000000 Exp=00000000 Obs=00000000

Error (W7): Unexpected Memory Write.
 Addr=00000000 Exp=00000000 Obs=00000000

Error (W7): Unexpected Data RAM update.
 Addr=00000000 Exp=00000000 Obs=00000000
```

Write Exerciser Test

```
Error (W8): Read Hit data is different from content of Data RAM.
 Addr=00000000 Exp=00000000 Obs=00000000

Error (W8): On a Write Hit, main memory incorrectly updated.
 Addr=00000000 Exp=00000000 Obs=00000000

Error (W8): On a Write Hit, Data RAM incorrectly updated.
 Addr=00000000 Exp=00000000  Obs=00000000
```

Flush Menu Error Messages

The following error messages are displayed when an error is encountered while running the Flush tests:

Cache Context Flush Test

```
Error (F1): After Flushing, Tag Valid Bit Still Set.
 Addr=00000000 Data=00000000

Error (F1): After Flushing, Tag Valid Bit Incorrectly cleared.
 Addr=00000000 Data=00000000

Error (F1): Tag Valid Bit is 0, After Being Set to 1.
 Addr=00000000 Data=00000000
```

Cache Segment Flush Test

```
Error (F2): Tag Valid Bit is 0, After Being Set to 1.
 Addr=00000000 Data=00000000

Error (F2): After Flushing, Tag Valid Bit Still Set.
 Addr=00000000 Data=00000000

Error (F2): After Flushing, Tag Valid Bit Incorrectly cleared.
 Addr=00000000 Data=00000000
```

Cache Page Flush Test

```
Error (F3): After Flushing, Tag Valid Bit Still Set.
 Addr=00000000 Data=00000000

Error (F3): After Flushing, Tag Valid Bit Incorrectly cleared.
 Addr=00000000 Data=00000000
```

4.9. Glossary

Cache	The combination of cache data, cache tags, and their associated logic.
Cache Block (Line)	Four words (16 bytes) of memory that are associated with a tag.
Cache Data	High-speed RAM that holds a copy of portions of main memory.
Cache Tags	High-speed RAM that holds control and status bits unique to each cache block.
CPU	Central Processing Unit.
Doubleword	Eight consecutive bytes.
Flushing	Invalidating cache blocks (lines).
Halfword	Two consecutive bytes.
MMU	Memory Management Unit.
Page	The smallest contiguous block of memory that can be mapped by the MMU.
RAM	Random Access Memory.
Segment	Thirty-two pages of memory.
Word	Four consecutive bytes.

5

I/O Cache Diagnostic

5.1. General Description

The I/O Cache Diagnostic is provided to test the I/O cache on the CPU boards of Sun-3/400 series systems. The I/O cache resides between the I/O mapper and system memory. It provides a cache line for each 8 Kbyte pages of system DVMA space. The cache line is four long words, or 128 bits.

5.2. What This Chapter Contains

Following an overview of the I/O Cache Diagnostic and a list of required hardware, the Main Menu and its submenus are discussed. Optional parameters are described. The end of the chapter contains a list of error messages and a glossary.

5.3. Overview of the Diagnostic

The I/O cache is designed to buffer DVMA data transfers to and from the VMEbus. If the I/O cache is enabled, data is transferred to and from the cache as follows:

> The system DVMA address space for the VMEbus is the highest Mbyte of virtual address space (0xFFF00000–0xFFFFFFFF). In equivalent 8Kbyte pages, this represents 128 pages. In the I/O cache, a one-to-one hardware mapping is provided between each of the 128 system DVMA page addresses and each of the 128 I/O cache block addresses. This means that there is one I/O cache line (four long words) per system DVMA page which caches all DVMA writes and reads within that page. Through the I/O mapper, which resides between the system DVMA bus and the I/O cache, each page may be mapped selectively as cached or not cached.

The I/O Cache Diagnostic makes use of a VME loopback mode provided in the CPU hardware. When this mode is enabled, system DVMA writes and reads can be executed, permitting the I/O cache and I/O mapper to be tested.

In order to verify working SRAM, the I/O cache SRAM is tested first before the functional I/O cache tests are performed. Special emphasis is placed upon checking the consistency of data between the I/O cache and the central cache.

5.4. Hardware Requirements

The following hardware is required to run the I/O Cache Diagnostic:

- □ A Sun-3/400 series system with verified functional memory board
- □ A monitor
- □ A keyboard

- A boot device (local disk, local tape, or remote disk over Ethernet)

5.5. Firmware Requirements

The standard power-up boot PROM is required to run the I/O Cache Diagnostic. Execution of the power-up selftest is necessary to verify the functionality of main memory before the I/O Cache Diagnostic is run.

NOTE *To verify consistency between the caches, the Central Cache Diagnostic program should be run before running the I/O Cache Diagnostic.*

5.6. User Interface

From the Main Menu of the I/O Cache Diagnostic, you can run All, Quick, or Default test sequences, or display the submenus for selection of individual tests. This system permits direct access to specific tests, as well as the capability to perform extensive sequential testing on the entire I/O cache.

Each option may be selected from a menu by typing the letter or letters displayed in uppercase in the column on the left side of the menu.

Additional parameters and options may be specified on the command line. If a parameter is not specified on the command line, the default value is used.

To display online Help for any menu option, enter the following on the command line:

```
? option_name
```

5.7. Starting the Diagnostic

For information on starting the Exec, refer to Chapter 2, "Using the SunDiagnostic Executive." Any caches that can cache data must be disabled before you run the I/O Cache Diagnostic. You must disable the caches when booting the Exec. For example, if you were using a Sun-4 system and booting over Ethernet, you would enter a command similar to this:

```
bie() /stand/exec4 cache=dis
```

If you want to run the I/O Cache Diagnostic and you are already in the Exec, exit from the Exec and reboot, disabling the cache with a command similar to the one shown above.

After you have started the Exec, choose the I/O Cache Diagnostic from the Diagnostics Menu.

5.8. The Diagnostic Menus

This section of the chapter provides a modular description of the I/O Cache Diagnostic. It discusses each of the options available on the Main Menu and submenus and describes the parameters that may be used.

Main Menu

The Main Menu, which displays when you start the I/O Cache Diagnostic, provides access to the individual tests and to the All, Quick, and Default test sequences:

```
Sun3/400 Series I/O Cache Diagnostic   Rev. X.X   MM/DD/YY
Main Menu

TR        Tag RAM Memory Tests Menu
DR        Data RAM Memory Tests Menu
NC        Noncached Hits/Misses Tests Menu
CA        Cached Hits/Misses Menu
FL        Flush Tests Menu
OP        Options Menu
All       Execute all of the above Tests
Quick     Quick Test Sequence
Default   Default Test Sequence

Command==>
```

Optional Parameters

As specified in the following sections, the tests in the I/O Cache Diagnostic take optional parameters which you may enter on the command line. If a parameter value is not specified, the default value is used. The meanings of the parameters are shown below.

Parameter	*Description*
pa	Pass count
sa	Hex start address
si	Size in bytes
da	Pattern

Tag RAM Memory Tests Menu

TR

When you choose the *Tag RAM Memory Tests Menu* option from the Main Menu, this submenu displays:

```
Sun3/400 Series I/O Cache Diagnostic   Rev. X.X   MM/DD/YY
Tag RAM Tests Menu

PT        Pattern Test
AT        Address Test
CT        Checker Test
NT        NTA Test
All       Execute all of the above Tests

Command==>
```

The Tag RAM Memory Tests Menu provides a selection of memory pattern tests designed to test the I/O cache tag SRAM.

PT *pa= sa= si= da=*

The *Pattern Test* option on the Tag RAM Memory Tests Menu writes the pattern specified for the number of passes specified, beginning at the start address and extending to the specified size.

AT *pa= sa= si=*

The *Address Test* option on the Tag RAM Memory Tests Menu writes an address pattern for the number of passes specified, beginning at the start address and extending to the specified size.

CT *pa= sa= si=*

The *Checker Test* option on the Tag RAM Memory Tests Menu writes alternating patterns of all zeros and all ones for the number of passes specified, beginning at the start address and extending to the specified size.

NT *pa= sa= si=*

The *NTA Test* option on the Tag RAM Memory Tests Menu performs three operations in the following order:

1. Writes all zeros for each address from the start address to the size specified.
2. For each address from the start address *plus* the size, reads zeros and writes ones.
3. For each address from the start address to the start address plus the specified size, reads ones and writes zeros.

This three-part test is repeated for the specified number of passes.

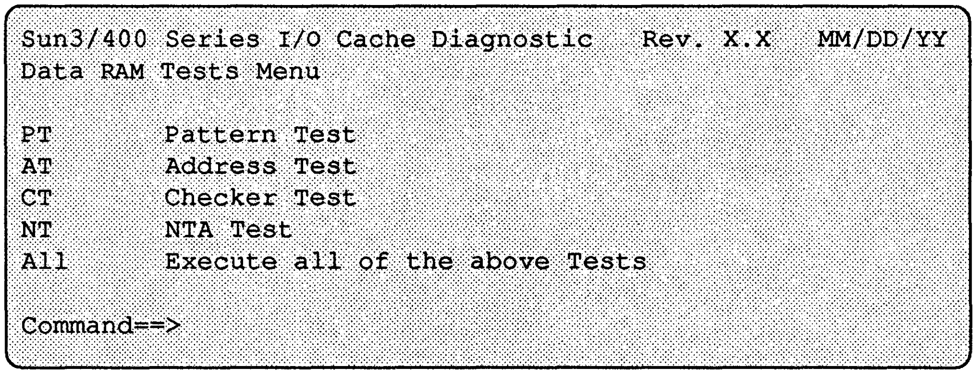
A

The *Execute All of the Above Tests* option on the Tag RAM Memory Tests Menu executes all of the tests, in the order that they appear on the menu.

Data RAM Memory Tests Menu

DR

When you choose the *Data RAM Memory Tests Menu* option from the Main Menu, this submenu displays:



```
Sun3/400 Series I/O Cache Diagnostic   Rev. X.X   MM/DD/YY
Data RAM Tests Menu

PT        Pattern Test
AT        Address Test
CT        Checker Test
NT        NTA Test
All       Execute all of the above Tests

Command==>
```

The options on the Data RAM Memory Tests Menu perform the same tests on the I/O cache data SRAM that the tests on the Tag RAM Memory Tests Menu perform on the I/O cache tag SRAM.

See the description of the Tag RAM Memory Tests for more information.

Noncached Hits/Misses Tests Menu

NC

When you choose the *Noncached Hits/Misses Tests Menu* option from the Main Menu, this submenu displays:

```
Sun3/400 Series I/O Cache Diagnostic   Rev X.X   MM/DD/YY
Noncached Hits/Misses Menu

RM      Read Non-IOCable page, CPU Cache Miss
RH      Read Non-IOCable page, CPU Cache Hit
WM      Write Non-IOCable page, CPU Cache Miss
WH      Write Non-IOCable page, CPU Cache Hit
All     Execute all of the above Tests

Command==>
```

The Noncached Hits/Misses Tests Menu provides a selection of read and write tests for cases in which data is not cached in the I/O cache but either is or is not cached in the central cache.

RM *pa= sa= si=*

The *Read Non-IOCable page, CPU Cache Miss* option on the Noncached Hits/Misses Tests Menu tests the case in which the I/O cache line's page is marked no-cache in the I/O mapper and is not cached in the central cache. In this case, the I/O cache controller should read the data from system memory, not from the central cache.

RH *pa= sa= si=*

The *Read Non-IOCable page, CPU Cache Hit* option on the Noncached Hits/Misses Tests Menu tests the case in which the I/O cache line's page is marked no-cache in the I/O mapper and is cached in the central cache. In this case, the I/O cache controller should read the data from the central cache, not from the system memory.

WM *pa= sa= si=*

The *Write Non-IOCable page, CPU Cache Miss* option on the Noncached Hits/Misses Tests Menu tests the case in which the I/O cache line's page is marked no-cache in the I/O mapper and is not cached in the central cache. In this case, the I/O cache controller should write the data in main memory, and invalidate the central cache line.

WH *pa= sa= si=*

The *Write Non-IOCable page, CPU Cache Hit* option on the Noncached Hits/Misses Tests Menu tests the case in which the I/O cache line's page is marked no-cache in the I/O mapper and is cached in the central cache. In this case, the I/O cache controller should write the data to main memory, and invalidate the central cache line.

A

The *Execute All of the Above Tests* option on the Noncached Hits/Misses Tests Menu executes all of the tests, in the order that they appear on the menu.

Cached Hits/Misses Tests Menu

CA

When you choose the *Cached Hits/Misses Tests Menu* option from the Main Menu, this submenu displays:

```
Sun3/400 Series I/O Cache Diagnostic   Rev X.X   MM/DD/YY
Cached Hits/Misses Menu

RHT      Read Hit
RMM      Read Miss and CPU Cache Miss
RMH      Read Miss and CPU Cache Hit
RWM      Read Miss with WB and CPU Cache Miss
RWH      Read Miss with WB and CPU Cache Hit
WHT      Write Hit
WHM      First Write Hit and CPU Cache Miss
WHH      First Write Hit and CPU Cache Hit
WMM      Write Miss and CPU Cache Miss
WMH      Write Miss and CPU Cache Hit
WWM      Write Miss with WB and CPU Cache Miss
WWH      Write Miss with WB and CPU Cache Hit
All      Execute all of the above Tests

Command==>
```

The Cached Hits/Misses Tests Menu provides a selection of read and write tests for cases in which data is marked "cacheable" in the I/O mapper and is cached or not in the I/O cache and cached or not in the central cache. These tests also determine whether or not any cached data is modified.

RHT *pa= sa= si=*

The *Read Hit* test on the Cached Hits/Misses Tests Menu verifies that a system DVMA read operation that is an I/O cache hit reads the data from the I/O cache, not from main memory.

The steps in the test are performed in the following order:

1. The I/O cache tags are set to valid, and not modified.
2. A pattern of 0x2's is stored in the I/O cache line.
3. The central cache line for the address is set to valid, not modified.
4. A pattern of 0x1's is stored in the central cache line.
5. A pattern of 0x0's is stored in the memory line.
6. The I/O cache only is enabled, not the central cache.

7. A DVMA long read operation is performed.

8. The read data is confirmed to have come from the I/O cache.

9. The I/O cache is verified as still valid, not modified.

10. The I/O cache line data is verified as unchanged.

11. The central cache line is verified as unchanged.

RMM *pa= sa= si=*

The *Read Miss and CPU Cache Miss* test on the Cached Hits/Misses Tests Menu verifies that a system DVMA read operation in which the data is not in the I/O cache nor in the central cache reads the data from main memory, not from the I/O cache or central cache.

The steps in thc test are performed in the following order:

1. The I/O cache tags are set to valid, and not modified.

2. A pattern of 0x2's is stored in the I/O cache line.

3. The central cache line for the address is set to valid, not modified.

4. A pattern of 0x1's is stored in the central cache line.

5. A pattern of 0x2's is stored in the memory line.

6. The I/O cache only is enabled, not the central cache.

7. A DVMA long read operation is performed.

8. The read data is confirmed to have come from main memory.

9. The I/O cache is verified as still valid, not modified.

10. The I/O cache line data is verified as unchanged.

11. The central cache line is verified as unchanged.

RMH *pa= sa= si=*

The *Read Miss and CPU Cache Hit* test on the Cached Hits/Misses Tests Menu verifies that a system DVMA read operation in which the data is not in the I/O cache but is in the central cache reads the data from the central cache, not from the I/O cache or from main memory.

The steps in the test are performed in the following order:

1. The I/O cache tags are set to valid, and not modified.

2. A pattern of 0x2's is stored in the I/O cache line.

3. The central cache line for the address is set to valid, not modified.

4. A pattern of 0x1's is stored in the central cache line.

5. A pattern of 0x4's is stored in the memory line.

6. The I/O cache only is enabled, not the central cache.

7. A DVMA long read operation is performed.
8. The read data is confirmed to have come from the central cache.
9. The I/O cache is verified as still valid, not modified.
10. The I/O cache line data is verified to be 0x1's from the central cache.
11. The central cache line is verified as unchanged.
12. The central cache line is verified as still containing a pattern of 0x1's.

RWM *pa= sa= si=*

The *Read Miss with WB and CPU Cache Miss* test on the Cached Hits/Misses Tests Menu verifies that, in the case of a DVMA read operation with modified data in the I/O cache, the data is written back to main memory and the I/O cache is updated with the new data.

The steps in the test are performed in the following order:

1. The I/O cache tags are set to valid, and modified.
2. A pattern of 0x2's is stored in the I/O cache line.
3. The central cache line for the address is set to valid, not modified.
4. A pattern of 0x1's is stored in the central cache line.
5. A pattern of 0x4's is stored in the memory line.
6. The I/O cache only is enabled, not the central cache.
7. A DVMA long read operation is performed.
8. The read data is confirmed to have come from main memory.
9. The I/O cache is verified as still valid, not modified.
10. The I/O cache line data is verified to be 0x4's from main memory.
11. The central cache line is verified to be valid and not modified.
12. The central cache line is verified as still containing a pattern of 0x1's.
13. The old I/O cache line data is verified to have been written back to main memory.

RWH *pa= sa= si=*

The *Read Miss with WB and CPU Cache Hit* test on the Cached Hits/Misses Tests Menu verifies that, in the case of a DVMA read operation with the I/O cache line modified and the new data in the central cache, the old data is written back to main memory and is updated from the central cache with the new data read.

The steps in the test are performed in the following order:

1. The I/O cache tags are set to valid, and modified.
2. A pattern of 0x2's is stored in the I/O cache line.

3. The central cache line for the address is set to valid, not modified.
4. A pattern of 0x1's is stored in the central cache line.
5. A pattern of 0x0's is stored in the memory line.
6. The I/O cache only is enabled, not the central cache.
7. A DVMA long read operation is performed.
8. The read data is confirmed to have come from the central cache.
9. The I/O cache is verified as still valid, not modified.
10. The I/O cache line data is verified to be 0x1's from main memory.
11. The central cache line is verified to be valid and not modified.
12. The central cache line is verified as still containing a pattern of 0x1's.
13. The old I/O cache line data is verified to have been written back to main memory.

WHT *pa= sa= si=*

The *Write Hit* test on the Cached Hits/Misses Tests Menu verifies that, in the case of a DVMA write operation with the data in the I/O cache, the data is read from the I/O cache, not from main memory.

The steps in the test are performed in the following order:

1. The I/O cache tags are set to valid, and modified.
2. A pattern of 0x2's is stored in the I/O cache line.
3. The central cache line for the address is set to valid, not modified.
4. A pattern of 0x4's is stored in the central cache line.
5. A pattern of 0x0's is stored in the memory line.
6. The central cache line is set to not valid, and not modified.
7. A DVMA write operation, writing a word of 0x7's, is performed.
8. The I/O cache line data is verified to be written with a word of 0x7's.
9. The I/O cache line is verified to be set to valid, and modified.
10. The central cache line is verified to be not valid, and not modified.

WHM *pa= sa= si=*

The *First Write Hit and CPU Cache Miss* test on the Cached Hits/Misses Tests Menu verifies that, when the data in the I/O cache is valid and not modified, a DVMA write operation updates the I/O cache data but does not change the central cache data or main memory.

The steps in the test are performed in the following order:

1. The I/O cache tags are set to valid, and not modified.

2. A pattern of 0x2's is stored in the I/O cache line.
3. The central cache line for the address is set to valid, not modified.
4. A pattern of 0x4's is stored in the central cache line.
5. A pattern of 0x0's is stored in the memory line.
6. A DVMA write operation, writing a word of 0x7's, is performed.
7. The I/O cache line data is verified to be written with a word of 0x7's.
8. The I/O cache line is verified to be set to valid, and modified.
9. The central cache line is verified to be valid, and not modified.

WHH *pa= sa= si=*
The *First Write Hit and CPU Cache Hit* test on the Cached Hits/Misses Tests Menu verifies that, when the I/O cache is valid and not modified, a DVMA write operation updates the I/O cache but does not affect the central cache or main memory.

The steps in the test are performed in the following order:

1. The I/O cache tags are set to valid, and not modified.
2. A pattern of 0x2's is stored in the I/O cache line data.
3. The central cache line for the address is set to valid, and not modified.
4. A pattern of 0x4's is stored in the central cache line.
5. A pattern of 0x0's is stored in the memory line.
6. A DVMA write operation, writing a word of 0x7's, is performed.
7. The I/O cache line data is verified to be written with a word of 0x7's.
8. The I/O cache line is verified to be set to valid, and modified.
9. The central cache line is verified to be valid, and not modified.

WMM *pa= sa= si=*
The *Write Miss and CPU Cache Miss* test on the Cached Hits/Misses Tests Menu verifies that, when the data is not in the I/O cache or in the central cache, a DVMA write operation writes to the I/O cache but does not affect the central cache or main memory.

The steps in the test are performed in the following order:

1. The I/O cache tags are set to valid, and not modified.
2. A pattern of 0x2's is stored in the I/O cache line.
3. The central cache line for the address is set to valid, not modified.
4. The central cache line tag address is set to a different modulo 64 Kbyte address.

5. A pattern of 0x4's is stored in the central cache line.
6. A pattern of 0x0's is stored in the memory line.
7. A DVMA write operation, writing a word of 0x7's, is performed.
8. The I/O cache line data is verified to be written with a word of 0x7's.
9. The I/O cache line is verified to be set to valid, and modified.
10. The central cache line is verified to be valid, and not modified.

WMH *pa= sa= si=*

The *Write Miss and CPU Cache Hit* test on the Cached Hits/Misses Tests Menu verifies that, when the data is not in the I/O cache but is in the central cache, a DVMA write operation updates the I/O cache line, and the central cache line is invalidated.

The steps in the test are performed in the following order:

11. The I/O cache tags are set to valid, and not modified.
12. A pattern of 0x2's is stored in the I/O cache line.
13. The central cache line for the address is set to valid, not modified.
14. A pattern of 0x4's is stored in the central cache line.
15. A pattern of 0x0's is stored in the memory line.
16. A DVMA write operation, writing a word of 0x7's, is performed.
17. The I/O cache line data is verified to be written with a word of 0x7's.
18. The I/O cache line is verified to be set to valid, and modified.
19. The central cache line is verified to be invalidated.

WWM *pa= sa= si=*

The *Write Miss with WB and CPU Cache Miss* test on the Cached Hits/Misses Tests Menu verifies that, when the I/O cache line is modified and a write miss occurs, the I/O cache line data is written back to main memory, the I/O cache line is updated with the new data, and the I/O cache tags are set to modified.

The steps in the test are performed in the following order:

1. The I/O cache tags are set to valid, and modified.
2. A pattern of 0x2's is stored in the I/O cache line.
3. The central cache line for the address is set to valid, not modified.
4. A pattern of 0x4's is stored in the central cache line.
5. A pattern of 0x0's is stored in the memory line.
6. A DVMA write operation, writing a word of 0x7's, is performed.
7. The I/O cache line data is verified to be written with a word of 0x7's.

8. The I/O cache line is verified to be set to valid, and modified.
9. The central cache line is verified to be invalidated.
10. The main memory line is verified to have been written back with a pattern of 0x2's.

WWH *pa= sa= si=*

The *Write Miss with WB and CPU Cache Hit* test on the Cached Hits/Misses Tests Menu verifies that, in the case of a DVMA write operation when the I/O cache line is already modified, the I/O cache line is updated and remains modified, and the central cache line is invalidated.

The steps in the test are performed in the following order:

1. The I/O cache tags are set to valid, and modified.
2. A pattern of 0x2's is stored in the I/O cache line.
3. The central cache line for the address is set to valid, and modified.
4. A pattern of 0x4's is stored in the central cache line.
5. A pattern of 0x0's is stored in the memory line.
6. A DVMA write operation, writing a word of 0x7's, is performed.
7. The I/O cache line data is verified to be written with a word of 0x7's.
8. The I/O cache line is verified to be set to valid, and modified.
9. The central cache line is verified to be invalidated.
10. The main memory line is verified to have been written back with a pattern of 0x2's.

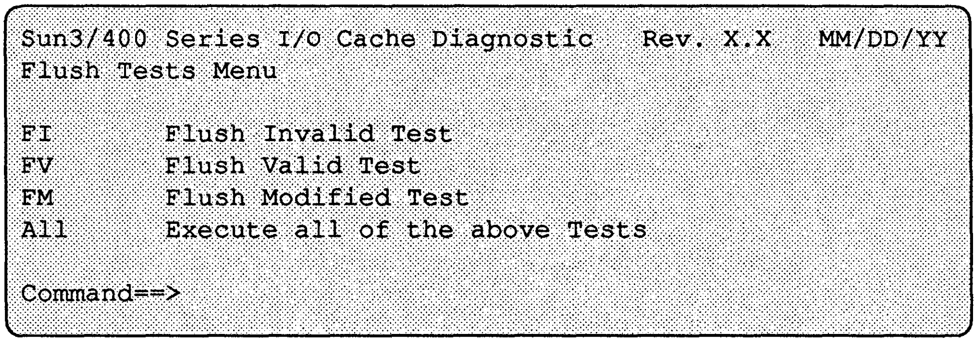
A

The *Execute All of the Above Tests* option on the Cached Hits/Misses Tests Menu executes all of the tests, in the order that they appear on the menu.

Flush Tests Menu

FL

When you choose the *Flush Tests Menu* option from the Main Menu, this submenu displays:



```
Sun3/400 Series I/O Cache Diagnostic   Rev. X.X   MM/DD/YY
Flush Tests Menu

FI        Flush Invalid Test
FV        Flush Valid Test
FM        Flush Modified Test
All       Execute all of the above Tests

Command==>
```

The tests on the Flush Tests Menu verify that the hardware flush command works correctly in cases in which the I/O cache line is valid and modified. It also verifies that no flush occurs in the cases in which the I/O cache line is either valid or invalid and not modified.

FI *pa= sa= si=*

The *Flush Invalid Test* on the Flush Tests Menu verifies that, for each I/O cache line from the start address to the start address plus the specified size address, the following conditions are true:

- The I/O cache line, being invalid, is not flushed to main memory.
- The I/O cache line remains invalid after the flush operation.

FV *pa= sa= si=*

The *Flush Valid Test* on the Flush Tests Menu verifies that, for each I/O cache line from the start address to the start address plus the specified size address, the following conditions are true:

- The I/O cache line, being valid and not modified, is not flushed to main memory.
- The I/O cache line is set to invalid after the flush operation.

FM *pa= sa= si=*

The *Flush Modified Test* on the Flush Tests Menu verifies that, for each I/O cache line from the start address to the start address plus the specified size address, the following conditions are true:

- The I/O cache line, being valid and modified, is flushed to main memory.
- The I/O cache line is set to invalid after the flush operation.

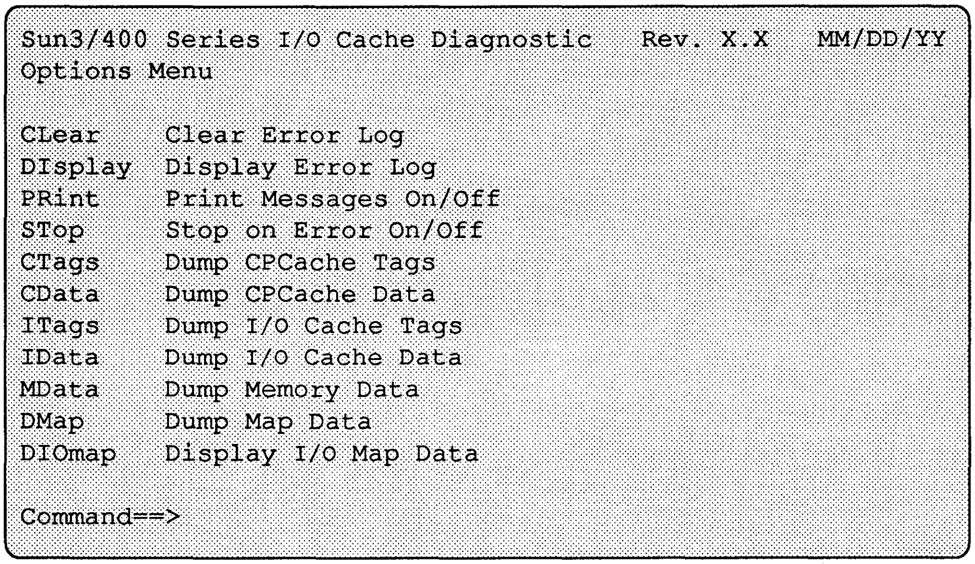
A

The *Execute All of the Above Tests* option on the Flush Tests Menu executes all of the tests, in the order that they appear on the menu.

Options Menu

OP

When you choose the *Options Menu* option from the Main Menu, this submenu displays:



```
Sun3/400 Series I/O Cache Diagnostic   Rev. X.X   MM/DD/YY
Options Menu

CLear    Clear Error Log
DIsplay  Display Error Log
PRint    Print Messages On/Off
STop     Stop on Error On/Off
CTags    Dump CPCache Tags
CData    Dump CPCache Data
ITags    Dump I/O Cache Tags
IData    Dump I/O Cache Data
MData    Dump Memory Data
DMap     Dump Map Data
DIOmap   Display I/O Map Data

Command==>
```

The choices on the Options Menu allow you to do the following:

- Clear or display the error message log
- Set or clear program run mode options
- Display data
- Display the state of the I/O cache and central cache tags

In addition to the other parameters used in the I/O Cache Diagnostic, some choices on the Options Menu also support the `s=` parameter. This parameter is described in the following table:

Parameter	*Description*
s	Run mode set variable Clear (0) Set (1)

CL

The *Clear Error Log* option on the Options Menu clears the program error message log.

DI

The *Display Error Log* option on the Options Menu displays the error log. You are prompted to terminate or to continue the display when it exceeds the screen or terminal display line space.

PR *s=*

The *Print Messages On/Off* option on the Options Menu controls printing of error messages. If you specify a parameter value of `s=0`, the option is cleared, and error messages do not print. If you specify `s=1`, the option is set, and future error messages are printed.

ST *s=*

The *Stop on Error On/Off* option on the Options Menu controls whether or not the test continues when an error is encountered. If you specify a parameter value of `s=0`, the option is cleared, and the test continues after encountering an error. If you specify `s=1`, the option is set, and the program stops when an error is encountered.

CT *sa= si=*

The *Dump CPCache Tags* option on the Options Menu displays the contents of the central cache tags over the range from the start address (the base of the central cache tag address space) to the start address plus the specified size.

CD *sa= si=*

The *Dump CPCache Data* option on the Options Menu displays the contents of the central cache data over the range from the start address (the base of the central cache data address space) to the start address plus the specified size.

IT *sa= si=*

The *Dump I/O Cache Tags* option on the Options Menu displays the contents of the I/O cache tags over the range from the start address (the base of the I/O cache tag address space) to the start address plus the specified size.

ID *sa= si=*

The *Dump I/O Cache Data* option on the Options Menu displays the contents of the I/O cache data over the range from the start address (the base of the I/O cache data address space) to the start address plus the specified size.

MD *sa= si=*

The *Dump Memory Data* option on the Options Menu displays the contents of memory over the range from the start address (the base of the memory space) to the start address plus the specified size.

DM *sa= si=*

The *Dump Map Data* option on the Options Menu displays the contents of the PMMU map over the range from the start address to the start address plus the specified size.

DIO *sa= si=*

The *Dump I/O Map Data* option on the Options Menu displays the contents of the I/O mapper over the range from the start address to the start address plus the specified size.

All Test

A

The *Execute All of the Above Tests* option on the Main Menu executes all of the I/O Cache Diagnostic tests, in the order that they appear on the menu. The tests on each submenu are performed in order. The All test is long-running and verifies the complete functionality of the I/O cache.

Quick Test

Q

The *Quick Test Sequence* option on the Main Menu executes all of the tests on the menu in order. The tests on each submenu are also performed in order.

Default Test

D

The *Default Test Sequence* option on the Main Menu executes all of the tests on the Main Menu and submenus in order.

5.9. Error

The following error messages may be displayed by the I/O Cache Diagnostic:

```
Data read by vmeloopback is incorrect
Vaddr(0xXXXXXXXX) Paddr(0xXXXXXXXX) exp(0xXXXXXXXX) obs(0xXXXXXXXX)

    where:

        Vaddr = virtual memory address of System DVMA read
        Paddr = physical memory address of System DVMA read

IO cache tag compare error
Vaddr=(0xXXXXXXXX) Paddr=(0xXXXXXXXX)

    where:

        Vaddr = virtual memory address of data
        Paddr = physical memory address of data

IO cache data compare error
Vaddr(0xXXXXXXXX) Paddr(0xXXXXXXXX)

    where:

        Vaddr = virtual memory address of data
        Paddr = physical memory address of data

Memory compare error
Vaddr=0xXXXXXXXX Paddr=0xXXXXXXXX

    where:
        Vaddr = virtual memory address of data
        Paddr = physical memory address of data
```

5.10. Glossary

Cache	An associative, fast RAM between the CPU and main memory.
Cache Line	The least number of words copied between the cache and memory.
Cache Hit	An access in which the data is in the cache.
Cache Miss	An access in which the data is not in the cache.
DVMA	Direct Virtual Memory Access.
I/O	Input and output.
RAM	Random Access Memory.
SRAM	Shared Random Access Memory.
Tags	Control/status bits which represent the cache line's status.
VMEbus	Motorola bus interface which connects the CPU board with other peripherals.

6

Sun CPU Diagnostic

6

Sun CPU Diagnostic

6.1. 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 or Mostek MK48T02)
- System Enable Register
- Interrupt Register and Interrupts
- PROMs : IDPROM, EEPROM, BOOT PROM
- Serial Ports A and B (Zilog 8530 SCC)

NOTE *To test the Sun Floating Point chip, select* `FPU` *from the Exec's Diagnostics Menu.*

6.2. Hardware Requirements

The 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 *Appendix B* for pin assignments).

6.3. Command-line Parameters

All tests in the 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.

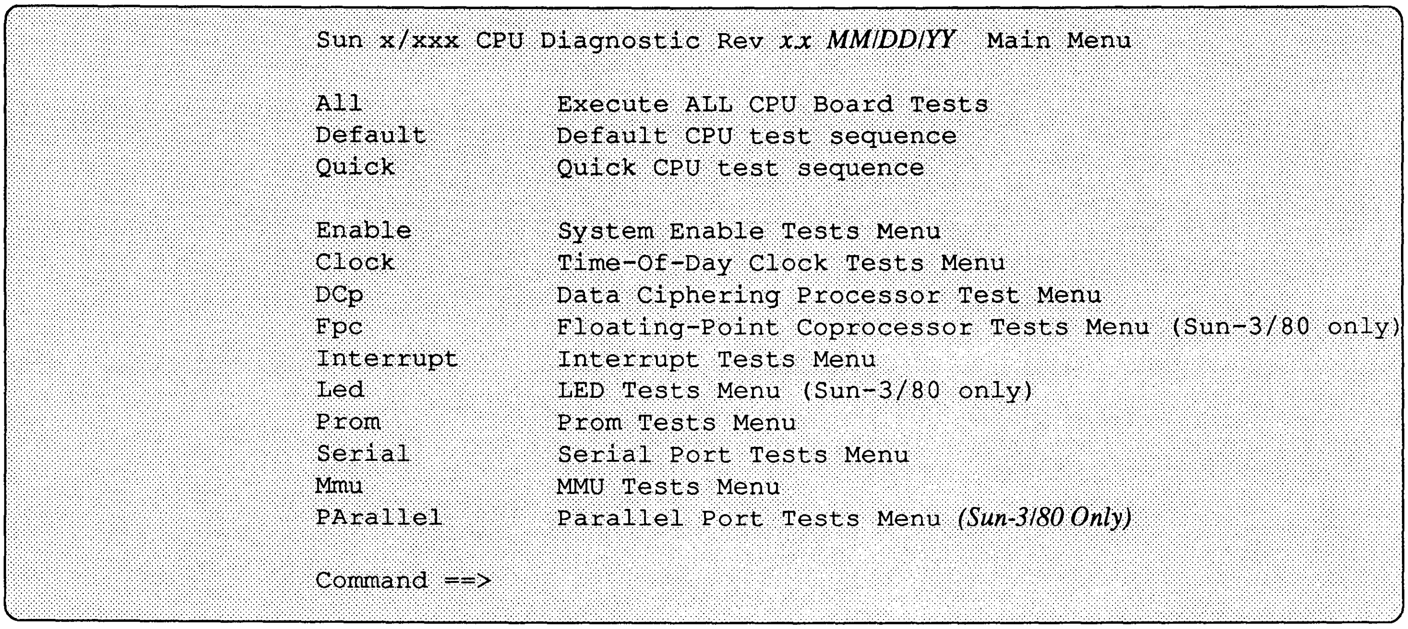
6.4. Looping on Read and Write

While looping on read or write, you may use the keys shown below to change the address being looped-on or the pattern written:

- 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.

6.5. Main Menu

The Main Menu is the first menu displayed on the screen after the Exec loads the diagnostic. From the 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 x/xxx CPU Diagnostic Rev xx MM/DD/YY  Main Menu

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

Enable         System Enable Tests Menu
Clock          Time-Of-Day Clock Tests Menu
DCp            Data Ciphering Processor Test Menu
Fpc            Floating-Point Coprocessor Tests Menu (Sun-3/80 only)
Interrupt      Interrupt Tests Menu
Led            LED Tests Menu (Sun-3/80 only)
Prom           Prom Tests Menu
Serial         Serial Port Tests Menu
Mmu            MMU Tests Menu
PArallel       Parallel Port Tests Menu (Sun-3/80 Only)

Command ==>
```

A 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.

D 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.

Q

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 submenu in the diagnostic, and runs the quick set of tests there.

E

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

C

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.

Fpc

The *Floating Point Coprocessor Test Menu* command displays a menu containing the tests for the floating-point coprocessor of the Sun-3/80 (only).

I

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

L The *LED Tests Menu* command displays a menu of tests to check Sun-3/80 CPU board LED functionality.

P

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

S

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

PA

The Parallel Port Diagnostic appears as a main menu choice only when testing a Sun-3/80 CPU board.

6.6. Clock Tests Menu

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

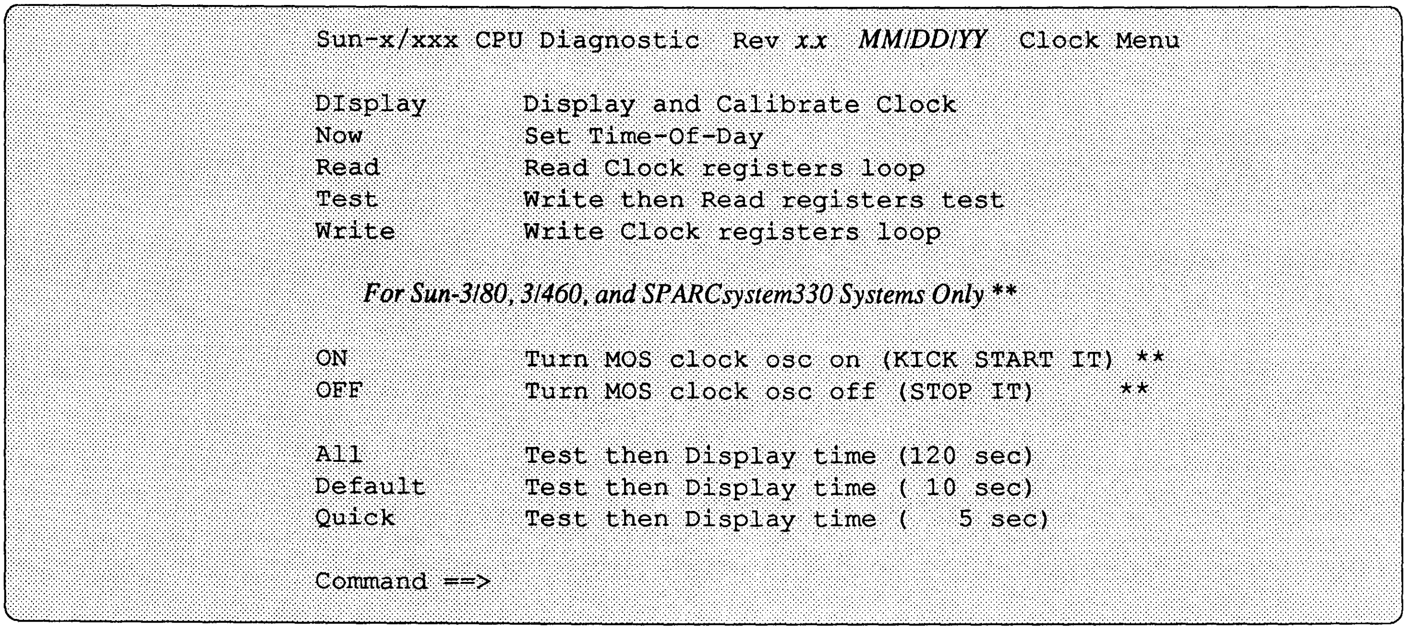
FAst

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.*



```
Sun-x/xxx 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

   For Sun-3/80, 3/460, and SPARCsystem330 Systems Only **

ON           Turn MOS clock osc on (KICK START IT) **
OFF          Turn MOS clock osc off (STOP IT)      **

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

Command ==>
```

Following are descriptions of the various Clock Menu choices.

`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 (that is, 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); that is, 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)

ON

The *Turn MOS clock oscillator on* command sets the KS (kick start) bit and waits two seconds to ensure the clock has started. The output of the clock is monitored to determine if it has been started. Five attempts will be made to start the clock before signaling the failure.

OFF

The *Turn MOS clock oscillator off* command sets the stop bit and monitors the clock output to ensure that the device has stopped. Five attempts to stop the clock are made before signaling the failure.

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
```

6.7. System Enable Tests Menu

Each Enable test (except the Diagnostic Switch test) toggles its associated bit within the System Enable Register to OFF, then ON without doing anything to any other component that might be affected by changing that bit. For example, the Copy test only toggles the Copy bit, without actually copying anything to video memory. 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=

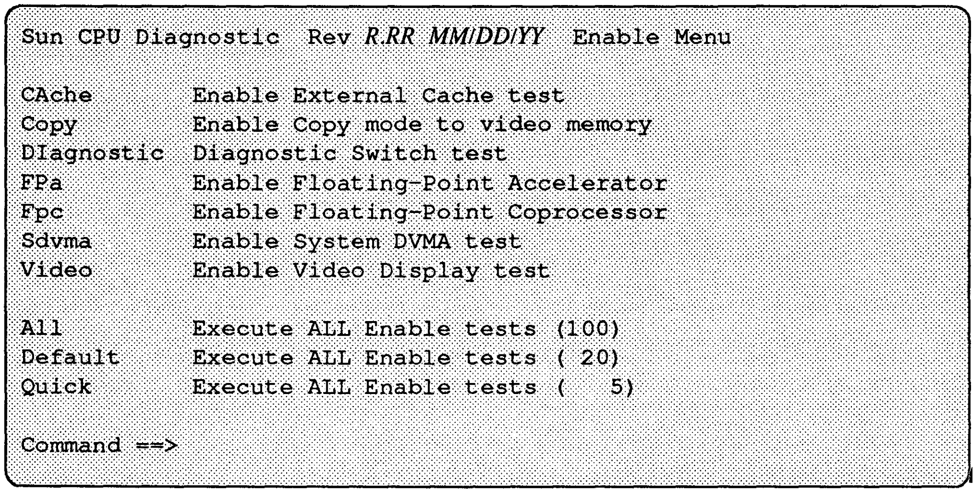
controls how fast or slow to toggle the tested bit. The value after the equals sign should be a non-negative integer. The higher it is, the longer it will take between togglings.

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 current the setting of the Diagnostic Switch.

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

OFF : switch is 1et to NORMal position.

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

Default :

```
Pass= 100

Delay= 30
```

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:

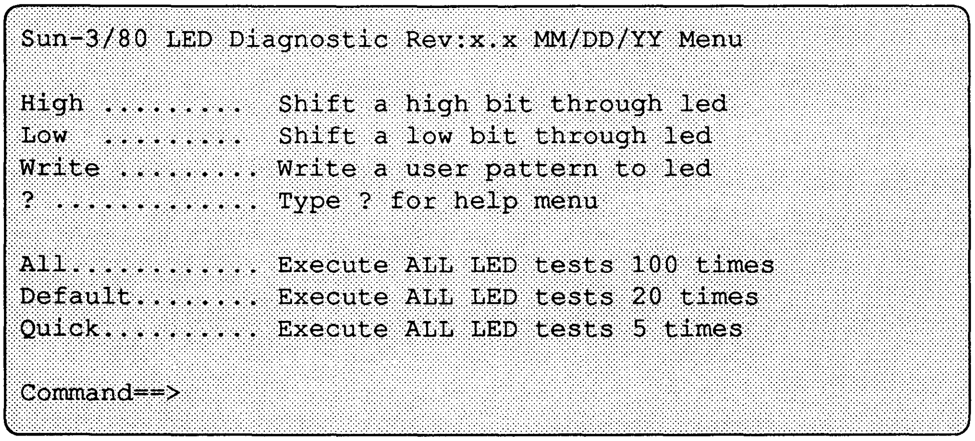
```
set Pass=5    ;  cache-video
```

6.8. Sun-3/80 LED Test

A single bit LED is used to indicate activity of the Sun-3/80 processor. The LED is made available through the reset EPLD chip. As the sysenable registers are active, a pulse is given to the LED on the next available clock cycle.

The following sections describe each module of the Sun 3/80 LED Diagnostic in detail as well as the modules usage.

The LED Diagnostic Menu, shown below, appears shen you choose **L** from the CPU Diagnostic Main Menu.



```
Sun-3/80 LED Diagnostic Rev:x.x MM/DD/YY Menu

High ........  Shift a high bit through led
Low  ........  Shift a low bit through led
Write ........ Write a user pattern to led
? ............ Type ? for help menu

All........... Execute ALL LED tests 100 times
Default....... Execute ALL LED tests 20 times
Quick......... Execute ALL LED tests 5 times

Command==>
```

High *pass=*

This test shifts a high bit through the LED. For example, entering the command

```
Command===> h pass=100
```

shifts a high bit 100 times. The default number of passes is one.

Low *pass=*

This test shifts a low bit through the led. For example, entering the command

```
Command==> l pass=100
```

shifts a low bit 100 times.

Write *pass= pat=*

This test shifts a user defined bit pattern through the LED. For example, entering the command

```
Command==> w patt=f7
```

shifts a pattern of f7 20 times through the LED.

? Entering a question mark brings up a help menu.

A Entering **All** executes the high test 10 times in long word, verbose mode, then repeats the test in word and byte modes.

D Selecting Default executes the high test five times in long word, verbose mode, and then repeats the test five times each in word and byte modes.

Q Selecting Quick executes the high test one time in long word, verbose mode.

6.9. Interrupt Tests Menu

When you choose **I** from the CPU Diagnostic Main Menu, this sub-menu is displayed.

```
Sun CPU Diagnostic  Revxx 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 ==>
```

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, 4, 6 and **8**

The *Level 1, 4, 6 and 8 Interrupt tests* turns ON the appropriate bit within the Interrupt Register, then waits for the expected interrupt to occur.

NOTE *In order to test the Sun-4/110 level 8 interrupt, the program first enables the P4 interrupt register.*

Syntax : *1 or 4 or 6 or 8 Pass= Delay=*

Default : `Pass= 100`

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

10 *Pass= FAst Freq=*

14 *Pass= FAst Freq=*
The *Level 10 and 14 Clock Interrupt tests* program 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)

12 *Pass= Delay=*
The *Level 12, 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.

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
```

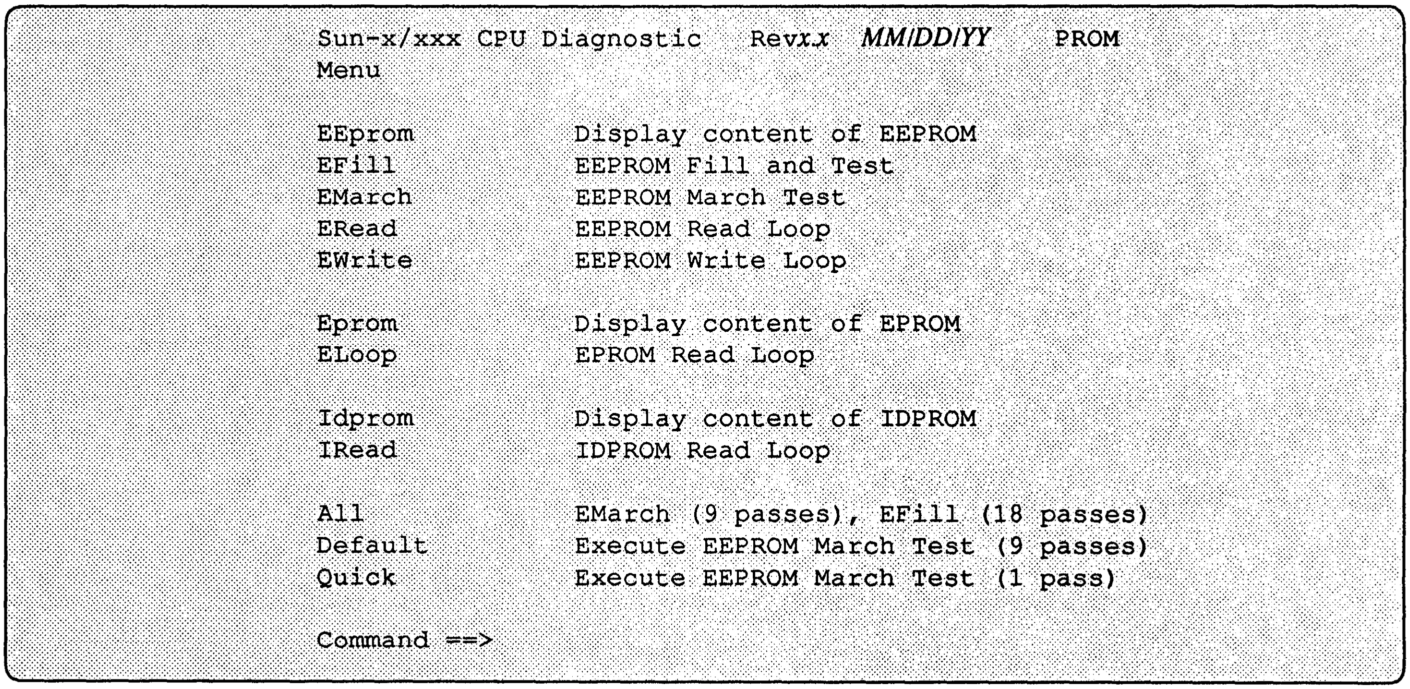
Default
The *Default* command executes the following command sequence:

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

Quick
The *Quick* command executes the following command sequence:

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

1.10. PROM Tests Menu



```
Sun-x/xxx CPU Diagnostic   Revx.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 ==>
```

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 milliseconds 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 path/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.

`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 is 0x10, 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:

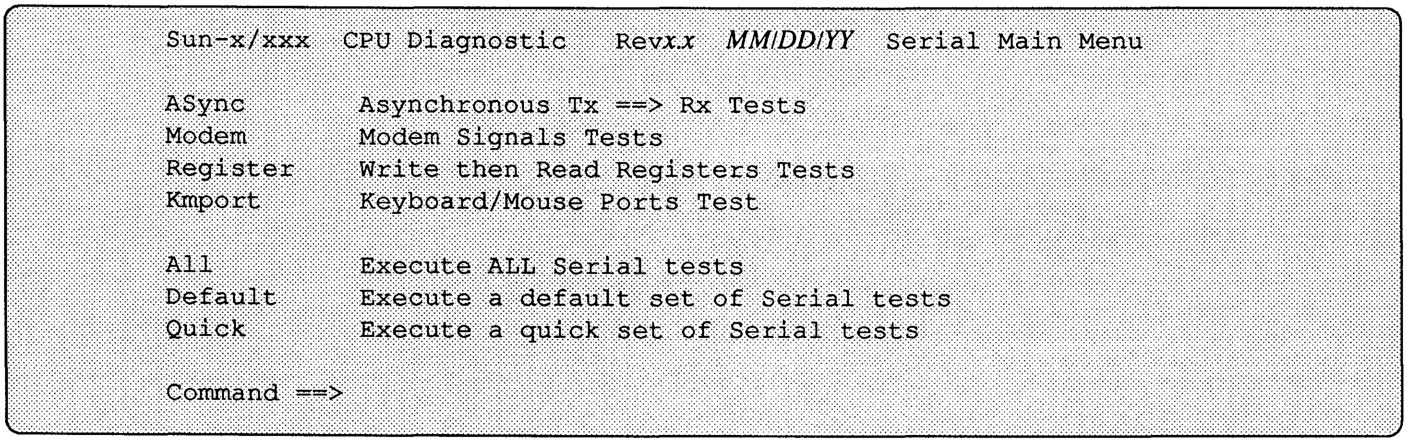
```
emarch pass=1
```

6.11. Serial Port Tests Menu

All tests in the three sub-menus offered by 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-x/xxx  CPU Diagnostic   Revx.x  MM/DD/YY  Serial Main Menu

ASync       Asynchronous Tx ==> Rx Tests
Modem       Modem Signals Tests
Register    Write then Read Registers Tests
Kmport      Keyboard/Mouse Ports Test

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

Command ==>
```

The following paragraphs describe the Serial Port menu selections.

AS

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

K The *Keyboard/Mouse Ports Test* is available only when there is no video frame buffer attached to the P4 bus. This test is available only if the CPU diagnostic is executed through a "dumb" terminal.

Modem

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>
```

Default

The *Default* command executes the following command sequence:

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

Quick

The *Quick* command executes the following command sequence:

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

Asynchronous Tests Submenu

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.

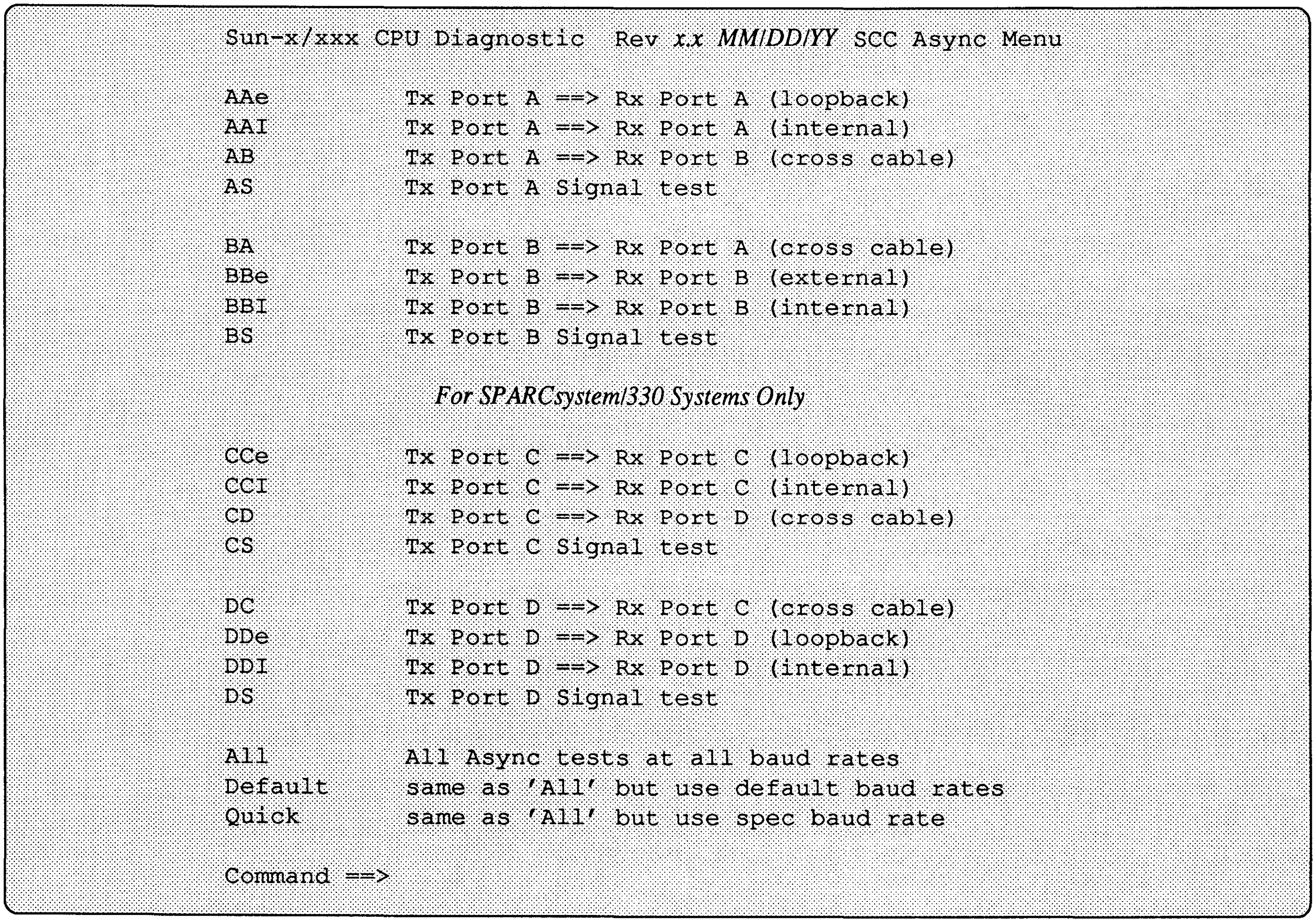
Baud=Default
Runs a test at 2 baud rates : 300 and 9600.

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.

The SCC Async tests also accept the *Delay=* parameter, which specifies the delay (in arbitrary units) after each output or transmission.



```
Sun-x/xxx CPU Diagnostic  Rev x.x MM/DD/YY SCC Async Menu

AAe         Tx Port A ==> Rx Port A (loopback)
AAI         Tx Port A ==> Rx Port A (internal)
AB          Tx Port A ==> Rx Port B (cross cable)
AS          Tx Port A Signal test

BA          Tx Port B ==> Rx Port A (cross cable)
BBe         Tx Port B ==> Rx Port B (external)
BBI         Tx Port B ==> Rx Port B (internal)
BS          Tx Port B Signal test

               For SPARCsystem/330 Systems Only

CCe         Tx Port C ==> Rx Port C (loopback)
CCI         Tx Port C ==> Rx Port C (internal)
CD          Tx Port C ==> Rx Port D (cross cable)
CS          Tx Port C Signal test

DC          Tx Port D ==> Rx Port C (cross cable)
DDe         Tx Port D ==> Rx Port D (loopback)
DDI         Tx Port D ==> Rx Port D (internal)
DS          Tx Port D 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 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). It then compares 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 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 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 loopback.

CCe *Pass= Delay= Baud=*
The *Tx Channel C ==> Rx Port C (loopback)* command transmits all possible data patterns at pre-selected baud rates to the TxD output on Port C (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 C.

CCI *Pass= Delay= Baud=*
The *Tx Channel C ==> Rx Channel C (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 C (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 loopback.

CD *Pass= Delay= Baud=*
The *Tx Port C ==> Rx Port D* command transmits all possible data patterns at pre-selected baud rates to the TxD output on Port C (pin 15 on the SCC and pin 2 on the RS-232C connector, C end). It then compares with data received from the RxD input on Port D (pin 27 on the SCC and pin 3 on the RS-232C connector, D end). This test requires a loopback cable between Port C and Port D.

CS *Pass= Delay= Baud=*
The *Tx Port C Signal test* loops while transmitting a pattern to TxD output of Port C (pin 15 on the SCC and pin 2 on the RS-232C connector). This test does not require loopback.

DC *Pass= Delay= Baud=*
The *Tx Port D ==> Rx Channel C* command transmits all possible data patterns at pre-selected baud rates to the TxD output on Port D (pin 25 on the SCC and pin 2 on the RS-232C connector, D end), then compares it with data received from the RxD input on Port C (pin 13 on the SCC and pin 3 on the RS-232C connector, C end). This test requires a loopback cable between Port C and Port D.

DDe *Pass= Delay= Baud=*
The *Tx Channel D ==> Rx Channel D (external)* command transmits all possible data patterns at pre-selected baud rates to the TxD output on Port D (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 D.

DDI *Pass= Delay= Baud=*
The *Tx Port D ==> Rx Channel D (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 D (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 loopback.

DS *Pass= Delay= Baud=*

Then *Tx Port D Signal test* loops while transmitting a pattern to TxD output of Port D (pin 25 on the SCC and pin 2 on the RS-232C connector). This test does not require loopback. BOZO

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.

Keyboard/Mouse Port Sub-Menu

The Keyboard/Mouse Port is available through the Z8530 SCC Controller chip. The SCC implements an IBM-compatible serial communications port that has asynchronous capabilities. The chip includes the buffers and drivers necessary for transmitting and receiving signal from the outside. It is self-contained and therefore needs no external support circuitry other than decoding logic to access it.

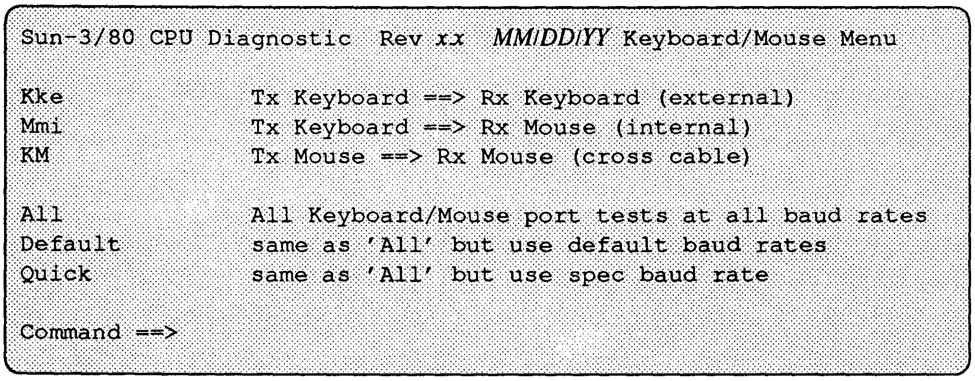
Requirements

This test requires a Keyboard Loopback Connector CPU board and keyboard.

The Serial Port

There are four signals that come from the SCC: two output data lines and two input data lines.

The external loopback connector that attaches to the keyboard port loops the two output lines to two of the input control lines. When you select **k** for *kmport* from the Serial Port Menu of the CPU diagnostic, this sub-menu is offered:



```
Sun-3/80 CPU Diagnostic  Rev xx  MM/DD/YY Keyboard/Mouse Menu

Kke             Tx Keyboard ==> Rx Keyboard (external)
Mmi             Tx Keyboard ==> Rx Mouse (internal)
KM              Tx Mouse ==> Rx Mouse (cross cable)

All             All Keyboard/Mouse port tests at all baud rates
Default         same as 'All' but use default baud rates
Quick           same as 'All' but use spec baud rate

Command ==>
```

All Keyboard/Mouse Port tests accept a *baud=* parameter to set the baud rates at which the test is to execute. 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=#**	Runs a test at the specified baud rate.

If not specified or improperly specified, *Baud=Default* is assumed.

Aai *pass= delay= baud=*

The Internal Transmit Keyboard to Receive Keyboard test programs the SCC to local loopback mode then transmits all possible data patterns at pre-selected baud rates to the TxD output on the keyboard (pin 16 on the SCC and pin 5 on the RS-423 connector) and compares with data received from its internal RxD input. This test does not require a external loopback.

Default parameter values are the same as those for the previous test.

AB *pass= delay= baud=*

The cross cable Transmit Keyboard to Transmit Mouse test transmits all possible data patterns at pre-selected baud rates to the TxD output on the keyboard (pin 16 on the SCC and pin 5 on the RS-423 connector, keyboard end) and compares with data received from the RxD input on the mouse (pin 31 on the SCC and pin 4 on the RS-423 connector, mouse end). This test requires a cross cable between keyboard and mouse. Default parameters are the same as that for the previous test.

Bbi *pass= delay= baud=*

The Internal Transmit Mouse to Receive Mouse test transmits all possible data patterns at pre-selected baud rates to the TxD output on the mouse (pin 29 on the SCC and pin 7 on the RS-423 connector, mouse end) and compares with data received from the RxD input on the mouse (pin 31 on the SCC and pin 4 on the RS-423 connector, mouse end). This test does not require an external loopback. Default parameters are the same as for the previous test.

All, Default or Quick commands

These commands execute a sequence of keyboard tests, each of which is automatically setup with proper baud rates depending on whether a loopback or a cross cable is used.

Modem Tests Sub-menu

When you select **m** for the *Modem* tests, the sub-menu that follows is displayed. 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 SCC pin and a low (-6v) on the corresponding pin of RS-232C connector.

`Signal=High`

Produces a low (0v) on the tested SCC pin and a high (+6v) on the corresponding RS-232C connector pin;

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. The Port C and D menu entries appear only when testing a SPARCsystem330 CPU board. Contact Sun Customer Support for special loopback connectors and cables. SPARCsystem330 Serial Ports B, C and D have 9-pin, rather than 25-pin connectors. Refer to Appendix B for pin assignments.

```
Sun-x/xxx CPU Diagnostic   Revx.x MM/DD/YY SCC Modem Menu

DAa         DTR Port A ==> DSR Port A (loopback)
DAB         DTR Port A ==> DSR Port B (cross cable)
DAS         DTR Port A Signal test
DBA         DTR Port B ==> DSR Port A (cross cable)
DBb         DTR Port B ==> DSR Port B (loopback)
DBS         DTR Port B Signal test

RAa         RTS Port A ==> CTS Port A (loopback)
RAB         RTS Port A ==> CTS Port B (cross cable)
RAS         RTS Port A Signal test
RBA         RTS Port B ==> CTS Port A (cross cable)
RBb         RTS Port B ==> CTS Port B (loopback)
RBS         RTS Port B Signal test

              For SPARCsystem/330 Systems Only

DCc             DTR Port C ==> DSR Port C (loopback)
DCD             DTR Port C ==> DSR Port D (cross cable)
DCS             DTR Port C Signal test
DDC             DTR Port D ==> DSR Port C (cross cable)
DDd             DTR Port D ==> DSR Port D (loopback)
DDS             DTR Port D Signal Test

RCc             RTS Port C ==> CTS Port C (loopback)
RCD             RTS Port C ==> CTS Port D (cross cable)
RCS             RTS Port C Signal Test
RDC             RTS Port D ==> CTS Port C (cross cable)
RDd             RTS Port D ==> CTS Port D (loopback)
RDS             RTS Port D 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 the 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 the 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 the 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 the signal from its CTS input (pin 18 on the SCC and pin 5 on the RS-232C connector). This test requires a loopba 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 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 stored in the variable `A_SModem` and can be modified with the Exec. 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.

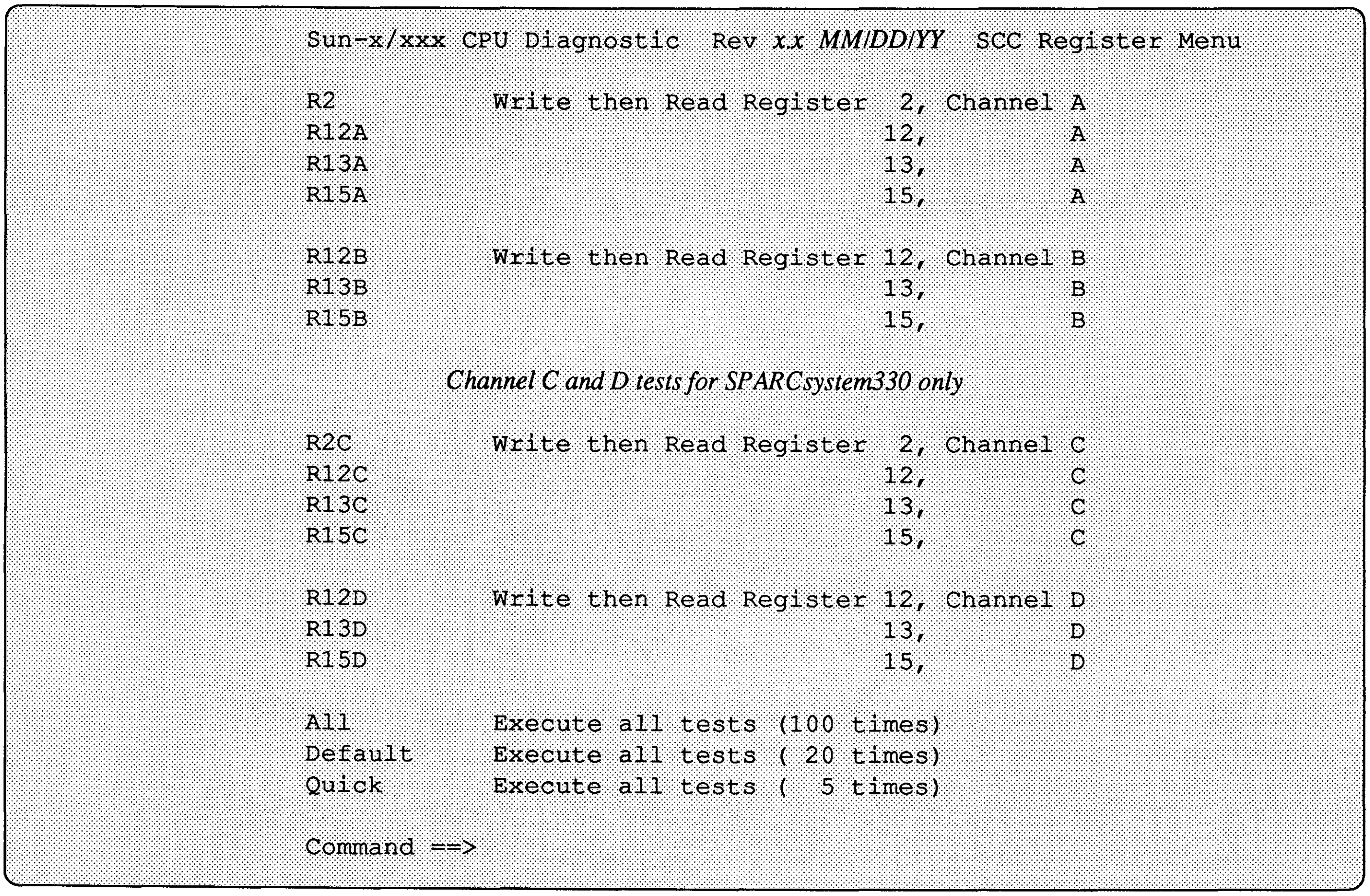
Quick

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

The SPARCsystem330 system has two additional SCC registers to test. The Channel "C" and "D" register test choices appear only on the Register Tests Sub-menu for a SPARCsystem330. On that system, Serial Ports B, C and D have 9-pin, rather than 25-pin connectors.



```
Sun-x/xxx CPU Diagnostic  Rev x.x MM/DD/YY  SCC Register Menu

R2          Write then Read Register  2, Channel A
R12A                                 12,         A
R13A                                 13,         A
R15A                                 15,         A

R12B        Write then Read Register 12, Channel B
R13B                                 13,         B
R15B                                 15,         B

          Channel C and D tests for SPARCsystem330 only

R2C         Write then Read Register  2, Channel C
R12C                                 12,         C
R13C                                 13,         C
R15C                                 15,         C

R12D        Write then Read Register 12, Channel D
R13D                                 13,         D
R15D                                 15,         D

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
```

R2C *Pass= Delay=*

R12C *Pass= Delay=*

R13C *Pass= Delay=*

R15C *Pass= Delay=*

The *Channel C : Write then Read Registers 2, 12, 13, 15* commands write to a register (in Port C) 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
```

R12D *Pass= Delay=*

R13D *Pass= Delay=*

R15D *Pass= Delay=*

The *Channel D : Write then Read Registers 12, 13, 15* commands write to a register (in Port D) 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
```

NOTE *For SPARCsystem330 systems, the* `All`, `Default` *and* `Quick` *commands execute this test sequence* `r2-r15d`.

6.12. Parallel Port Menu—Sun-3/80 Only

This menu is available only for the Sun-3/80 workstation at this time. When you invoke the Parallel Port diagostic from the CPU diagnostic main menu, this submenu appears:

```
Sun-3/80 (68030) Parallel Port Diagnostic Rev: mm/dd/yy

Register Test    Write/Read/Verify Parallel Port Registers
External Test    External Loopback Test
Interrupt Test   Parallel Port Interrupt Test

All        Execute ALL tests 100 times
Default    Execute ALL tests 10 times
Quick      Execute ALL tests 5 times

Command==>
```

The tests shown above are described in the following paragraphs.

Register *pass=*

This test writes and reads various patterns from the data register and verifies that the data read matches the written data. If you enter

```
r pass==100
```

the register test will run through the entire set of patterns for the data register 100 times. The default number of passes is one.

external *pass=*

The external loopback test requires that you install a special loopback connector on the Sun-3/80 parallel port. The pin designations of the DB25 connector are shown below. Contact Sun Customer Support for information on how to obtain this connector.

Figure 6-1 *Sun-3/80 Parallel Port Connector*

From		*To*	
Signal	*Pin*	*Signal*	*Pin*
STB	1	ERRN	15
AFX	14	SLCT	13
ININ	16	PAPE	12
SLC	17	ACKN	10
D7	9	BSYN	11

The external loopback test uses the loopback connector described above to verify that all the control and status signals are functioning properly. The diagnostic writes patterns to the control register that are sent out to the port. Because the external connector "loops" the control lines to the status lines, the same pattern is received at the status register. The diagnostic verifies that the data sent to the control register matches that read in the status register. If it does not match, an error is reported.

`Interrupt` *pass= SCOpe*

This test also requires the loopback connector described for the external loopback test.

The interrupt test simulates the existence of an external device. After enabling the PPC1 interrupt, the diagnostic writes to the corresponding output signal to assert the input signal input signal. It then verifies that the Level 1 interrupt has occurred.

If you enter

```
interrupt scope
```

the test will execute infinitely until the board is reset. If you enter

```
interrupt pass=100
```

the test will execute 100 times, and may be interrupted when you press any key.

6.13. Glossary

Exec

The Diagnostic Executive, the controlling program that starts all the diagnostics.

SCC

Serial Communications Controller chip, AMD or Zilog 8530.

TOD

Time-Of-Day Clock chip, Intersil 7170.

7

The EEPROM Editing Tool

7

The EEPROM Editing Tool

7.1. 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.

7.2. Hardware Requirements

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.

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.

7.3. 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.

7.4. The Main Menu

```
T        Primary Terminal type
R        Monitor resolution
S        Board Slots
B        Boot paths and devices
I        Initialize EEPROM (use default)
Z        Reset EEPROM to all Zeroes
SH       Show EEPROM fields
SWC      Show all Write Counts
H        High-res monitor cols, rows *
W        Write Data to EEPROM
```

**This choice appears on systems with a high resolution monitor.*

This menu looks and behaves like a typical menu under the SunDiagnostic Executive. The following text describes the sub-menus called up when you enter the various main menu selections. When you are satisfied with the changes you have made, use the **W** command from the main menu to write those changes into the appropriate EEPROM locations.

7.5. Primary Terminal Type

The EEPROM has an area for storing the primary means of communicating with the workstation user. For a typical workstation, it would be a black-and-white monitor and keyboard, and the default setting assumes this configuration.

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 Terminal Type is: B/W Console

Choose one:
```

7.6. 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 Display
B  1024 x 1024 Display
C  1600 x 1280 High Resolution Display
D  1440 x 1440 Display

Default Monitor Resolution is: 1152 x 900

Choose one:
```

The default EEPROM entry for monitor resolution is 1152 x 900, except for the Sun-4/2xx, which is 1600 x 1280.

High Resolution Monitor Columns And Rows

If your workstation model has a console monitor resolution of 1600 by 1280 pixels, you should use the Initialization 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.

7.7. 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, the tool displays a menu from which you may edit or display EEPROM information:

```
S       Slot Info
E SLot= Edit Slot
```

S If you selected `S` from this menu, the current configuration stored in EEPROM is displayed. It might look something like this:

```
   Slot 1: CPU 8 MB; WITH SF9010FPU; NO DCP; 0 kb cache
   Slot 2: Empty
   Slot 3: Memory Board 8 MB's on it
   Slot 4: Empty
   Slot 5: SCSI Sun 3; MT02 Tape ctlr; Adaptec Disk ctlr: 141 MB
   Slot 6: Empty
   Slot 6 is the last one

   Press Return to continue:
```

Your EEPROM may have some board data for a slot after the last one, since it can hold data for at least 12 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.

E To change the configuration information for a specific slot, enter

E *slot=slot_number*

from the Board Slots menu. Replace *slot_number* with the number (from 1-12) belonging to the appropriate cardcage slot. You may want to refer to the Cardcage Slot Assignment and Backplane Configuration document for your system, to ensure that the configuration has been tested and approved. If you do not enter a slot number after **E**, a menu that looks something like that shown below is offered. Italics show optional arguments or note that the command brings up a sub-menu which is described later.

Some of the menu selections shown below require one or more arguments in order to correctly use the command. To see a listing of the command usage, enter **-h** after the command:

```
command_name -h
```

Board Slots Menu **E** Command, continued.

If an argument is required and you enter just the command, a "usage" message will be displayed. As with all Exec menus, you need only enter the letters shown in upper case. For example, to specify the first Xylogics 472 tape controller for Drive Number 0, you may simply enter:

```
t ty=a n=0 d=0
```

```
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.
```

N

Entering **N** indicates that there is no board in the selected slot.

C

Entering **C** indicates that a CPU board is in the selected slot and displays a sub-menu.

M *Size=*

Entering **M** indicates that a Memory board is in the selected slot. The size can be from 0 to 255 Megabytes.

CO *type=2/3*

Entering **CO** indicates that a Color board is in the selected slot. The Color board type can be either CG2 or CG3.

FB

Entering indicates a black and white (B/W) video frame buffer is in the selected slot.

FPa

Entering **FP** indicates that a floating-point accelerator is in the selected slot.

SMd

Entering **SM** indicates that an SMD Disk Controller is in the selected slot and displays a sub-menu.

T *TYpe= Num= Drive=*

Entering **T** indicates that a tape controller is in the selected slot.

```
Entering
t ty=a
indicates a XyLogics 472 tape controller.
Entering
t ty=b
indicates a Ciprico Tapemaster controller.
```

E

Entering **E** indicates that an Ethernet Controller is in the selected slot.

A *Board= Line = Mfg=*

Entering **A** indicates that an MTI/ALM board is in the selected slot.

```
Possible entries for board are:
       A = MTI/ALM
       B = MCP/ALM2

Possible entries for Mfg are:
       A = Systech
       B = Sun
       C = unknown
```

Gp

Entering **G** indicates that a Graphics Processor is in the selected slot.

SCp

Entering **SCp** indicates that an SCP Controller is in the selected slot.

Scsi

Entering **S** indicates that a SCSI Controller is in the selected slot.

I

Entering **I** indicates that an IPC Board is in the selected slot.

GB

Entering **GB** indicates that a Graphics Buffer is in the selected slot.

SCM

Entering **SCM** indicates that a 3/75 SCSI board is in the selected slot.

MAp

Entering **MA** indicates that a MAPKIT Assembly is in the selected slot.

END

Entering **end** indicates that *Slot n* is the last one.

Board Slot Sub-Menus

When you select the CPU board, the SMD controller board, or the SCSI host adapter board from the Edit Slot Menu, a configuration sub-menu comes up.

CPU Board

When you enter **C** from the Edit Slot menu, this sub-menu is displayed.

```
                    CPU Board Menu

    M        Set Memory Size (in Mbytes) on CPU Board
    P1       Set 68881/FPU Option
    P2       Set DCP Option
    C        Set Cache Size (in kbytes)
```

SMD Controller

When you enter **SM** from the Edit Slot Menu, this configuration sub-menu is displayed:

```
                       SMD Disk Menu

    C        SMD Controller Number
    CT       SMD Controller Type
    DT       SMD Disk Drive Type
                  ?:
                 N = None (no disk)
                 A = 8"  130 MB
                 B = 8"  280 MB
                 C = 10.5" 380 MB
                 D = 10.5" 575 MB
                 E = 9" 900 MB (451 and 7053 models only)

    ND       No. of SMD Disk Drives
```

C *num=*

The controller number may be 0, 1, 2 or 3. For example, you would enter

```
c num=0
```

for the first SMD controller board.

CT *Type=*

Controller types are listed. To specify, for example, enter

```
ct t=a
```

DT= *D0= D1= D2= D3=*

The Disk Drive Type is one of the selections shown after the **DT** selection in the SMD Disk Menu. Enter the appropriate letter in place of the question mark. For example,

```
dt d0=a
```

specifies that Drive 0 is an 8-inch, 130 MB SMD drive.

ND *Total=*

Enter the total number of SMD drives installed in the system:

```
nd t=4
```

When you have finished entering configuration information for a board, the tool will show the updated board slot display.

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 something like the example below.

Board	***Default Configuration.***
CPU,Model 4/110 or 4/2xx	8 Megabytes RAM, FPC, 16 KB cache (Sun-4/110) or 128 KB (Sun-4/2xx).
Memory	0 Megabytes.
Color	Type CG3.
SCSI	Type 2: 0 Tape controllers, 0 Disk controllers. Tape Controller default: MT02. Disk Controller default: Adaptec. Disk Drive default: 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.

7.8. 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.
```

Each of the Boot Path menus selections brings up a sub-menu, described below.

U

The default operating system boot device selection gives you the choice of allowing the Boot PROM to use a list of its own to search for a bootable device during a normal boot-up, or to find, in the EEPROM, your choice of boot devices. It assigns zeroes for the control, unit and partition fields of the boot devices.

The menu that comes up when you enter **u** looks something like this:

```
Unix Default Boot Menu

A       Use Poll Devices
B       Use EEPROM's Specified Device

     Default is: Poll
```

Boot Path menu selection **E** (described below) allows you to specify the boot device.

E

The EEPROM operating system Boot Device selection from the Boot Paths and Devices menu brings up this sub-menu:

```
                         Unix Boot 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 Unix boot
```

L If you choose **L** from the Unix Boot menu, the tool will present this sub-menu:

```
    Le    Lance net
    Ie    Ethernet
    Sd    SCSI disk
    Xy    Xylogics 450/451
    XD    Xylogics 7053
    ST    SCSI Tape
    Mt    Tapemaster tape
    XT    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.

C, U and P

The **C, U,** and **P** menu items on the Unix Boot Menu refer to the controller, unit and partition numbers that would normally be entered when booting manually. For example, to manually boot a SCSI disk from Intel Ethernet and controller Number 0, unit Number 0, and Partition Number 0, the manual boot entry would be:

```
> b ie(0,0,0)
```

The **L** menu item previously described allows you to change the boot device, shown as `ie` in the example above. The C, U and P menu items allow you to change the controller, unit and partition numbers for the automatic boot from an EEPROM specified device.

If you select *Change the Unix Controller number*, you may enter that command, followed by the appropriate number. For example, to change the boot device controller number to "one", enter

```
c n=1
```

Use the same method to change the unit and partition numbers used during a normal boot.

S

If you select the *Show Unix Boot* command from the Unix Boot Menu, the current boot device stored in EEPROM will be displayed.

7.9. Diagnostic Boot Device and Path

If you chose **D** from the Boot menu, to change the Diagnostic boot device, the tool displays this sub-menu:

```
L       Change the Diagnostic boot Device letters.
C       Diagnostic boot device Controller number.
U       Diagnostic boot device Unit number.
P       Diagnostic boot device Partition number.
PA      Diagnostic Boot Path
S       Show current Diagnostic boot device and path.
```

This menu controls the information stored in EEPROM to be used during a diagnostic boot. During a diagnostic boot, the CPU board diagnostic switch is set to ON, and a terminal is usually connected to Serial Port A. The Boot PROM then checks an EEPROM location to see which device to boot from, and whether there is a special boot path. For example, you may program the EEPROM to automatically boot the SunDiagnostic Executive during a diagnostic boot, by specifying the disk where it is stored, along with the path to its directory.

L

If you choose **L** from this menu, 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.

C, U or **P**

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, as described previously for changing the normal default boot device.

PA

Enter **PA name=***diagnostic_boot_path*" and replace *diagnostic_boot_path* with the complete path to the program you wanted booted when the system diagnostic switch is set to ON. If you just enter **PA** you will receive a "usage" message.

S

As described for the normal boot EEPROM setting, entering **S** shows you what is present in the EEPROM as a diagnostic boot device and path. There is no default value for this setting, so there may be no information in this location.

7.10. Initialization

If, from the Main Menu, you select to *Initialize Everything*, the tool displays this menu:

```
A       Initialize all (T,R,S,B,H,P)
T       Initialize Primary Terminal
R       Initialize Monitor Resolution
S       Initialize Board Slots
B       Boot paths and devices
H       High-res Monitor cols, rows
P       Initialize Test Pattern
Z       Set Field
```

When you select to initialize everything, these defaults are set:

Primary Terminal Type: B/W Monitor and keyboard.

Monitor resolution: 1152 x 900 for Sun-3 and Sun-4/110, 1600 x 1280 for Sun-3/2xx and Sun-4/2xx.

Board slots:

1	CPU with 8 MB RAM, FPC, 0 KB cache.
2	Empty.
3	Empty
4-12	Empty.

NOTE *The program checks for a value in EEPROM that represents the last slot. The word* END *will appear next to that slot, indicating that the previous slot is the last to contain a board.*

Please note that, after this initialization, if you wish to modify the board slot information to describe a larger number of slots, you must first request to change the slot marked END. *Change it so that it no longer indicates that the previous is the last one.*

To set the primary device you use to communicate with the system, enter **T** from the Initialization menu.

To set the monitor resolution, enter **R** from the Initialization menu.

To set the board slot information, enter **S** from the Initialization menu.

To set the boot path and boot device information, enter **B** from the Initialization menu.

To set the number of columns and rows your high resolution (1600 x 1280 pixels) monitor has to the default of 80 columns and 34 rows, enter **H** from the Initialization menu.

To set the default test pattern, enter **P** from the Initialization menu.

7.11. EEPROM Reset

You can elect to clear the entire EEPROM to all zeros by selecting **Z**, for `Reset EEPROM to all zeros`, from the Main Menu. This does not, however, clear the fields that hold the EEPROM write count.

7.12. 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).

7.13. Show All Write Counts

Selecting **SWC** 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." For more information on the EEPROM layout, refer to the *PROM User's Manual.*

7.14. Write Data to EEPROM

Selecting **W** from the Main menu updates the EEPROM with the changes you have entered. All user-entered data is stored in a temporary buffer. If you do not wish to write that data to EEPROM, simply answer **N** when the tool prompts you to write data to EEPROM, and then press the [Esc] key.

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:

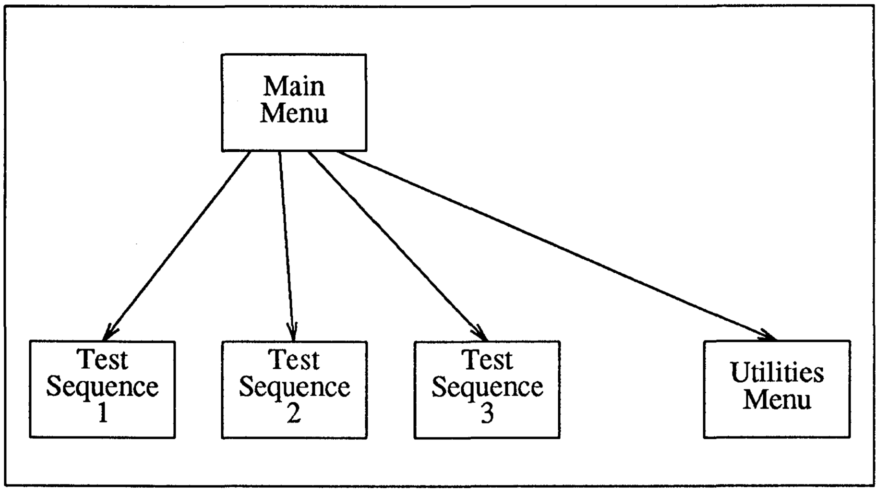
- □ register tests
- □ Weitek tests
- □ pipeline tests
- □ microcode controller tests

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:

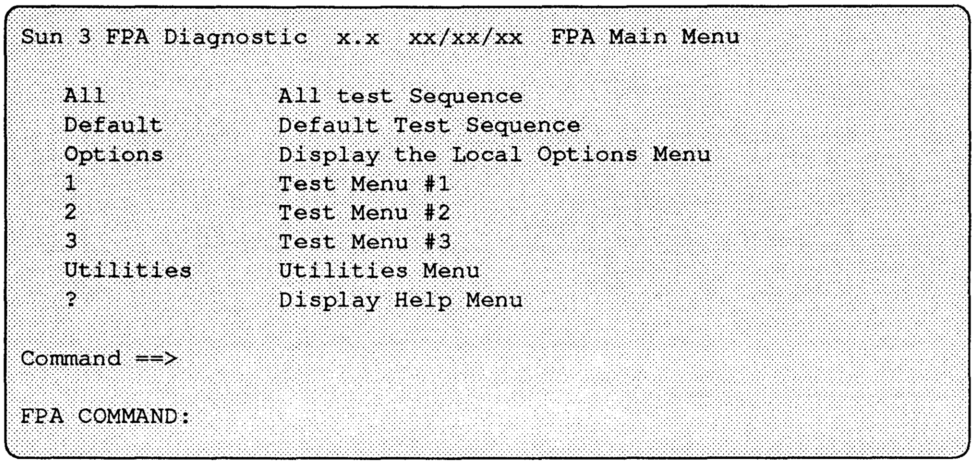
`All`

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

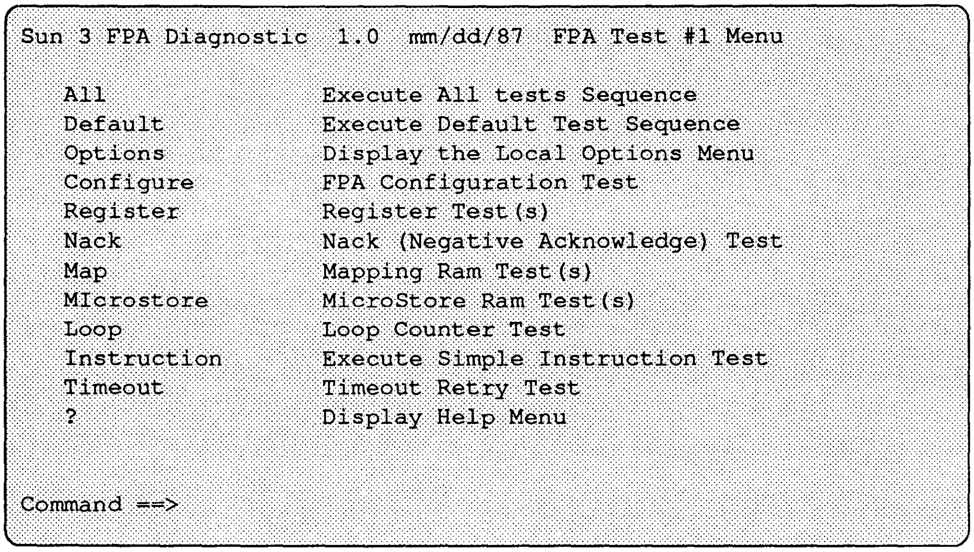
This command is not functional at this time.

?

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=*

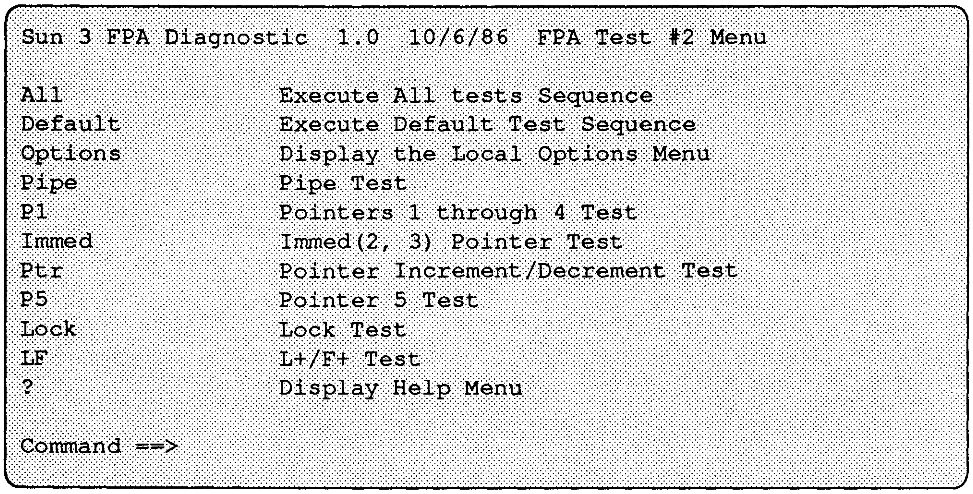
The *timeout/retry test* tests the FPA board's timeout circuitry using a microcode program. The test runs a set of microcode instructions, one after another, in an infinite loop program, to test the timeout circuitry. The instructions tested are; pipe write, pipe read, context write, context read, and 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

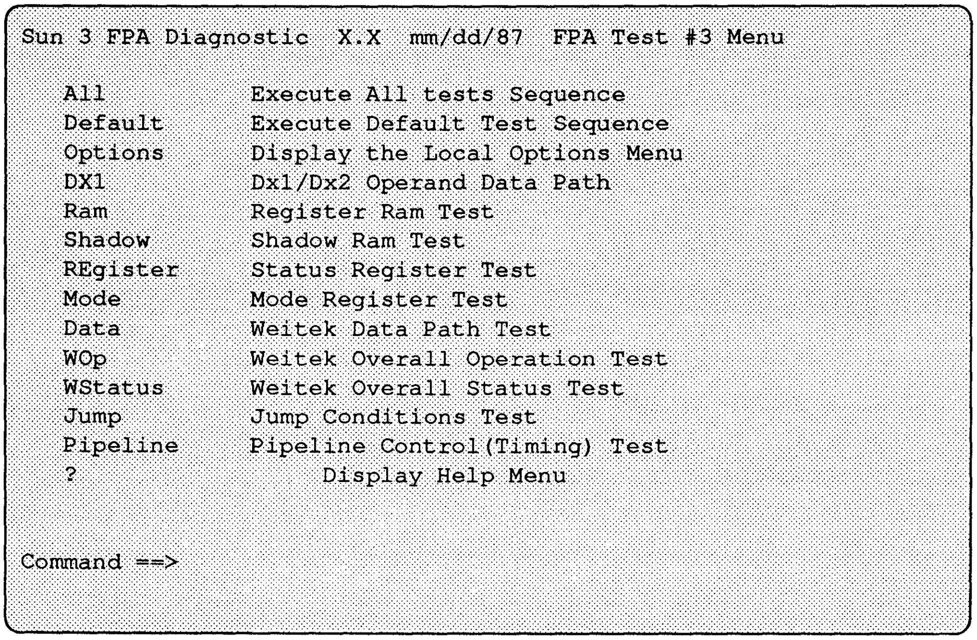
In this test, the micromachine itself sends F+ codes to the Weitek chips. The microstore is filled with instructions that send F+ values. The instructions are then executed, one at a time. After each instruction, the test reads the weitek registers to ensure the proper value was received. This cycle 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     Pipeline 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.

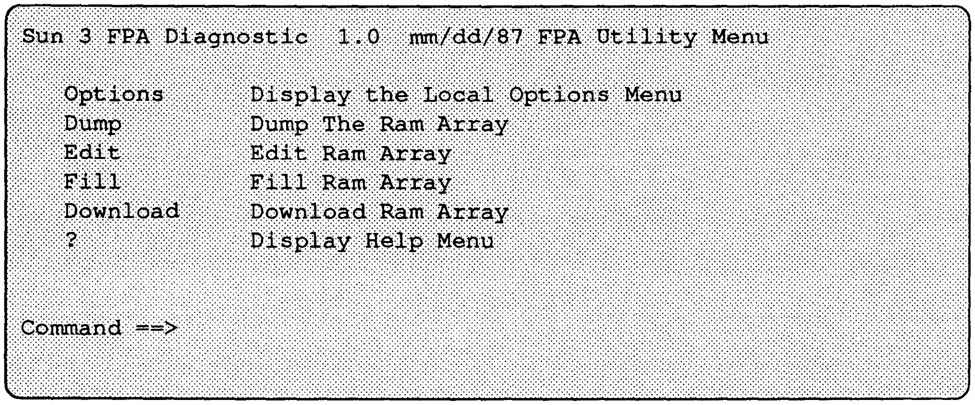
?

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

Sun-3/400 Series FPA-Plus Diagnostic

9

Sun-3/400 Series FPA-Plus Diagnostic

9.1. General Description

The FPA-Plus (Floating Point Accelerator-Plus) is an optional single board that can be installed on Sun-3/400 series systems containing the Motorola MC68030 CPU. The FPA-Plus executes floating point operations at rates approaching 1.6 Mflop. It supports full 32-bit and 64-bit formats, and operations conforming to the I.E.E.E. standard 754.

The board consists of the ABACUS Gate Array (which contains the data register and microcode state machine), a TI8847 floating point processor, the RAM Register File (1024 x 64 bits), Constants RAM (1024 x 64 bits), and microstorage RAM (16,384 x 64 bits). Communication with the MC68030 processor is implemented through the P2-MEZ connector. This interface is asynchronous, to allow for varying processor speeds.

The Sun-3/400 Series FPA-Plus Diagnostic provides basic hardware tests, functional tests, and utilities for comprehensive testing and debugging of the FPA-Plus board.

9.2. What This Chapter Contains

Following an overview of the diagnostic and a list of required hardware and software, the FPA-Plus Diagnostic Main Menu and submenus are discussed. Menu selections and optional parameters are listed, along with brief descriptions. The end of the chapter contains a list of error messages and a glossary.

9.3. Overview of the Diagnostic

The FPA-Plus Diagnostic runs under the Exec and conforms to the interface standards of the Exec. All, Default, and Quick Test sequences are provided, as well as individual tests. Parameters of the tests are given default values upon execution. Online help is provided. The diagnostic also generates and logs error messages for later retrieval.

In addition to the basic hardware and functional tests, a submenu provides access to special tools and utilities. The utilities allow you to display the contents of memory locations, execute tight scope loops, and download then alter the data in specific areas of RAM.

9.4. Hardware Requirements

The following hardware is required to run the FPA-Plus Diagnostic:

□ A Sun-3/400 series system with MC68030 CPU

□ An FPA-Plus board

- A Sun Pedestal with VME backplane to contain the CPU board
- A monitor
- A keyboard
- A boot device (local disk, local tape, or remote disk over Ethernet)

9.5. User Interface

The user interface of the FPA-Plus Diagnostic adheres to the menu standards of the Exec. A Main Menu and submenus are provided. Each option may be selected from a menu by typing the letter or letters displayed in upper case in the column on the left side of the menu.

To move back one level in the menu hierarchy, press [ESC] then press [RETURN]. To exit the FPA-Plus Diagnostic and return to the Exec, type **x** when the Main Menu is displayed. You are asked to confirm if you want to leave the diagnostic.

Additional parameters may be specified on the command line. The FPA-Plus Diagnostic supports the common parameters and environment variables (`Pass=`, for example) that may be used with other tests that run under the Exec. A summary of these parameters is given in the "Command Parameters" section later in this chapter. For a complete discussion of the parameters, see Chapter 2, "Using the SunDiagnostic Executive."

All, Default, and Quick Test sequences are provided on both the Main Menu and submenus.

To display online Help for any menu option, enter the following on the command line:

? *option_name*

9.6. Starting the Diagnostic

For information on starting the Exec, refer to Chapter 2, "Using the SunDiagnostic Executive." After you have started the Exec, choose the Sun-3/400 Series FPA-Plus Diagnostic from the Diagnostics Menu.

9.7. The Diagnostic Menus

This section of the chapter provides a modular description of the FPA-Plus Diagnostic, beginning with the Main Menu and working down through the options available on each of the submenus. A list of error messages for each test is given in the section entitled "Error Messages."

Main Menu

The FPA-Plus Diagnostic Main Menu, which displays when you start the FPA-Plus Diagnostic, provides access to the submenus of the individual tests:

```
FPA+ Board Diagnostic   Rev: X.X    MM/DD/YY

                    Main Menu

A        - Perform All Test Sequence
D        - Perform the Default Test Sequence
Q        - Perform the Quick Test Sequence
Btm      - Basic Hardware Tests Menu
Ftm      - Functional Tests Menu
Tum      - Tools and Utilities Menu
DCE      - Display Current Environment settings

*************** Environment Variables *****************

BATCH= - Enable/Disable BATCH Processing [BATCH=ENable|DISable]
MORE= - Enable/Disable More Out prompt [MORE=ENable|DISable]
INFO= - Set INFO reporting Level  [INFO=level_num]
STATus= - Set Status reporting Level [STATus=level_num]
Pass= - Number of times a test is to be run [P=count | *]
SCope= - Enable/Disable Auto Scope Loop [SC=on|off]
STop= - Stop on Nth error [ST=count | *]


Command ==>
```

Command Parameters

The menu options in the FPA-Plus Diagnostic take optional parameters and environment variables. You can specify the parameters by entering them on the command line when you run individual tests. If no parameters are specified, the default values are used.

The environment variables (also shown on the Main Menu) are summarized in the table that follows. For more information on optional parameters and environment variables, refer to Chapter 2, "Using the SunDiagnostic Executive."

Parameter	*Description*	*Value*
BATCH	Enables or disables batch processing. When this variable is enabled, you can enter a series of valid options from different menus.	ENable/DISable
MORE	Enables or disables the More output prompt.	ENable/DISable

Parameter	*Description*	*Value*
`INFO`	Sets the level of verbosity of informational messages.	Level number
`STATus`	Sets the level of verbosity of status messages.	Level number
`Pass`	Determines the number of times a test or sequence of tests is run.	Positive integer
`SCope`	Enables or disables the automatic scope loop.	on/off
`STop`	Stops testing when Nth error is encountered.	Positive integer

Any parameters that are specific to individual tests are described together with the tests to which they apply.

Perform All Test Sequence

A

The *Perform All Test Sequence* option on the Main Menu executes the All Test option within each of the submenus. This sequence is long-running and executes tests that may require your intervention. The default value of `Pass=` is 1.

Perform the Default Test Sequence

D

The *Perform the Default Test Sequence* option on the Main Menu executes the Default Test option within each of the submenus. This sequence is long-running but executes only tests that do not require intervention. The default value of `Pass=` is 1.

Perform the Quick Test Sequence

Q

The *Perform the Quick Test Sequence* option on the Main Menu executes the Quick Test option within each of the submenus. This sequence is relatively short and executes only tests that do not require intervention. The default value of `Pass=` is 1.

Basic Hardware Tests Menu

B

When you choose **B** from the Main Menu, the Basic Hardware Tests Menu is displayed. This menu provides access to three submenus that you can use for in-depth hardware testing of the FPA-Plus board. For a complete description of the options on this menu and its submenus, see the section entitled "Basic Hardware Tests Menu," later in this chapter.

Functional Tests Menu

F

When you choose **F** from the Main Menu, the Functional Tests Menu is displayed. This menu provides access to two submenus that contain options for testing the functionality of the FPA-Plus. For a complete description of the options on this menu and its submenus, see the section entitled "Functional Tests Menu," later in this chapter.

Tools and Utilities Menu

T

When you choose **T** from the Main Menu, the Tools and Utilities Menu is displayed. The options on this menu are not tests, but rather functions which aid testing and debugging. They include utilities for displaying environment settings and the contents of memory locations, downloading data to specific areas of RAM, and displaying and clearing the error log. For a complete description of these options, see the section entitled "Tools and Utilities Menu," later in this chapter.

Display Current Environment Settings

DCE

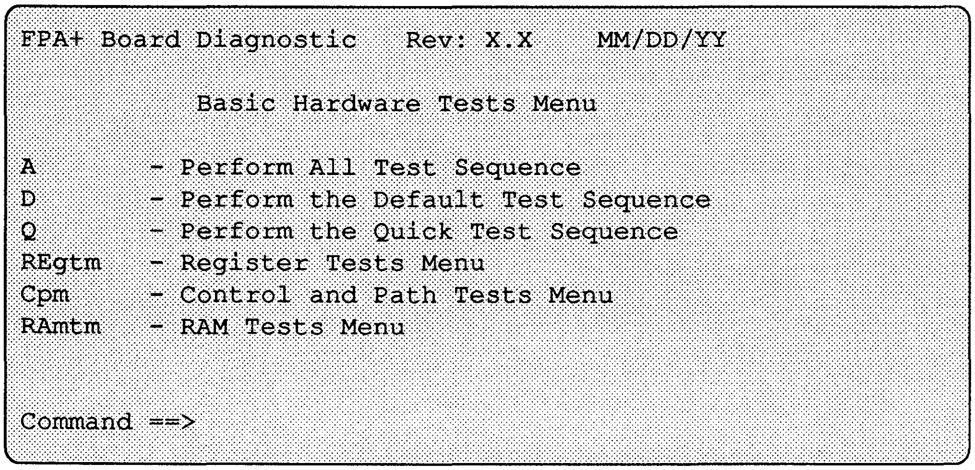
The *Display Current Environment Settings* option on the Main Menu displays the current values of all environment variables and options. A sample result of entering this command is shown in the following display:

```
 -------------------------------------------------------
FPA-Plus Diagnostics Environment Setting:

Board/Device Name='FPA-Plus' Physical Address=0x5c000000
Size = 0x1fff bytes            Base Address = 0xe0000000
BATCH Processing  : 'DISable'
MORE Out Prompt   : 'DISable'
Scope Loop        : 'off'
INFO   Level      : 1
STATus Level      : 1
Test Passcount        : 1
Stop On Nth Error     : *
Soft Error retry Count: 0
 -------------------------------------------------------

 Please Press the 'Return' Key to Continue.
```

Basic Hardware Tests Menu

B

When you choose **B** from the Main Menu, the Basic Hardware Tests Menu is displayed:



```
FPA+ Board Diagnostic    Rev: X.X    MM/DD/YY

          Basic Hardware Tests Menu

A        - Perform All Test Sequence
D        - Perform the Default Test Sequence
Q        - Perform the Quick Test Sequence
REgtm    - Register Tests Menu
Cpm      - Control and Path Tests Menu
RAmtm    - RAM Tests Menu


Command ==>
```

This menu provides access to three submenus that you can use for in-depth hardware testing of the FPA-Plus board.

A

The *Perform All Test Sequence* option on the Basic Hardware Tests Menu executes the All Test option within each of the submenus. This sequence is long-running and executes tests that may require your intervention. The default value of `Pass=` is 1.

D

The *Perform the Default Test Sequence* option on the Basic Hardware Tests Menu executes the Default Test option within each of the submenus. This sequence is long-running but executes only tests that do not require intervention. The default value of `Pass=` is 1.

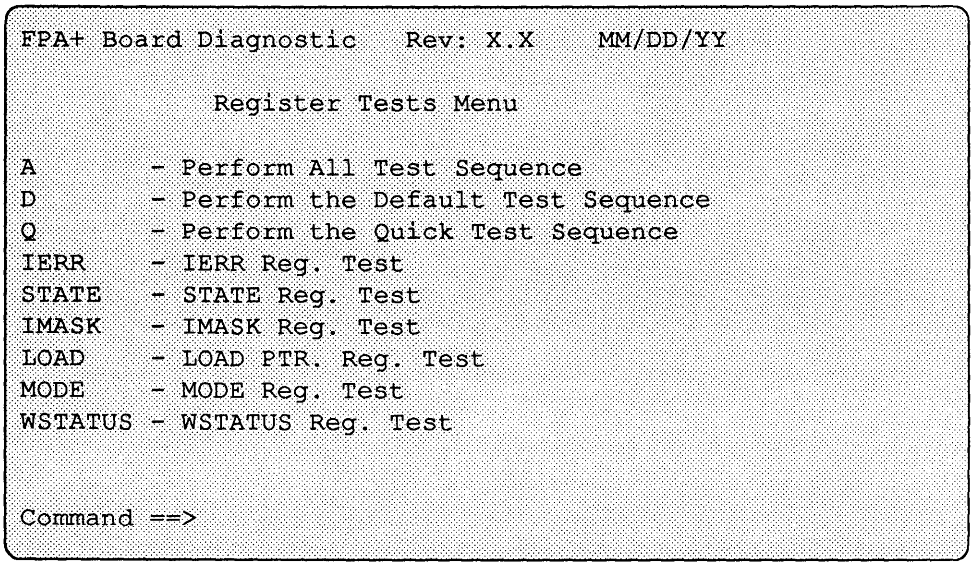
Q

The *Perform the Quick Test Sequence* option on the Basic Hardware Tests Menu executes the Quick Test option within each of the submenus. This sequence is relatively short and executes only tests that do not require intervention. The default value of `Pass=` is 1.

Register Tests Menu

RE

When you choose **RE** from the Basic Hardware Tests Menu, the Register Tests Menu is displayed:



```
FPA+ Board Diagnostic    Rev: X.X     MM/DD/YY

            Register Tests Menu

A       - Perform All Test Sequence
D       - Perform the Default Test Sequence
Q       - Perform the Quick Test Sequence
IERR    - IERR Reg. Test
STATE   - STATE Reg. Test
IMASK   - IMASK Reg. Test
LOAD    - LOAD PTR. Reg. Test
MODE    - MODE Reg. Test
WSTATUS - WSTATUS Reg. Test


Command ==>
```

The tests on this menu allow you to verify the FPA-Plus registers and counters, associated logic, and basic functional behavior.

A

The *Perform All Test Sequence* option on the Register Tests Menu executes all of the tests on this menu in sequence. The default value of `Pass=` is 1.

D

The *Perform the Default Test Sequence* option on the Register Tests Menu executes all of the tests on this menu in sequence. The default value of `Pass=` is 1.

Q

The *Perform the Quick Test Sequence* option on the Register Tests Menu executes all of the tests on this menu in sequence. The default value of `Pass=` is 1.

IERR

The *IERR Register Test* option on the Register Tests Menu performs a write/read operation to the IERR Register. This register is used for error indications. The only eight significant bits in the register are bits 23–16. The hexadecimal values 0–FF are written and verified within field [23:16] of the IERR Register.

STATE

The *STATE Register Test* option on the Register Tests Menu writes and verifies certain hexadecimal patterns between 0 and FF in the STATE Register. This register is used to enable or disable access to the Register File or microstore RAM. The significant bits in the register are bits 7–0 (bit 5 is always the same as bit 7). The STATE Register can only be written from the supervisor state.

IMASK

The *IMASK Register Test* option on the Register Tests Menu performs a write/read operation to the IMASK Register. The only significant bits in this register are bits 3–0. Bits 3–1 are "hardwired" with the board version number. Bit 0, which can be written and read, is used for enabling or disabling the TI inexact result error status mask.

LOAD

The *LOAD PTR Register Test* option on the Register Tests Menu writes and reads all hexadecimal patterns from 0 to FFFF in the LOAD PTR Register. This register can be used with the LD RAM Register for write/read operations to the Register-File and the microstore RAM.

MODE

The *MODE Register Test* option on the Register Tests Menu checks the MODE Register by performing a write/read operation to it. The significant bits are bits 3–0. Values are loaded into this register using microinstructions then read back directly in stable and clear modes.

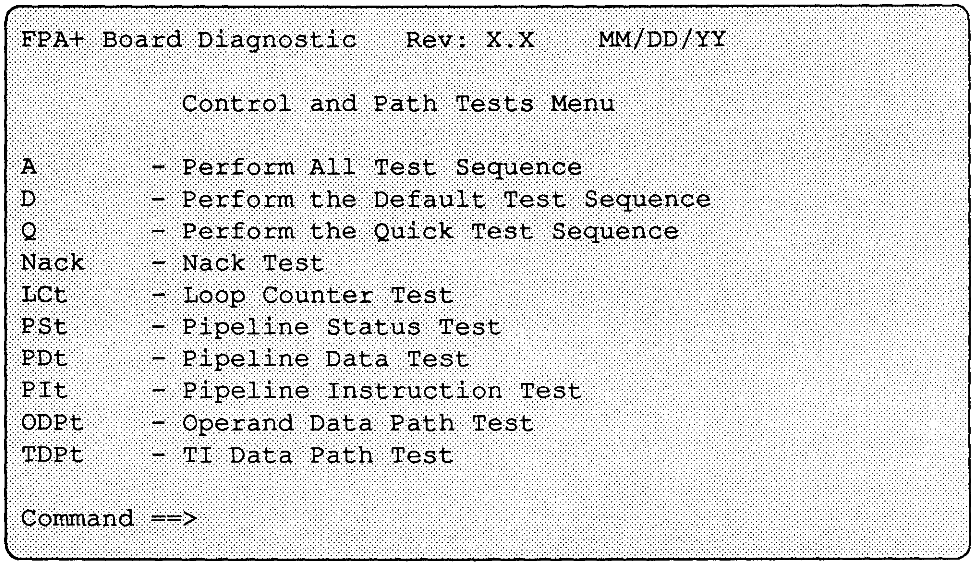
WSTATUS

The *WSTATUS Register Test* option on the Register Tests Menu writes to the WSTATUS Register bits 11–8 to generate the TI status. It then reads back and verifies the full status condition. When written, bits 11–8 of the WSTATUS Register decode the value of bits 0–4 and bit 15. The test is performed twice, once with the inexact error mask bit (from the IMASK Register) set, and once with the inexact error mask bit clear.

Control and Path Tests Menu

C

When you choose **C** from the Basic Hardware Tests Menu, the Control and Path Tests Menu is displayed:



```
FPA+ Board Diagnostic   Rev: X.X    MM/DD/YY

          Control and Path Tests Menu

A        - Perform All Test Sequence
D        - Perform the Default Test Sequence
Q        - Perform the Quick Test Sequence
Nack     - Nack Test
LCt      - Loop Counter Test
PSt      - Pipeline Status Test
PDt      - Pipeline Data Test
PIt      - Pipeline Instruction Test
ODPt     - Operand Data Path Test
TDPt     - TI Data Path Test

Command ==>
```

The tests on this menu allow you to verify the FPA-Plus control logic and data paths.

A

The *Perform All Test Sequence* option on the Control and Path Tests Menu executes all of the tests on this menu in sequence. The default value of `Pass=` is 1.

D

The *Perform the Default Test Sequence* option on the Control and Path Tests Menu executes all of the tests on this menu in sequence. The default value of `Pass=` is 1.

Q

The *Perform the Quick Test Sequence* option on the Control and Path Tests Menu executes a sequence of the following tests on this menu:

1. Nack Test
2. Loop Counter Test
3. Pipeline Status Test
4. Operand Data Path Test
5. TI Data Path Test

The default value of `Pass=` is 1.

N

The *Nack Test* option on the Control and Path Tests Menu generates error conditions to verify the negative acknowledgement from the FPA-Plus board. When an error occurs, the IERR Register sets the corresponding status bit; resulting in a bus error exception to the CPU. The following operations are performed to cause errors:

- □ Non-32-bit access. (Address bit 1 or 0 is non-zero.)
- □ Write to supervisor space in user mode.
- □ Write to Register File with Register File access bit disabled.
- □ Write to microstore RAM with microstore RAM access bit disabled.
- □ Access illegal FPA-Plus address (other than control Register).
- □ Read an FPA-Plus write-only address.
- □ Write to an FPA-Plus read-only address.
- □ Attempt to execute microcode with load enable bit set.
- □ Generate instructions to cause illegal access sequence.
- □ Generate instruction to hang pipe.
- □ Generate instructions to cause retry timeout.
- □ Access illegal FPA-Plus Control Register address.

LC

The *Loop Counter Test* option on the Control and Path Tests Menu verifies the internal loop counter and the jump conditions with the support of a diagnostic microroutine. The microroutine loads the loop counter and hangs. The loop counter is then decremented and jump conditions are tested until the value of the loop counter reaches zero.

Because this test relies on the function of the microroutine, it cannot distinguish whether a failure is caused by the loop counter or by a bad jump condition.

PS

The *Pipeline Status Test* option on the Control and Path Tests Menu verifies the pipeline by checking the PIPE STATUS Register. This test first maps the mapping RAM with unimplemented instructions. It then generates test conditions and verifies that the pipe status is correct. Using a test table, the test performs the following:

1. Issues an instruction from a test table.
2. Reads the PIPE STATUS Register and verifies with status from the table.
3. Repeats Steps 1 and 2 with different test data from the table.

PD

The *Pipeline Data Test* option on the Control and Path Tests Menu checks the pipeline through the data pipe registers PIPE ACT D1, PIPE ACT D2, PIPE NXT D1, and PIPE NXT D2. This test first maps the mapping RAM with unimplemented instructions. It then repeats a series of steps with different hexadecimal patterns designed to identify incorrect bit connections.

The test performs the following steps (X = 0):

1. Issues double NOP instruction with (pattern<<X) and (pattern<<X+1).
2. Issues double NOP instruction with (pattern<<X+2) and (pattern<<X+3).
3. Reads PIPE ACT D1 Register and verifies data is (pattern<<X).
4. Reads PIPE ACT D2 Register and verifies data is (pattern<<X+1).
5. Reads PIPE NXT D1 Register and verifies data is (pattern<<X+2).
6. Reads PIPE NXT D2 Register and verifies data is (pattern<<X+3).
7. Performs SOFT CLEAR PIPE.
8. Repeats the above steps as X increments from 0 to 31.

PI

The *Pipeline Instruction Test* option on the Control and Path Tests Menu checks the pipeline through the instruction pipe registers PIPE ACT INS and PIPE ACT NXT. This test first maps the mapping RAM with unimplemented instructions. It then generates most legal instructions. The following steps are performed:

1. Issues an instruction to the active stage.
2. Issues an instruction to the next stage.
3. Reads the PIPE ACT INS Register and verifies against the active instruction.
4. Reads the PIPE ACT NXT Register and verifies against the next instruction.
5. Performs HARD CLEAR PIPE.
6. Repeats the above steps for all test data from the table.

ODP

The *Operand Data Path Test* option on the Control and Path Tests Menu checks the operand data path with the support of a double-precision diagnostic microroutine which does the following:

1. Writes D21 into the TEMP Register and enters idle2.
2. Writes D22 into the TEMP Register and hangs.

This routine is remapped to a double valid address, such as double-precision NOP. The Operand Data Path Test performs the following steps:

1. Issues double-precision diagnostic microroutine (at double NOP).
2. Writes walking ones pattern to upper 32-bit data.
3. Writes walking zeros pattern to lower 32-bit data.
4. Reads PIPE ACT D1 Register and verifies the walking ones pattern.
5. Reads TEMP D1 (WLWF) Register and verifies the walking ones pattern.
6. Reads PIPE ACT D2 Register and verifies the walking zeros pattern.
7. Reads TEMP D2 (READ REG) Register and verifies the walking zeros pattern.

TDP

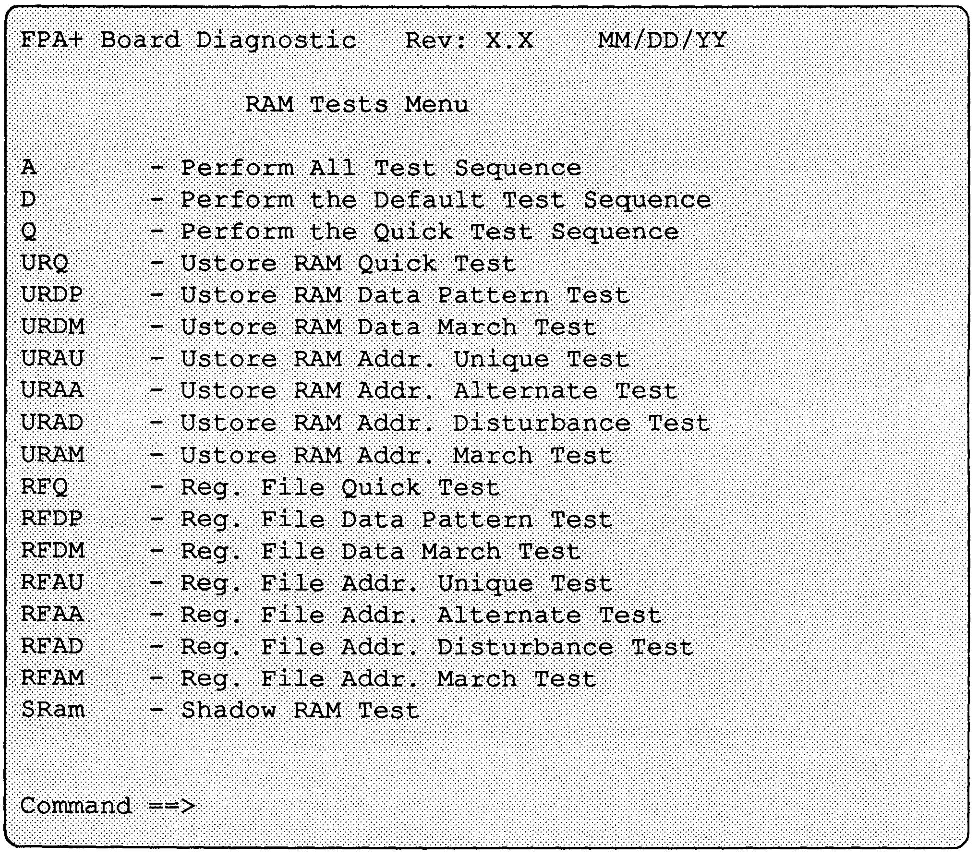
The *TI Data Path Test* option on the Control and Path Tests Menu checks the TI chip data path by issuing single- and double-precision instructions that are performed by the TI chip. The test then reads back the result and verifies. Both single- and double-precision operations perform the following steps, with walking ones and walking zeros test values:

1. Sets register 0 = 0.
2. Sets register 1 = 1.
3. Sets register 2 = *test value.*
4. Issues TI add instruction:
   register 3 = register 0 + register 2
5. Verifies that register 3 = *test value.*
6. Issues TI multiply instruction:
   register 4 = register 1 * register 2
7. Verifies that register 4 = *test value.*

RAM Tests Menu

RA

When you choose **RA** from the Basic Hardware Tests Menu, the RAM Tests Menu is displayed:



```
FPA+ Board Diagnostic   Rev: X.X    MM/DD/YY

              RAM Tests Menu

A       - Perform All Test Sequence
D       - Perform the Default Test Sequence
Q       - Perform the Quick Test Sequence
URQ     - Ustore RAM Quick Test
URDP    - Ustore RAM Data Pattern Test
URDM    - Ustore RAM Data March Test
URAU    - Ustore RAM Addr. Unique Test
URAA    - Ustore RAM Addr. Alternate Test
URAD    - Ustore RAM Addr. Disturbance Test
URAM    - Ustore RAM Addr. March Test
RFQ     - Reg. File Quick Test
RFDP    - Reg. File Data Pattern Test
RFDM    - Reg. File Data March Test
RFAU    - Reg. File Addr. Unique Test
RFAA    - Reg. File Addr. Alternate Test
RFAD    - Reg. File Addr. Disturbance Test
RFAM    - Reg. File Addr. March Test
SRam    - Shadow RAM Test


Command ==>
```

The tests on this menu allow you to verify the RAM logic and functionality.

A

The *Perform All Test Sequence* option on the RAM Tests Menu executes all of the tests (except the µstore RAM Address March Test and the Register File Address March Test) on this menu in sequence. The default value of `Pass=` is 1.

D

The *Perform the Default Test Sequence* option on the RAM Tests Menu executes all of the tests (except the µstore RAM Address March Test and the Register File Address March Test) on this menu in sequence. The default value of `Pass=` is 1.

Q

The *Perform the Quick Test Sequence* option on the RAM Tests Menu executes a sequence of the following tests on this menu:

1. µstore RAM Quick Test
2. µstore RAM Address Unique Test
3. Register File Quick Test
4. Register File Address Unique Test

The default value of `Pass=` is 1.

URQ

The *µstore RAM Quick Test* option on the RAM Tests Menu briefly checks the first ten locations of the microstore RAM. The first ten locations, each 64 bits wide, are written and verified with the following hexadecimal patterns:

0000000000000000
FFFFFFFFFFFFFFFF
5A5A5A5A5A5A5A5A
A5A5A5A5A5A5A5A5

The test uses functions of the STATE Register, LOAD PTR Register, and LD RAM Register to test the microstore RAM. The upper and lower 32 bits of each 64-bit wide microstore RAM location are written and read separately. The test performs the following steps for writing and reading each microstore RAM location:

1. Enables access to microstore bit in STATE Register (if not already enabled).
2. Loads microstore address and selects upper field [63:32] in LOAD PTR Register.
3. Loads the upper 32-bit data in LD RAM Register.
4. Loads microstore address and selects lower field [31:0] in LOAD PTR Register.
5. Loads the lower 32-bit data in LD RAM Register.

URDP

The *µstore RAM Data Pattern Test* option on the RAM Tests Menu verifies that certain sensitive patterns can be written and read properly in the microstore RAM. The test executes the following steps for each sensitive pattern:

1. Writes 64-bit pattern to all microstore RAM.
2. Reads back each location and verifies pattern written.

The following hexadecimal patterns are used:

0000000000000000
FFFFFFFFFFFFFFFF

5555555555555555
AAAAAAAAAAAAAAAA
6666666666666666
9999999999999999
1818181818181818
E7E7E7E7E7E7E7E7
7171717171717171
8E8E8E8E8E8E8E8E
C3C3C3C3C3C3C3C3
3C3C3C3C3C3C3C3C

URDM

The µ*store RAM Data March Test* option on the RAM Tests Menu verifies the uniqueness of each data bit by marching ones to all data bits. The test performs the following steps:

1. Starting with hexadecimal pattern 0000000000000001, writes pattern to entire microstore RAM space.
2. Reads back each location and verifies.
3. Repeats Steps 1 and 2 with pattern shift left one until all bits are zeros.

URAU

The µ*store RAM Address Unique Test* option on the RAM Tests Menu verifies the uniqueness of the addressing of microstore RAM. The test performs the following steps:

1. For each microstore RAM location, writes a unique address pattern.
2. Reads back each location and verifies the unique address pattern.

URAA

The µ*store RAM Address Alternate Test* option on the RAM Tests Menu checks the alternate addressing of microstore RAM. The test performs the following steps:

1. Fills microstore RAM with zeros.
2. Writes hexadecimal pattern 55555555AAAAAAAA to first and every alternate location.
3. Verifies that all even address locations contain hexadecimal pattern 55555555AAAAAAAA.
4. Verifies that all odd address locations contain zeros.

URAD

The *μstore RAM Address Disturbance Test* option on the RAM Tests Menu verifies that writing repeatedly to a specific location does not disturb other locations. The following RAM hexadecimal locations are tested:

0
3C3
C7C
1555
1999
2666
2AAA
3333
3C3C
3FFF

The test performs the following steps to all selected locations:

1. Fills microstore RAM with zeros.
2. Writes hexadecimal pattern FFFFFFFFFFFFFFFF to one location 1024 times.
3. Reads microstore RAM and verifies that no other locations are changed.

URAM

The *μstore RAM Address March Test* option on the RAM Tests Menu verifies that each microstore RAM location is unique and that driving the data bus up and down does not affect the addressing of microstore RAM. This test performs the following steps:

1. Fills microstore RAM with zeros as background.
2. Writes hexadecimal pattern FFFFFFFFFFFFFFFF to first location.
3. Reads and verifies that all other locations are zeros.
4. Writes back zero to first location.
5. Repeats Steps 2–4 for all locations.
6. Repeats Steps 1–5 with background FFFFFFFFFFFFFFFF and pattern zero.

RFQ

The *Register File Quick Test* option on the RAM Tests Menu briefly checks the first ten locations of the Register File. The first ten locations, each 64 bits wide, are written and verified with the following hexadecimal patterns:

0000000000000000
FFFFFFFFFFFFFFFF
5A5A5A5A5A5A5A5A
A5A5A5A5A5A5A5A5

The Register File is written and read through functions of the STATE Register and the LOAD PTR Register, and with support of the microcode. The size of the Register File is 2 Kbytes x 64 bits, and the upper and lower 32 bits of data are written and read separately. The test performs the following steps for writing and reading each Register File RAM location:

1. Enables Register File access bit in STATE Register (if not already enabled).
2. Loads Register File address in LOAD PTR Register.
3. Issues instruction for write/read through the LOAD PTR Register to the upper 32 bits of data.
4. Issues instruction for write/read through the LOAD PTR Register to the lower 32 bits of data.

RFDP

The *Register File Data Pattern Test* option on the RAM Tests Menu verifies that certain sensitive patterns can be written and read properly in the Register File. The test executes the following steps for each sensitive pattern:

1. Writes 64-bit pattern to the entire Register File.
2. Reads back each location and verifies pattern written.

The following hexadecimal patterns are used:

0000000000000000
FFFFFFFFFFFFFFFF
5555555555555555
AAAAAAAAAAAAAAAA
6666666666666666
9999999999999999
1818181818181818
E7E7E7E7E7E7E7E7
7171717171717171
8E8E8E8E8E8E8E8E
C3C3C3C3C3C3C3C3
3C3C3C3C3C3C3C3C

RFDM

The *Register File Data March Test* option on the RAM Tests Menu verifies the uniqueness of each data bit by marching ones to all data bits. The test performs the following steps:

1. Starting with hexadecimal pattern 0000000000000001, writes pattern to entire microstore RAM space.
2. Reads back each location and verifies.
3. Repeats Steps 1 and 2 with pattern shift left one until all bits are zeros.

RFAU

The *Register File Address Unique Test* option on the RAM Tests Menu verifies the uniqueness of the addressing of the Register File. The test performs the following steps:

1. For each Register File location, writes a unique address pattern.
2. Reads back each location and verifies the unique address pattern.

RFAA

The *Register File Address Alternate Test* option on the RAM Tests Menu checks the alternate addressing of the Register File. The test performs the following steps:

1. Fills the Register File with zeros.
2. Writes hexadecimal pattern 55555555AAAAAAAA to first and every alternate location.
3. Verifies that all even address locations contain hexadecimal pattern 55555555AAAAAAAA.
4. Verifies that all odd address locations contain zeros.

RFAD

The *Register File Address Disturbance Test* option on the RAM Tests Menu verifies that writing repeatedly to a specific location does not disturb other locations. The following RAM hexadecimal locations are tested:

0
3C3
C7C
1555
1999
2666
2AAA
3333
3C3C
3FFF

The test performs the following steps to all selected locations:

1. Fills Register File with zeros.
2. Writes hexadecimal pattern FFFFFFFFFFFFFFFF to one location 1024 times.
3. Reads the Register File and verifies that no other locations are changed.

RFAM

The *Register File Address March Test* option on the RAM Tests Menu verifies that each Register File location is unique and that driving the data bus up and down does not affect the addressing of the Register File. This test performs the following steps:

1. Fills the Register File with zeros as background.
2. Writes hexadecimal pattern FFFFFFFFFFFFFFFF to first location.
3. Reads and verifies that all other locations are zeros.
4. Writes back zero to first location.
5. Repeats Steps 2–4 for all locations.
6. Repeats Steps 1–5 with background FFFFFFFFFFFFFFFF and pattern zero.

SR

The *Shadow RAM Test* option on the RAM Tests Menu verifies the functionality of Shadow RAM. Shadow RAM provides fast read access to the first 8 registers of each of the Register File contexts. The lower 1 Kbyte of the Register File contains 32 contexts, and each context holds 32 registers. The first part of this test writes to all Register File contexts then reads back from Shadow RAM to verify the data match. The test performs the following steps:

1. Writes unique data for each register to all contexts of Register File.
2. Selects context 0 in STATE Register.
3. Reads Shadow RAM Registers and verifies that data matches Register File context 0.
4. Repeats Steps 2 and 3 for all 32 contexts.

The upper 1 Kbyte of the Register File, known as Constants RAM, is used to store constant data. Constants RAM should not be accessible through Shadow RAM. The second part of this test writes to Constants RAM then reads back from Shadow RAM to verify that the data do not match. The test performs the following steps:

1. Fills upper 1 Kbyte of Register File with FFFFFFFFFFFFFFFF hexadecimal pattern.
2. Reads all Shadow RAM registers and verifies that their contents remain unchanged.

Functional Tests Menu

F

When you choose **F** from the Main Menu, the Functional Tests Menu is displayed:

```
FPA+ Board Diagnostic Rev: X.X MM/DD/YY Functional Tests Menu

A        - Perform All Test Sequence
D        - Perform the Default Test Sequence
Q        - Perform the Quick Test Sequence
F1tm     - Functional Tests Menu 1
F2tm     - Functional Tests Menu 2

Command ==>
```

This menu provides access to two submenus that you can use to verify the logic and functionality of the FPA-Plus subsystem.

A

The *Perform All Test Sequence* option on the Functional Tests Menu executes the All Test option within each of the submenus. This sequence is long-running and executes tests that may require your intervention. The default value of `Pass=` is 1.

D

The *Perform the Default Test Sequence* option on the Functional Tests Menu executes the Default Test option within each of the submenus. This sequence is long-running but executes only tests that do not require intervention. The default value of `Pass=` is 1.

Q

The *Perform the Quick Test Sequence* option on the Functional Tests Menu executes the Quick Test option within each of the submenus. This sequence is relatively short and executes only tests that do not require intervention. The default value of `Pass=` is 1.

Functional Tests Menu 1

F1

When you choose **F1** from the Functional Tests Menu, the Functional Tests Menu 1 is displayed:

```
FPA+ Board Diagnostic    Rev: X.X    MM/DD/YY

          Functional Tests 1 Menu

A        - Perform All Test Sequence
D        - Perform the Default Test Sequence
Q        - Perform the Quick Test Sequence
MSAt     - Micro Store Address Test
NEXT     - NEXT Address Test
JMP      - JMP Address Test
CALL     - CALL and RETURN Test
SIt      - Simple Instruction Test
TARt     - Timeout And Retry Test
LOCK     - Lock Test
JCt      - Jump Conditions Test


Command ==>
```

The tests on this menu allow you to verify selected functions of the FPA-Plus subsystem. The tests on this menu, when combined with those on the Functional Tests Menu 2, provide for complete functional testing of the FPA-Plus.

A

The *Perform All Test Sequence* option on the Functional Tests Menu 1 executes all of the tests on this menu in sequence. The default value of `Pass=` is 1.

D

The *Perform the Default Test Sequence* option on the Functional Tests Menu 1 executes all of the tests on this menu in sequence. The default value of `Pass=` is 1.

Q

The *Perform the Quick Test Sequence* option on the Functional Tests Menu 1 executes all of the tests (except the NEXT Address Test and the CALL and RETURN Test) on this menu in sequence. The default value of `Pass=` is 1.

MSA

The *Microstore Address Test* option on the Functional Tests Menu 1 verifies the proper connection of the address lines between the ABACUS Gate Array and the microstore. First the Load Pointer is enabled as the microstore address source. A sequential test is then performed by loading the Load Pointer with a sequence of microstore addresses and checking the UST ADDR(0xE0000F64) Register for the currently loaded value. The microstore addresses used begin at 0 and end at the last microstore location.

NEXT

The *NEXT Address Test* option on the Functional Tests Menu 1 verifies the proper generation of the NEXT microstore instruction address. First the microstore RAM is filled with micro-instructions. These instructions select the microstore PC as the source of the next instruction address and hang. The test then steps sequentially through the microstore contents. Between micro-instructions, the NEXT address is verified for the proper sequential value by checking the contents of the UST ADDR Register. The NEXT address should sequence from 0 to the last microstore location.

JMP

The *JMP Address Test* option on the Functional Tests Menu 1 verifies the proper generation of the Jump Address. The microstore RAM is filled with micro-instructions which Jump Always to a specified address and hang. The test then steps through the microstore contents, verifying the current Jump Address between micro-instructions. The Jump Address is verified by checking the contents of the UST ADDR Register.

CALL

The *CALL and RETURN Test* option on the Functional Tests Menu 1 verifies the proper generation of NEXT microstore instruction addresses from Call and Return micro-instructions. The test performs the following two steps:

1. Fills the first half of the microstore RAM with calls and the second half with returns. It then initiates the first half of calls to the second half of returns.
2. Fills the first half of the microstore RAM with returns and the second half with calls. It then initiates the second half of calls to the first half of returns.

The NEXT address is verified between micro-instructions by checking the contents of the UST ADDR Register.

SI

The *Simple Instruction Test* option on the Functional Tests Menu 1 verifies that the simplest instructions can be executed. The instructions are the following:

- Single-precision NOP (0xE0000000)
- Double-precision NOP (0xE0000004 and 0xE0001000)
- Single-precision unimplemented instruction (0xE000095C)

The test executes the following steps:

1. Transmits the instruction under test.
2. Reads the PIPE STATUS Register (0xE0000F48).
3. Compares the pipe status with the expected status.
4. Repeats Steps 1–3 for all the simple instructions.

TAR

The *Timeout and Retry Test* option on the Functional Tests Menu 1 verifies the timeout and retry capabilities by executing an infinite loop and checking to confirm that the following types of instructions time out:

- Pipe write
- Pipe read
- Shadow read
- Context read
- Context write

The test performs the following steps:

1. Maps the address of the infinite loop test to the beginning of mapping RAM.
2. Fills the pipe so that all instructions given will time out (by writing 0x0 to location 0xE0000000 twice).
3. Transmits the instruction under test.
4. Confirms the generation of a bus error indicating 256 tries.
5. Repeats Steps 2–4 for each type of instruction to be tested.

LOCK

The *Lock Test* option on the Functional Tests Menu 1 checks the interlock decoding for each type of FPA instruction. The types of instructions include the following four classes, with both single-precision classes and double-precision classes being tested together:

- Single-precision, address-encoded register address
- Single-precision, data-encoded register address
- Double-precision, register address in address of first transfer

The test performs the following steps:

1. Sets mapping RAM so that all instructions point to the unimplemented instruction.
2. Transmits instructions which use all possible result registers for a particular instruction type.
3. After transmitting each instruction, verifies that all shadow registers that should or should not interlock perform properly.
4. After each check of the shadow registers, writes to the SOFT CLEAR PIPE register.

This test is performed for both the next and active stages of the pipe.

JC

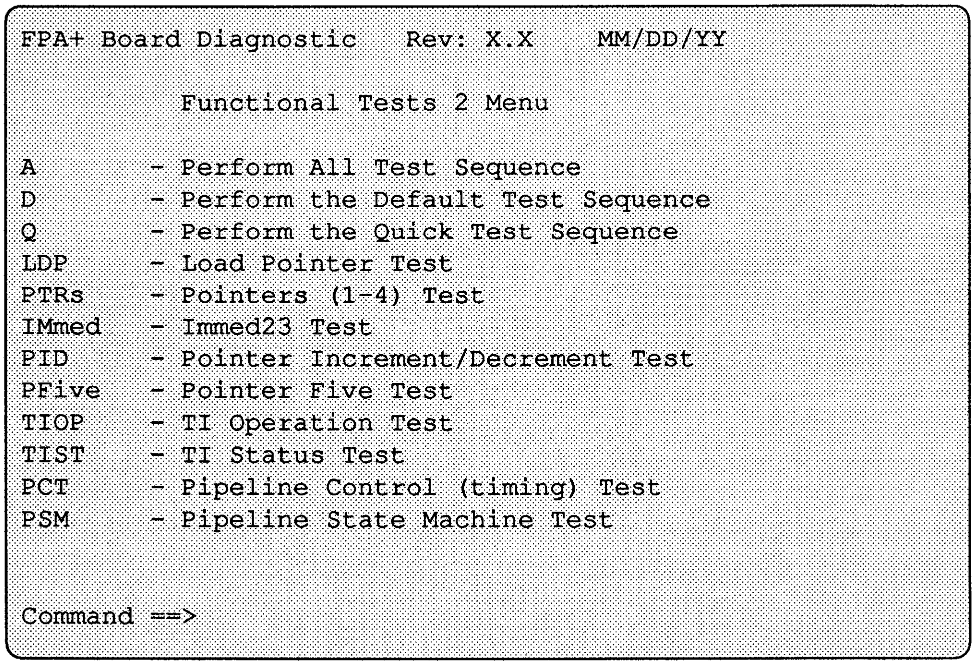
The *Jump Conditions Test* option on the Functional Tests Menu 1 verifies that all the possible jump and branching conditions of microcode execute as expected. The test uses the diagnostic microroutine that executes each jump and branch condition. This microroutine executes the JMP, JGT, JGE, JEQ, JLE, JLT, JNE, JNOTIERR, and JTIERR branch instructions.

Successful branches from all executed instructions cause the Microstore Address Register to point to an expected final location. An unsuccessful branch from one instruction causes the microroutine to halt prematurely and the Microstore Address Register to point to a location identifying the instruction that failed.

Functional Tests Menu 2

F2

When you choose **F2** from the Functional Tests Menu, the Functional Tests Menu 2 is displayed:



```
FPA+ Board Diagnostic   Rev: X.X    MM/DD/YY

          Functional Tests 2 Menu

A       - Perform All Test Sequence
D       - Perform the Default Test Sequence
Q       - Perform the Quick Test Sequence
LDP     - Load Pointer Test
PTRs    - Pointers (1-4) Test
IMmed   - Immed23 Test
PID     - Pointer Increment/Decrement Test
PFive   - Pointer Five Test
TIOP    - TI Operation Test
TIST    - TI Status Test
PCT     - Pipeline Control (timing) Test
PSM     - Pipeline State Machine Test


Command ==>
```

The tests on this menu allow you to verify selected functions of the FPA-Plus subsystem. The tests on this menu, when combined with those on the Functional Tests Menu 1, provide for complete functional testing of the FPA-Plus.

A

The *Perform All Test Sequence* option on the Functional Tests Menu 2 executes all of the tests on this menu in sequence. The default value of `Pass=` is 1.

D

The *Perform the Default Test Sequence* option on the Functional Tests Menu 2 executes all of the tests on this menu in sequence. The default value of `Pass=` is 1.

Q

The *Perform the Quick Test Sequence* option on the Functional Tests Menu 2 executes all of the tests on this menu in sequence. The default value of `Pass=` is 1.

LDP

The *Load Pointer Test* option on the Functional Tests Menu 2 verifies the proper functioning of the data path from the Load Pointer source through the pointer multiplexers to the multiplexer readback. It also verifies that the Load Pointer may be set with any legal value. The test performs the following steps:

1. Transmits the diagnostic instruction that selects the Load Pointer as the Register RAM address source.
2. Writes an address to the Load Pointer Register.
3. Reads the address from the UST ADDR Register and compares.
4. Clears the SOFT CLEAR PIPE Register.
5. Repeats Steps 1–4 for all the addresses (0x0–0x7FF).

PTR

The *Pointers (1–4) Test* option on the Functional Tests Menu 2 tests pointers 1–4. Pointers may be classified into four distinct types:

- Load Pointer
- Pointer 5 (loadable only by the microstore)
- Pointers 1, 2, 3, and 4 (loaded using the command register instruction)
- Immediate pointers 2 and 3 (values taken from D11, I11, and I21)

Each type of pointer functions differently and is tested differently. Only Pointers 1-4 are verified in this test.

This test verifies the proper functioning of the data path from each pointer source through the pointer multiplexers to the multiplexer readback. It also verifies that all four pointers may be loaded with any legal value, and tests the pointer multiplexing by exercising all types of instructions.

The test uses the diagnostic instructions that set up the pointer select for the Register RAM address as pointers 1-4 and hang. Pointers 1-4 are tested within each of the 32 contexts. The following instruction types are used during each context:

- Command register, data-encoded register address

- □ Single-precision, address-encoded register address
- □ Double-precision, address-encoded register address

The test performs the following steps:

1. Clears the SOFT CLEAR PIPE Register.
2. Maps the instruction under test to the appropriate diagnostic instruction.
3. Sets up the context number by writing to the STATE REGISTER Register.
4. Transmits the mapped instruction and writes the data.
5. Reads the data from UST ADDR Register and compares.
6. Repeats Steps 2–4 for each type of instruction.
7. Repeats Steps 2–6 for all 32 contexts.

IM

The *Immed23 Test* option on the Functional Tests Menu 2 verifies the correct setting of Register RAM address sources immed2 and immed3 during IDLE1 and IDLE2.

The test uses the diagnostic instructions that first set up the pointer select for the Register RAM address as immed2 or immed3 then hang. Immed2 and immed3 are tested within each of the 32 contexts. The instruction types used during each context are single- and double-precision short, command register, and extended. The steps performed by this test are the same as those performed by the Pointers (1–4) Test, described previously.

PID

The *Pointer Increment/Decrement Test* option on the Functional Tests Menu 2 exercises the increment/decrement features of pointers 1–4. For each pointer to be tested, the test loads the microstore with micro-instructions which increment or decrement that pointer, select the pointer as the Register RAM address pointer, and hang. The test then transmits a command register instruction to initialize the pointer and walks through the instructions using the clear pipe mechanism. After each instruction, the value of the pointer is checked against the expected value.

PF

The *Pointer Five Test* option on the Functional Tests Menu 2 checks pointer 5 to verify that the pointer can be loaded with any legal value and that it can count up and down through all possible legal values. The test performs the following steps:

1. Fills microstore RAM with hang instructions which load pointer 5 and select pointer 5 for the Register RAM access.
2. Walks through these instructions using the SOFT CLEAR PIPE, verifying pointer value after each micro-instruction.

3. Fills RAM with count-up instructions.
4. Walks through and verifies count of pointer 5.
5. Fills µstore RAM with count-down instructions.
6. Walks through and verifies count of pointer 5.

TIOP

The *TI Operation Test* option on the Functional Tests Menu 2 performs all possible combinations of microcode operations and tests whether they function properly. It performs all of the following types of microcode instructions:

- Command register single-precision
- Command register double-precision
- Extended double-precision
- Extended single-precision
- Single-precision short
- Double-precision short

Each instruction is executed with the user registers loaded with specific values. The expected results are compared with the actual calculated results after execution of each instruction.

TIST

The *TI Status Test* option on the Functional Tests Menu 2 causes the TI chip to produce all possible combinations of status output. Single- and double-precision arithmetic instructions are executed with a variety of operand values stored in the user registers. The expected status result is compared to the actual status in the WSTATUS Register.

PCT

The *Pipeline Control (timing) Test* option on the Functional Tests Menu 2 tests whether or not the pipeline control mechanism can handle a high volume of operations. The test sends several similar operations consecutively and verifies the results. The test is performed with the following types of instructions:

- Command register single-precision
- Command register double-precision
- Extended double-precision
- Extended single-precision
- Single-precision short
- Double-precision short

Each type of instruction is tested 1000 times before a different type is tested.

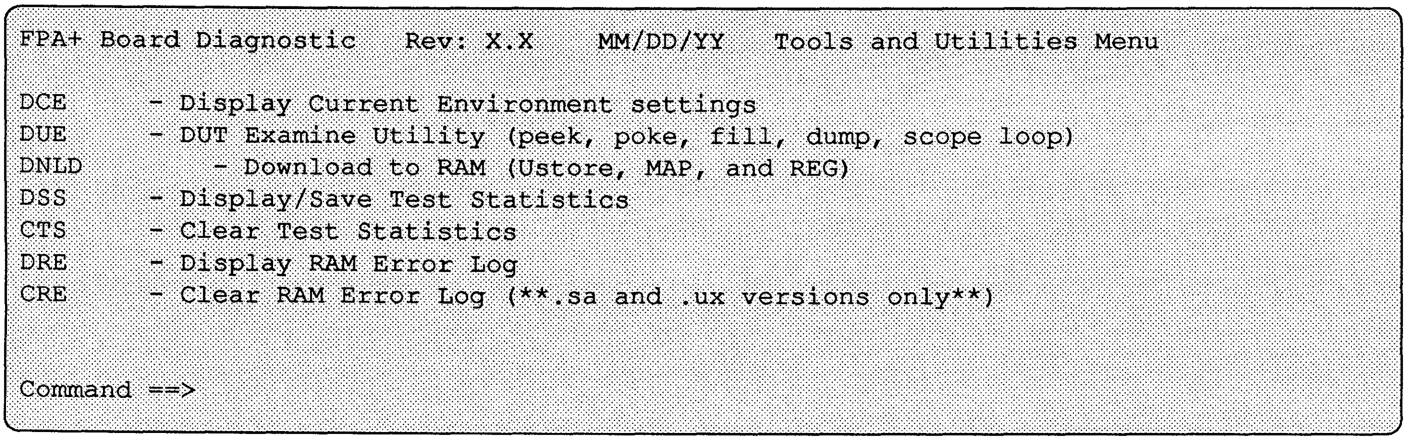
PSM

The *Pipeline State Machine Test* option on the Functional Tests Menu 2 verifies the proper functionality of the instruction pipe state machine by traversing the verifiable paths through the machine. The state of each path is reached and verified by issuing specific instructions and checking the Pipe Status Register and the IERR Register.

Tools and Utilities Menu

T

When you choose **T** from the Main Menu, the Tools and Utilities Menu is displayed:



```
FPA+ Board Diagnostic   Rev: X.X    MM/DD/YY   Tools and Utilities Menu

DCE      - Display Current Environment settings
DUE      - DUT Examine Utility (peek, poke, fill, dump, scope loop)
DNLD        - Download to RAM (Ustore, MAP, and REG)
DSS      - Display/Save Test Statistics
CTS      - Clear Test Statistics
DRE      - Display RAM Error Log
CRE      - Clear RAM Error Log (**.sa and .ux versions only**)


Command ==>
```

This menu provides access to options which are not tests in themselves but which are associated with testing. The tools and utilities provided are particularly useful for debugging.

Display Current Environment Settings

DCE

The *Display Current Environment Settings* option on the Tools and Utilities Menu allows you to display the current values of all parameters and environment variables. For a description of this option and a sample of its output, see the section entitled "Main Menu," at the beginning of this chapter.

DUT Examine Utility

DUE

The *DUT Examine Utility* option on the Tools and Utilities Menu invokes the DUT Memory Examine Utility (also known as the peek-'n-poke utility). This utility allows you to view the contents of a memory location, edit a memory location, and perform a block dump or block fill of a contiguous block of memory.

The utility also provides a reverse Polish notation calculator that performs addition, subtraction, multiplication, division, unary operations, negation, exclusive/inclusive OR, AND, and bitwise shift operations on integers. The results can be stored and used as a part of the utility. The command and value stack are fixed at 100 elements each. Integers may be entered as decimal, octal, or hexadecimal, but all output is hexadecimal.

This option also allows you to execute a tight scope loop on any memory location. Upon detection of a bus error, the program continues. To end the loop, you must terminate the diagnostic.

Online Help provides a list of valid commands and syntax. To request Help within this option, type the **help** command.

Sample output from Help follows:

```
Command ==> DUE
 O.K. help
 <offset>  <count> <pattern|hword lword> [ufill|sfill|rfill|cfill]
   Fills starting at <offset> to <offset> + <count> with <pattern>
   Note : rfill (used for ustore and map RAM) and cfill (for reg RAM)
 <offset> <count> [udump|sdump|rdump|cdump]
   Dumps contents starting at <offset> to <offset> + <count>
   [rdump|cdump,  see above NOTE]
 <offset> [u?|s?|?] Reads contents at <offset> and places the
  Read value on Top of Stack and displays it.  To get value type '.' or '='
 <offset> [ul?|sl?] Super/User Read Scope Loop
 <offset> <value> [u!|s!|!] Writes <Value> into <offset>
 <offset> <value> [ul!|sl!] Super/User Write Scope Loop
 <offset> <value> [ul#|sl#] Super/User Write Read Scope Loop

 Stack Operations :
  $- flush Value Stack,  @ - flush command stack
  '.' or '='  Displays Top of Value stack
  <n> dsp  - displays n elements from the top of value stack
  <value1> <value2> ['+' (add) | '-' (sub) | '/' (div) | '*' (mult)
  | '^' (Xor) | '|' (Ior) | '<<' (Shl) | '>>' (Shr) ]
     <value2> operates on  <value1>  Result goes on top of Stack

  <ESC>, 'exit', 'quit', '^X' -- Exit.
 O.K.
```

As shown at the bottom of the Help screen, you can exit the option by pressing ESC or CTRL-X, or by typing **exit** or **quit** and pressing RETURN.

Download to RAM

DNLD

The *Download to RAM* option on the Tools and Utilities Menu allows you to download a specific part or all of the target RAM and Register File. The target RAM is downloaded using the static information embedded within the diagnostics program. You can download the μstore RAM, mapping RAM, or Register File data:

Parameter	*Description*	*Defaults*
op	Download selection. U (µstore RAM) M (Map RAM) R (Register File)	U and M

Display/Save Test Statistics

DSS

The *Display/Save Test Statistics* option on the Tools and Utilities Menu allows you to display the internally stored statistical information generated by the FPA-Plus Diagnostic tests. The information presented includes the total number of runs for each test in the diagnostic and the total number of failures. At the end of the listing, the accumulated total for all of the tests is displayed.

A brief sample of output from this option follows:

```
---------------------------------------------
FPA-Plus Test Statistics  :
---------------------------------------------
---------------------------------------------
***** 'IERR Reg.' Test #4 ******
[  Total #RUNS= 0  Total #FAILS= 0 ]
---------------------------------------------
***** 'STATE Reg.' Test #5 ******
[  Total #RUNS= 0  Total #FAILS= 0 ]
---------------------------------------------
---------------------------------------------
***** 'Pipeline Control(timing)' Test #47 ******
[  Total #RUNS= 0  Total #FAILS= 0 ]
---------------------------------------------
***** 'r' Test #48 ******
[  Total #RUNS= 0  Total #FAILS= 0 ]
```

Clear Test Statistics

CTS

The *Clear Test Statistics* option on the Tools and Utilities Menu allows you to clear the internally stored statistical information generated by the FPA-Plus Diagnostic tests. This option resets all statistical counters to zero.

Display RAM Error Log

DRE

The *Display RAM Error Log* option on the Tools and Utilities Menu allows you to display the contents of the Exec Error RAM Log.

Clear RAM Error Log

CRE

The *Clear RAM Error Log* option on the Tools and Utilities Menu allows you to clear the contents of the Exec Error RAM Log. This option can only be used if you are running the FPA-Plus Diagnostic as a standalone program, not under the Exec.

9.8. Error Messages

The error messages generated by the FPA-Plus Diagnostic tests are listed in this section, together with their respective tests.

IERR Register Test

If this register test fails, the following message is displayed:

```
Error: IERR register, write ????????, read ????????
```

STATE Register Test

If this register test fails, the following message is displayed:

```
Error: STATE register, write ????????, read ????????
```

IMASK Register Test

If this register test fails, the following message is displayed:

```
Error: IMASK register, write ????????, read ????????
```

LOAD_PTR Register Test

If this register test fails, the following message is displayed:

```
Error: LOAD_PTR register, write ????????, read ????????
```

MODE Register Test

If this register test fails, the following message is displayed:

```
Error: MODE register, write ????????, read ????????
```

WSTATUS Register Test

If a WSTATUS pattern has been generated and the observed status does not match the expected status, the following message is displayed:

```
Error: WSTATUS register, write ????, exp ????, read ????
```

Nack Test

If an error is generated and Nack is expected but does not occur, depending on the error generated, one of the following messages is displayed:

```
Error: Nack (non 32-bit) did not occur, IERR exp ????????, IERR obs ????????
Error: Nack (supervisor only) did not occur, IERR exp ????????, IERR obs ????????
Error: Nack (Register File disable) did not occur, IERR exp ????????, IERR obs ????????
Error: Nack (Microstore RAM disable) did not occur, IERR exp ????????, IERR obs ????????
Error: Nack (illegal address) did not occur, IERR exp ????????, IERR obs ????????
Error: Nack (read write-only address) did not occur, IERR exp ????????, IERR obs ????????
Error: Nack (write to read-only address) did not occur, IERR exp ????????, IERR obs ????????
Error: Nack (execute with load enable) did not occur, IERR exp ????????, IERR obs ????????
Error: Nack (illegal sequence) did not occur, IERR exp ????????, IERR obs ????????
Error: Nack (pipe hung pipe access) did not occur, IERR exp ????????, IERR obs ????????
Error: Nack (pipe hung control access) did not occur, IERR exp ????????, IERR obs ????????
Error: Nack (pipe hung shadow access) did not occur, IERR exp ????????, IERR obs ????????
Error: Nack (retry timeout) did not occur, IERR exp ????????, IERR obs ????????
Error: Nack (illegal control access) did not occur, IERR exp ????????, IERR obs ????????
```

If an error is generated and Nack occurs but status is not as expected, depending on the error generated, one of the following messages is displayed:

```
Error: Nack (non 32-bit) status, IERR exp ????????, IERR obs ????????
Error: Nack (supervisor only) status, IERR exp ????????, IERR obs ????????
Error: Nack (Register File disable) status, IERR exp ????????, IERR obs ????????
Error: Nack (Microstore RAM disable) status, IERR exp ????????, IERR obs ????????
Error: Nack (illegal address) status, IERR exp ????????, IERR obs ????????
Error: Nack (read write-only address) status, IERR exp ????????, IERR obs ????????
Error: Nack (write to read-only address) status, IERR exp ????????, IERR obs ????????
Error: Nack (execute with load enable) status, IERR exp ????????, IERR obs ????????
Error: Nack (illegal sequence) status, IERR exp ????????, IERR obs ????????
Error: Nack (pipe hung pipe access) status, IERR exp ????????, IERR obs ????????
Error: Nack (pipe hung control access) status, IERR exp ????????, IERR obs ????????
Error: Nack (pipe hung shadow access) status, IERR exp ????????, IERR obs ????????
Error: Nack (retry timeout) status, IERR exp ????????, IERR obs ????????
Error: Nack (illegal control access) status, IERR exp ????????, IERR obs ????????
```

Loop Counter Test

If the microstore execution address from the REG UST ADDR Register does not stop at the expected address, the following message is displayed:

```
Error: loop count ????, ustaddr exp ????, ustaddr obs ????
```

Pipeline Status Test

If a status is generated at some instruction map address and the PIPE STATUS Register does not indicate the expected status, the following message is displayed:

```
Error: pipe status, addr ????????, stat exp ????????, stat obs ????????
```

Pipeline Data Test

If upon verifying the pipe data registers, one of the register does not hold the correct data that was written, one of the following messages is displayed:

```
Error: PIPE_ACT_D1 register, write ????????, read ????????
Error: PIPE_ACT_D2 register, write ????????, read ????????
Error: PIPE_NXT_D1 register, write ????????, read ????????
Error: PIPE_NXT_D2 register, write ????????, read ????????
```

Pipeline Instruction Test

If one of the pipe instruction registers does not hold the instruction that was given, one of the following messages is displayed:

```
Error: pipe active instruction ????????, PIPE_ACT_INS read ????????
Error: pipe next instruction ????????, PIPE_NXT_INS read ????????
```

Operand Data Path Test

If a single- or double-precision instruction has been given and the pipe data register or the TEMP Register does not hold the proper data field, one of the following messages is displayed:

```
Error: PIPE_ACT_D1 register, write ????????, read ????????
Error: TMP_D1 register, write ????????, read ????????
Error: PIPE_ACT_D2 register, write ????????, read ????????
Error: TMP_D2 register, write ????????, read ????????
```

TI Data Path Test

If a single-precision TI addition test result is incorrect, the following message is displayed:

```
Error: single precision add result exp ????????, obs ????????
```

If a single-precision TI multiplication test result is incorrect, the following message is displayed:

```
Error: single precision multiply result exp ????????, obs ????????
```

If a double-precision TI addition test result is incorrect, the following message is displayed:

```
Error: double precision add result exp ????????.????????, obs ????????.????????
```

If a double-precision TI multiplication test result is incorrect, the following message is displayed:

```
Error: double precision multiply result exp ????????.????????, obs ????????.????????
```

µstore RAM Quick Test

If one of the microstore RAM locations fails to compare after write/read, the following message is displayed:

```
Error: Microstore RAM, addr ????, write ????????.????????, read ????????.????????
```

µstore RAM Data Pattern Test

If one of the microstore RAM locations fails to compare after write/read, the following message is displayed:

```
Error: Microstore RAM, addr ????, write ????????.????????, read ????????.????????
```

µstore RAM Data March Test

If one of the microstore RAM locations fails to compare after write/read, the following message is displayed:

```
Error: Microstore RAM, addr ????, write ????????.????????, read ????????.????????
```

µstore RAM Address Unique Test

If one of the microstore RAM locations fails to compare after write/read, the following message is displayed:

```
Error: Microstore RAM, addr ????, write ????????.????????, read ????????.????????
```

µstore RAM Address Alternate Test

If one of the microstore RAM locations fails to compare after write/read, the following message is displayed:

```
Error: Microstore RAM, addr ????, write ????????.????????, read ????????.????????
```

µstore RAM Address Disturbance Test

If one of the microstore RAM locations fails to compare after write/read, the following message is displayed:

```
Error: Microstore RAM, addr ????, write ????????.????????, read ????????.????????
```

µstore RAM Address March Test

If one of the microstore RAM locations fails to compare after write/read, the following message is displayed:

```
Error: Microstore RAM, addr ????, write ????????.????????, read ????????.????????
```

Register File Quick Test

If one of the Register File RAM locations fails to compare after write/read, the following message is displayed:

```
Error: Register File, addr ????, write ????????.????????, read ????????.????????
```

Register File Data Pattern Test

If one of the Register File RAM locations fails to compare after write/read, the following message is displayed:

```
Error: Register File, addr ????, write ????????.????????, read ????????.????????
```

Register File Data March Test

If one of the Register File RAM locations fails to compare after write/read, the following message is displayed:

```
Error: Register File, addr ????, write ????????.????????, read ????????.????????
```

Register File Address Unique Test

If one of the Register File RAM locations fails to compare after write/read, the following message is displayed:

```
Error: Register File, addr ????, write ????????.????????, read ????????.????????
```

Register File Address Alternate Test

If one of the Register File RAM locations fails to compare after write/read, the following message is displayed:

```
Error: Register File, addr ????, write ????????.????????, read ????????.????????
```

Register File Address Disturbance Test

If one of the Register File RAM locations fails to compare after write/read, the following message is displayed:

```
Error: Register File, addr ????, write ????????.????????, read ????????.????????
```

Register File Address March Test

If one of the Register File RAM locations fails to compare after write/read, the following message is displayed:

```
Error: Register File, addr ????, write ????????.????????, read ????????.????????
```

Shadow RAM Test

If one of the Shadow RAM context registers fails to compare after write/read, the following message is displayed:

```
Error: Shadow RAM, context ??, addr ????, write ????????.????????, read ????????.????????
```

Micro Store Address Test

When the Microstore Address Register value does not correspond to the address written into the Load Pointer, the following message is displayed:

```
1049  Map Addr Test  Expected μstore addr = eeeeeeee  Actual = aaaaaaaa.
```

NEXT Address Test

When the Microstore Address Register value does not correspond to the expected NEXT micro-PC value, the following message is displayed:

```
1017  NEXT Addr Test  Expected=eeeeeeee Actual=aaaaaaaa.
```

JMP Address Test

When the Microstore Address Register value does not correspond to the expected JUMP address, the following message is displayed:

```
1007  JUMP Addr Test  Expected=eeeeeeee Actual=aaaaaaaa.
```

CALL and RETURN Test

When the Microstore Address Register value does not correspond to the expected CALL or RETURN branch address (Calls from 1st half, Returns from 2nd half), the following message is displayed:

```
1001  Call & Return Test Step 1  Expected Branch Addr=eeeeeeee, Actual=aaaaaaaa.
```

When the Microstore Address Register value does not correspond to the expected CALL or RETURN branch address (Calls from 2nd half, Returns from 1st half), the following message is displayed:

```
1002  Call & Return Test Step 2  Expected Branch Addr=eeeeeeee, Actual=aaaaaaaa.
```

Simple Instruction Test

When the Pipe status resulting from execution of an instruction does not match the expected Pipe status, the following message is displayed:

```
1044  Simple Instruction Test  Address=xxxxxxxx Expected Status=eeeeeeee Actual=aaaaaaaa.
```

Timeout And Retry Test

When a bus error did not occur during a read or write access of an instruction address (xxxxxxxx = an instruction address), the following message is displayed:

```
1018  Time Retry Test  Bus error did not occur while reading/writing xxxxxxxx.
```

When a bus error occurred but the Retry bit was not set in the IERR Register, the following message is displayed:

```
1020  Time Retry Test  Retry bit not set while reading/writing xxxxxxxx. IERR=aaaaaaaa.
```

When an error bit other than the Retry bit was asserted in the IERR Register, the following message is displayed:

```
1021  Time Retry Test  Non Retry errors exist while reading/writing xxxxxxxx. IERR=aaaaaaaa.
```

Load Pointer Test

When the value set in the Load Pointer does not match the actual Register RAM address.

```
1043  Load Ptr Test  Reg RAM address error. Expected=eeeeeeee Actual=aaaaaaaa.
```

Pointers (1 - 4) Test

When the pointer under test does not reflect the expected Register RAM address after execution of a command register, double-precision, or single-precision instruction, the following message is displayed:

```
1036  Ptrs 1-4 Test  Error with Command Reg Inst. Pointer=1-4, Context=0-31,
Expected Addr=eeeeeeee, Actual=aaaaaaaa.
1037  Ptrs 1-4 Test  Error with Double Prec. Inst. Pointer=1-4, Context=0-31,
Expected Addr=eeeeeeee, Actual=aaaaaaaa.
1038  Ptrs 1-4 Test  Error with Single Prec. Inst. Pointer=1-4, Context=0-31,
Expected Addr=eeeeeeee, Actual=aaaaaaaa.
```

Immed23 Test

When the immed2 or immed3 Register under test does not reflect the expected Register RAM address after execution of a command register, double-precision, or single-precision instruction, the following message is displayed:

```
1004  Immed2/3 Test  Error with Command Reg Inst. Reg type=CONST/USER,
Context=0-31, Expected Addr=eeeeeeee, Actual=aaaaaaaa.
1005  Immed2/3 Test  Error with Double Prec. Inst. Context=0-31, Expected
Addr=eeeeeeee, Actual=aaaaaaaa.
1006  Immed2/3 Test  Error with Single Prec. Inst. Context=0-31, Expected
Addr=eeeeeeee, Actual=aaaaaaaa.
```

Pointer Increment/Decrement Test

When the pointer under test does not reflect the expected Register RAM address after incrementing or decrementing, the following message is displayed:

```
1042  Ptr Incr/Decr Test  Error incrementing/decrementing Ptr(1-4).
Expected Val=eeeeeeee Actual=aaaaaaaa.
```

Pointer Five Test

When Pointer 5 does not reflect the Register RAM address which should have been loaded, the following message is displayed:

```
1041  Pointer 5 Test  Invalid load value. Expected Val=eeeeeeee Actual=aaaaaaaa.
```

When Pointer 5 does not reflect the expected Register RAM address after incrementing, the following message is displayed:

```
1040  Pointer 5 Test  Error incrementing. Expected Val=eeeeeeee Actual=aaaaaaaa.
```

When Pointer 5 does not reflect the expected Register RAM address after decrementing, the following message is displayed:

```
1039  Pointer 5 Test  Error decrementing. Expected Val=eeeeeeee Actual=aaaaaaaa.
```

Lock Test

When a Shadow Register is locked unexpectedly, the following message is displayed:

```
1014  Lock Test  Unexpected Lock Error. Inst Addr=xxxxxxxx, Shadow Reg Addr=
xxxxxxxx, IERR Val=aaaaaaaa.
```

When a fatal error occurs, the following message is displayed:

```
1015  Lock Test  Pipe Not HUNG When Buserr. Inst Addr=xxxxxxxx, Shadow Reg
Addr=xxxxxxxx, IERR Val=aaaaaaaa.
```

When a Shadow Register did not lock when expected, the following message is displayed:

```
1019  Lock Test  No Lock When Expected. Inst Addr=xxxxxxxx, Shadow Reg Addr=
xxxxxxxx, IERR Val=aaaaaaaa.
```

TI Operation Test

When the calculated result from an instruction does not match the expected result (iiiiiii = instruction under test), the following message is displayed:

```
1022  TI Operation Test  Command Reg DP iiiiiiii. Expected MSW/LSW=eeeeeeee,
actual=aaaaaaaa.
1023  TI Operation Test  Command Reg SP iiiiiiii. Expected result=eeeeeeee,
actual=aaaaaaaa.
1027  TI Operation Test  DP iiiiiiii. Expected MSW/LSW=eeeeeeee,
actual=aaaaaaaa.
1028  TI Operation Test  Extended DP iiiiiiii. Expected MSW/LSW=eeeeeeee,
actual=aaaaaaaa.
1029  TI Operation Test  Extended w/2 SP operands iiiiiiii. Expected
result=eeeeeeee, actual=aaaaaaaa.
1030  TI Operation Test  Extended SP iiiiiiii. Expected result=eeeeeeee,
actual=aaaaaaaa.
1034  TI Operation Test  SP iiiiiiii. Expected result=eeeeeeee,
actual=aaaaaaaa.
```

When the expected status resulted from a compare instruction does not match the expected status, the following message is displayed:

```
1024  TI Operation Test  DP Compare 0 with 1. Expected ABACUS/Decoded
status=eeeeeeee, actual=aaaaaaaa.
1025  TI Operation Test  DP Compare 1 with 1. Expected ABACUS/Decoded
status=eeeeeeee, actual=aaaaaaaa.
1026  TI Operation Test  DP Compare Mag 2 with 3. Expected ABACUS/Decoded
status=eeeeeeee, actual=aaaaaaaa.
1031  TI Operation Test  SP Compare 0 with 1. Expected ABACUS/Decoded
status=eeeeeeee, actual=aaaaaaaa.
1032  TI Operation Test  SP Compare 1 with 1. Expected ABACUS/Decoded
status=eeeeeeee, actual=aaaaaaaa.
1033  TI Operation Test  SP Compare Mag 2 with 3. Expected ABACUS/Decoded
status=eee, actual=aaaaaaaa.
```

TI Status Test

When the calculated result from an addition or division instruction does not match the expected result, the following message is displayed:

```
1000  TI Status Test  (ALU) Expected status=eeeeeeee, Actual=aaaaaaaa.
```

When the calculated result from a multiplication instruction does not match the expected result, the following message is displayed:

```
1016  TI Status Test  (MULT) Expected status=eeeeeeee, Actual=aaaaaaaa.
```

Jump Conditions Test

When a branch condition is not met and the microroutine halts prematurely, the following message is displayed:

```
1008  Branch Cond Test  JMP/JGT/JGE/JEQ/JLE/JLT/JNE/JNOTIERR/JTIERR error. UST addr=xxxxxxxx.
```

When the microroutine halts at an unknown location, the following message is displayed:

```
1009  Branch Cond Test  Unexpected UST addr=xxxxxxxx.
```

Pipeline Control (timing) Test

When the calculated result from an instruction does not match the expected result (iiiiiii = instruction under test), the following message is displayed:

```
1045  TI Timing Test  CMD REG DP (e**n-1). Expected MSW/LSW=eeeeeeee Actual=aaaaaaaa.
1046  TI Timing Test  EXT DP iiiiiiii. Expected MSW/LSW=eeeeeeee Actual=aaaaaaaa.
1047  TI Timing Test  DP iiiiiiii. Expected MSW/LSW=eeeeeeee Actual=aaaaaaaa.
1048  TI Timing Test  SP iiiiiiii. Expected Result=eeeeeeee Actual=aaaaaaaa.
```

Pipeline State Machine Test

When the instruction pipeline state, as verified through the Pipe Status Register and/or the IERR Register, is not as expected after execution of specific instructions, the following message is displayed:

```
1053  Pipe State Machine 1/2 Test.  Path: nnnn. Pipe not clear after inst.
1054  Pipe State Machine 1/2 Test.  Path: nnnn. Exp Pipe Status=0xeeeeeeee, Act=0xaaaaaaaa.
1055  Pipe State Machine 1/2 Test.  Path: nnnn. Buserror did not occur.
1056  Pipe State Machine 1/2 Test.  Path: nnnn. Retry Bit not set. IERR = xxxxxxxx.
1057  Pipe State Machine 1/2 Test.  Path: nnnn. Hung Bit not set. IERR = xxxxxxxx.
1058  Pipe State Machine 1/2 Test.  Path: nnnn. Exp MSW/LSW result=eeeeeeee, Act=aaaaaaaa.
```

9.9. Glossary

Exec	The SunDiagnostic Executive Operating System.
FPA-Plus	Floating Point Accelerator-Plus board.
Mflop	Million floating point operations per second.
RAM	Random Access Memory.

10

Floating Point Unit Diagnostic

10

Floating Point Unit Diagnostic

10.1. Overview

The Floating Point Unit (FPU) consists of a Floating Point Controller 40281, Floating Point Arithmetic Logic Unit (Weitek 1065), and a Floating Point Multiplier (Weitek 1164), or GNUFPC Floating Point Controller and a TI8847 ALU. The FPU is an optional chip set that performs floating point operations for the SF9010IU (CPU). The FPU talks to the IU over a dedicated bus. The FPC controls the interface to the IU, the memory system, and the Weitek/TI chips. This unit resides on the Sun-4 CPU board. The FPU supports the operations conforming to the IEEE standard 754 with full 32-bit (single), 64-bit (double), and 80-bit (extended) data types.

The FPU can be divided into three functional areas. They are 1) Registers, 2) Instructions, and 3) the Weitek/TI chip set. The FPU diagnostic tests all three of these functional areas

In particular, the memory capability and functionality of the FPU registers is tested. The functionality of all the instructions is checked and finally, the Weitek/TI chip set operations and status are tested.

In terms of run time restraints, there are two tests, consuming different amounts of time. One is a quick test, which gives at least 95% confidence level and will take the least time (less than 2 minutes.) The second test is the default test, which will perform all the tests and may take more time than the quick test. The default test yields a confidence level of greater than 98% percent.

You may interrupt the execution of FPU tests after each test is executed, and in some cases in between test execution. Adequate on-line help is available.

10.2. Hardware Requirements

The system under test must have at least:

1. A working Sun-4 CPU Board with a FPU.
2. A video monitor.
3. A keyboard.
4. A boot device (i.e. local disk, local tape, or remote disk over Ethernet).

10.3. Diagnostic Functional Description

General attributes of the FPU Diagnostic are discussed first, followed with a discussion of each module.

The FPU Diagnostic permits individual testing of each functional section of the FPU. In addition, commands are provided to execute the tests in a continuous sequence.

The user interface of the FPU Diagnostic consists of a Main Menu with three sub-menus. A help option for each menu describes more detailed command syntax.

If you want to run multiple passes of any test, use the SunDiagnostic Executive `pass=` argument in the command line to invoke this diagnostic, or use the `pass=` selection from the Options Menu described in *Chapter 2*. Do not add a `pass=` argument to the menu selections shown here.

10.4. Modular Description Of The Diagnostic

Each menu is shown, followed with a description of the menu choices and any arguments that may be used with the commands.

Main Menu

The main menu allows access to the sub-menus. It also contains the "all" tests command and the "quick" test command. When the Main Menu is first displayed, a status message tells you which Floating Point Processor is installed in the system (TI or Weitek). The message looks something like this:

```
***FP processor type: TI or Weitek ***
```

```
Sun-4 FPU Diagnostic  REV.xx MM/DD/YY  Main Menu

R - Registers test menu
I - Instructions test menu
F - FPP test menu
O - Other test
A - All tests
D - Default tests
Q - Quick tests
T - halt on error (1)
S - Scope loop on error
? - Help
***FPU type: FAB4 and Weitek

Command ==>
```

R

Option `R` in the main menu brings up the *Register* test sub-menu, described later.

I

Option `I` in the main menu brings up the *Instructions* test menu, described later.

F

Option F in the main menu brings up the *FPP* test sub-menu, described later.

O

Option O executes a sequence of double floating point operations. The first pass halts on the first error encountered. The second pass repeats the same test 500 times and reports the total number of errors.

The expression being evaluated is

x*z*(double)w+q

A

If option A is selected, the program sequentially executes *all* the tests from all the sub-menus.

D

Option D sequentially executes the default sequence of the sub-menus.

Q

If option Q is selected, the program sequentially executes some of the tests from all the sub-menus.

T HA=*value (0 or 1)*

If option T is selected, the program changes the halt on error flag accordingly. The default is that the program will stop on error. The argument is either "0" or "1". A zero argument means the program will not halt on error. A "one" argument means the program will stop on error (which is also the default).

S Loop=*value (0 or 1)*

If option s is selected, the program loops the given test command sequence for the given number of times. The *value* argument may be either "0" or "1". A zero argument means the program will not loop on error. A "one" means the program will repeat the test that failed until you type **c**.

?

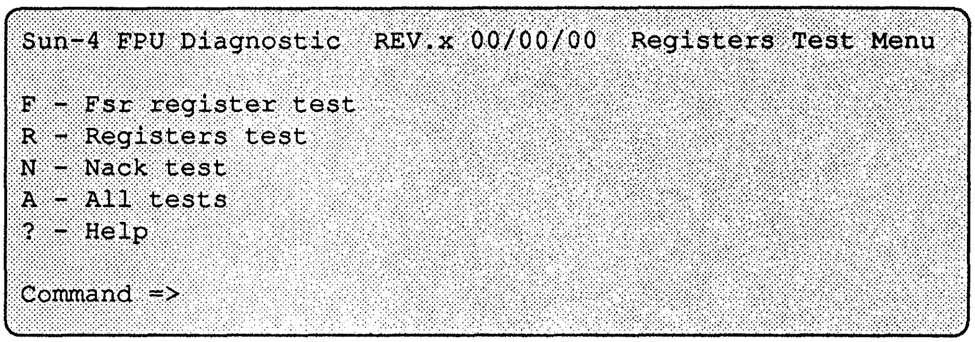
If option ? is selected, the program prints the complete command syntax for the menu being displayed.

***** FPU type:**

Indicates the type of FPU installed. Esc.

Registers Test Menu

This menu offers all the register related tests as well as an *all tests* option that executes all the tests in this menu.



```
Sun-4 FPU Diagnostic  REV.x 00/00/00  Registers Test Menu

F - Fsr register test
R - Registers test
N - Nack test
A - All tests
? - Help

Command =>
```

F

If option F is selected, the program will execute the *fsr register test.* In this test, the program writes rotating ones for all the writable bits of the `fsr` register and then reads them back.

R *TEst=number register_number pattern*

If option r is selected, the program executes the *registers* test. This test contains four sub-tests and the path between memory and the registers is also tested. All 32 registers are tested. The first test writes five different default patterns (0x00000000, 0x55555555, 0xAAAAAAAA, 0xCCCCCCCC, 0xFFFFFFFF) to the registers and reads them back. The second test writes all rotating ones to the registers and reads them back. The third test, which takes more time, writes all possible floating values to the registers and reads them back. For the fourth test, you may supply the register number to be tested and a pattern with which the test is to be executed.

The *test* argument specifies which sub-test to be executed. Possible arguments are:

`test = 0`	*Run tests 1, 2, and 3*
`test = 1`	*Pattern test*
`test = 2`	*Rotating ones test*
`test = 3`	*Burn-in test*
`test = 4`	*User defined pattern test*

The default is to execute the Pattern test. The *register number* and *pattern* arguments apply to Test 4 only.

N

If option N is selected from the Register Tests menu, the program executes the *Nack* (Negative Acknowledgement) test. In this test, all the possible exceptions are created and checked. The exceptions tested are the IEEE exception, Unfinished exception, and Unimplemented exception.

A

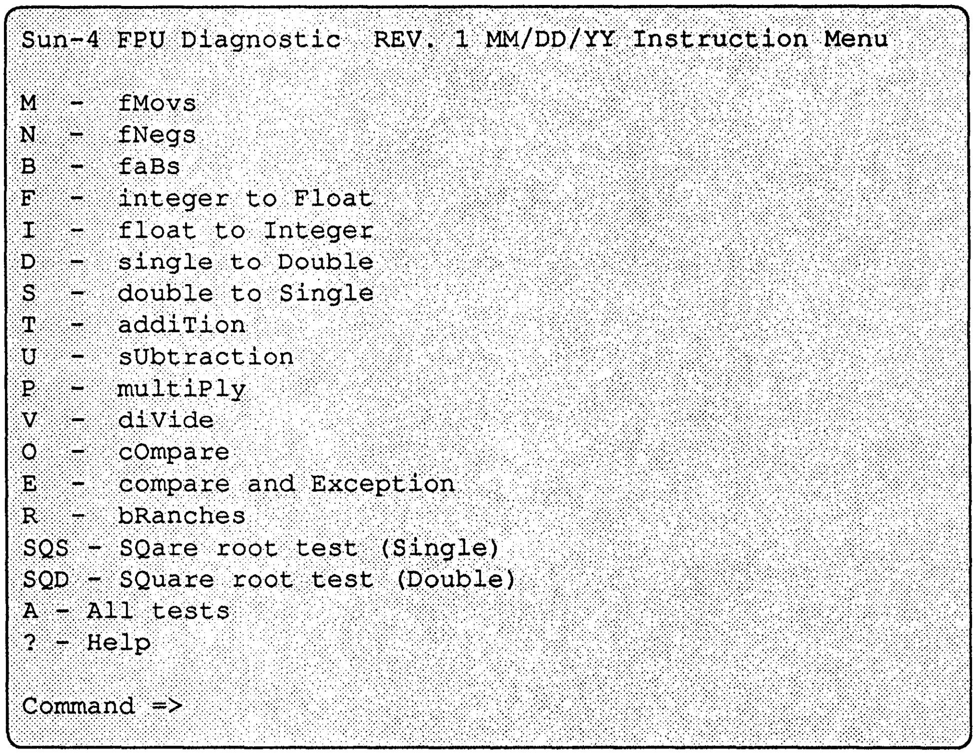
If option A is selected, the program sequentially executes all the tests in this menu.

?

If option ? is selected, the program prints the complete command syntax for each of the Registers Test Menu selections. To escape back to the previous menu, press Esc.

Instructions Test Menu

This menu contains tests for all the FPU chip instructions. There is an *all* command that sequentially executes all the tests in this menu.



```
Sun-4 FPU Diagnostic  REV. 1 MM/DD/YY Instruction Menu

M   -  fMovs
N   -  fNegs
B   -  faBs
F   -  integer to Float
I   -  float to Integer
D   -  single to Double
S   -  double to Single
T   -  addiTion
U   -  sUbtraction
P   -  multiPly
V   -  diVide
O   -  cOmpare
E   -  compare and Exception
R   -  bRanches
SQS - SQare root test (Single)
SQD - SQuare root test (Double)
A - All tests
? - Help

Command =>
```

M *DAta=*

If option M is selected, the program executes the *fmovs* test. In this test the program uses the `fmovs` instruction to move data between registers. The data is written to register 0 and from register 0 to register 1, and so on. Then the data read from register 30 is read and compared. This test is looped for different values.

The optional argument *DAta=* is followed with the "data" value to be written into the registers.

N *NTest=*

If option N is selected, the program executes *fnegs* test, which uses the `fnegs` instruction to negate the data. There are two sub-tests. In the first test a positive value is given and tested against the negated value, then, in the second test, a negative value is given and tested against the negated value. This test will be looped for different values.

The *ntest=* argument specifies which test is to be done, as follows:

`ntest = 0`	*Both test 1 and 2*
`ntest = 1`	*Positive to Negative test*
`ntest = 2`	*Negative to Positive test*

The default is zero, which executes both positive-to-negative and negative-to-positive test.

B

If option `B` is selected, the program executes the *fabs* test. In this test the program uses the `fabs` instruction to get the absolute data. A negated value is sent and compared against the absolute value. This test will be looped for different values.

F *PRecision=*

If option `F` is selected, the program executes the *integer to float* test. In this test, the program uses the `fints` and `fintd` instructions. You may test either single precision or double precision. The test checks the floating number returned.

The argument *PRecision=* may be followed with one of the choices shown below:

`precision = 0`	*Both single and double precision tests*
`precision = 1`	*Single precision*
`precision = 2`	*Double precision*

The default is zero, which executes both single and double precision tests.

I *PRecision=*

If the option `I` is selected, the program executes the *floating number to integer number* test. In this test, the program uses the `fitos` and `fitod` instructions. You may test either single precision or double precision.

The *pr=* argument may be followed with one of the `precision=` choices shown under the **F** test description.

D

If option `d` is selected from the Instruction menu, the program executes the *single to double precision* test. In this test the program converts the single to double precision values by using the `fstod` instruction.

S

If option `s` is selected, the program executes the *double to single precision* test. In this test the program converts the double to single precision values by using the `fdtos` instruction.

T *PRecision=*

If option `t` is selected, the program executes the *addition* test. In this test the program adds two different values by using `fadds` and `faddd` instructions. You can test either single precision or double precision, or both.

The *PRecision=* argument may be followed with one of the choices shown under the **F** test description.

U *PRecision*

If option u is selected, the program executes the subtraction test. In this test the program subtracts one from the different values, using `fsubs` and `fsubd` instructions. You may test either single or double precision.

The *PRecision=* argument may be followed with one of the choices shown under the **F** test description.

P *PRecision=*

If option p is selected, the program executes the multiply test. In this test the program multiplies two different values by using `fmuls` and `fmuld` instructions. You may test either single precision and/or double precision.

The *PRecision=* argument may be followed with one of the choices shown under the **F** test description.

V *PRecision=*

If option v is selected, the program executes the divide test. In this test, the program divides two different values by using `fdivs` and `fdivd` instructions. You may test either single or double precision.

The *PRecision=* argument may be followed with one of the choices shown under the **F** test description.

O *PRecision=*

If option o is selected, the program executes the compare test. In this test the program compares two different numbers, using `fcmps` and `fcmpd` instructions. The program checks the bits set in the Floating Point Status Register. You may test either single or double precision.

The *PR=* argument may be followed with one of the choices shown under the **F** test description.

E *PRecision*

If option e is selected, the program executes the *compare and exception* test. In this test the program compares two different numbers, using `fcmpes` and `fcmped` instructions. The program checks whether or not the exception is occurring when an unordered number is given as an operand. You may test either single or double precision.

The *PR=* argument may be followed with one of the `precision=` choices shown under the **F** test description.

R *BTest=*

If option r is selected from the Instruction menu, the program executes the branching test, which tests all the floating branching instructions. There are two sub-tests within this test. One does the branching and other one checks the no-branching.

The argument *BTest=* may be followed with one of these choices:

`test = 0`	*both branching and no-branching tests*
`test = 1`	*branching test*
`test = 2`	*no-branching test*

The default is zero, which executes both branching and no-branching tests.

SQS

If the `sqs` option is selected from the Instruction menu, the program executes the single precision square root test. In this test, the program takes the single precision square root of 20 different values, using the `fsqrts` instruction.

SQD

The `sqd` option executes the double precision square root of 20 different values, using the `fsqrtd` instruction.

A

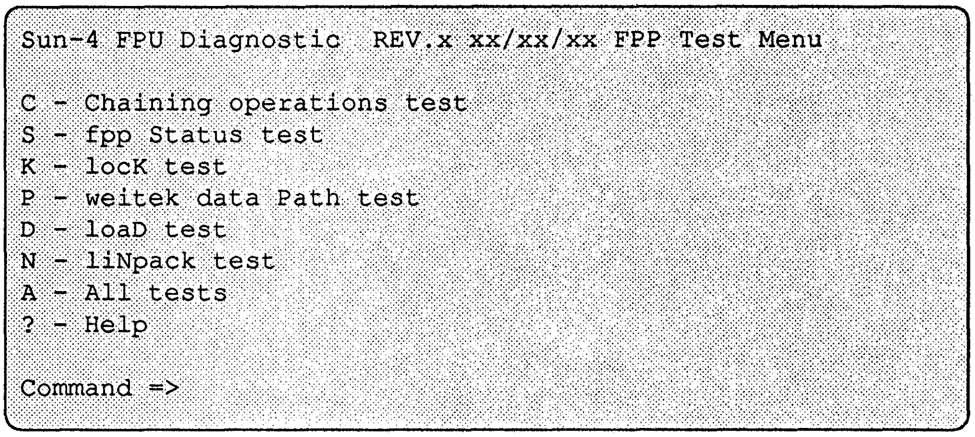
If option `a` is selected, the program sequentially executes all the tests given in this menu.

?

If option `?` is selected, the program prints the complete command syntax for each Instruction test menu choice. To escape back to the previous menu, press Esc.

FPP Test Menu

This menu offers FPP (Weitek/TI) related tests. The tests execute the Weitek/TI chip ALU/Multiplier set.



```
Sun-4 FPU Diagnostic  REV.x xx/xx/xx FPP Test Menu

C - Chaining operations test
S - fpp Status test
K - locK test
P - weitek data Path test
D - loaD test
N - liNpack test
A - All tests
? - Help

Command =>
```

C *PRecision=*

When option `c` is selected, the program executes the *chaining operations* test, explained later in this text.

S *STest=*

If the option `s` is selected, the program executes the *FPP status* test. In this test, the program creates all the possible IEEE exceptions, and checks whether or not they occurred. There are two sub-tests. One that uses =`NX`bit and other that uses =`NX`bit

The argument *STest=* may be followed with one of the arguments shown below:

`stest = 0`	*Both test 1 and test*
`stest = 1`	*Status test with NX bit = 0*
`stest = 2`	*Status test with NX bit = 1*

K

If the option `k` is selected, the program executes the lock test. In this test, an instruction that will create an exception is sent to the FPU, then the queue is checked, and the program checks to see if the given instruction address is present in the floating point queue register.

P *PRecision*

If the option `P` is selected, the program executes the *Weitek Data Path* test. In this test, the data path is checked, starting from the memory to the registers and then to the Weitek chips. To do this, the add and multiply commands for all possible values are executed.

The *PRecision=* argument may be followed with one of the values shown below:

precision = 0 : Both single and double precision tests
precision = 1 : Single precision
precision = 2 : Double precision

The default is zero, which executes both single and double precision tests.

D

If the option `D` is selected, the program execute the *load* test. In this test, the program executes instructions continuously, to check for problems caused by heavy use of the FPU. The test does the following:

a). Ten add single precision instructions,
b). Ten multiply single precision instructions,
c). Five add double precision instructions,
d). Five multiply double precision instructions.

These four steps are done 1000 times, and most of the registers are used.

N

Option `N` executes the linpack test suite.

A

If option `A` is selected, the program sequentially executes *all* the tests given in this menu.

?

If the *help* option is selected, the program prints the complete command syntax for the Weitek test menu. To escape back to the previous menu, press Esc.

10.5. Glossary

FPC	Floating Point Controller
FPU	Floating Point Unit
IU	Integer Unit (Sun-4 CPU)
FPP	Floating Point Processor (Weitek or TI)

11

Color Graphics Diagnostic

11

Color Graphics Diagnostic

11.1. Introduction to the Color Graphics Boards

This diagnostic program, which runs under the Exec, tests the Sun Color-2, Color-3, or Color-5 Graphics Boards. The Sun Color Graphics boards are VMEbus slaves that provide a megabyte of RAM (the frame buffer), and circuits to manipulate the data in this RAM. The frame buffer contains the image that appears on the video monitor. The Color-2 and Color-3 boards are very similar functionally, except that the Color-2 board has zoom and pan logic while the Color-3 and Color-5 boards do not. The Color-3 and Color-5 boards also have optional double buffering. The Sun-2 board, introduced with the Sun-2 product line, runs on both Sun-2 and Sun-3 workstations. It has four BNC connectors on the back of the board. The newer, CG3 and CG5 boards have 5 BNC connectors on the back.

11.2. What's In This Chapter

After an overview of the diagnostic and set-up information, the Main Menu and each Sub-Menu are discussed, with error message descriptions at the end of the chapter.

11.3. Objectives

The objective of this generic Color Graphics Diagnostic is to ensure that the Color-2, Color-3 and Color-5 Graphics boards work correctly. This diagnostic covers these functional areas of the Color Graphics Board:

Registers — the functional control registers on the color boards.

Color Maps — the 256 by 24-bit ECL RAM.

Frame Buffer — the on-board 1 Mbyte memory.

RasterOp Units — the raster chips.

Direct Memory Access Double Buffer Memory — optional double buffered memory.

Color Board outputs (DACs) — the outputs (digital-to-analog converters) that drive the monitor.

11.4. Required Equipment

The color diagnostic displays patterns on the color monitor, therefore it is best to run the diagnostic from a monochrome monitor OR from a terminal connected to Serial Port A.

You may also run the diagnostic from the color monitor using this command-line option from the Exec menu:

```
color -b
```

The -b option re-sets the color board and blanks the center of the screen AFTER EVERY TEST, allowing the menus to be seen over the possible test patterns on the color monitor. This option is not recommended since many of the diagnostic messages will NOT be seen during the test.

In order to use the Color Graphics Diagnostic, the system being tested must have:

1. A Sun CPU Board.
2. A Sun Color Board (either Color-2, Color-3 or Color-5).
3. A Sun Color Monitor.
5. A Sun Keyboard.
6. A boot device (i.e. local disk, local tape, or remote disk over Ethernet).

11.5. 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.

The Color Diagnostic provides for individual testing of each functional section of the color board, or for running the tests in a continuous sequence.

The user interface is menu driven, meaning that a menu will be displayed from which you select the tests you want to run. There are seven sub-menus available from the Test Menus.

11.6. Set-Up

The Color board address should be set on a four-megabyte boundary. The board should also be configured for 4 Mbyte decoding, meaning that the board occupies from 4 to 8 Mbytes in the VME space.

11.7. Modular Description Of The Diagnostic

The diagnostic is divided into Auto and Manual tests. The Auto test provides good test coverage to every part of the hardware. The manual test contains most of the sub-tests that the auto test runs, as well as scope loops and debugging tools for isolating hardware problems.

The Main Menu

The main menu has four options and allows access to the manual tests' sub-menus. The type of board installed, either Color-2, Color-3 or Color-5, will appear in the upper left corner of the screen. The menu items are the same for both board types. The options are described following the example of the Main menu, below.

```
Colornumber Diagnostic(SINGLE BUF) Rev: X  MM/DD/YY Main Menu

Auto test           Test all hardware
Manual test         Select any part of hardware
Change              Change type of board(color-2 3 or 5)
Switch              Switch type of buffer(double buf boards)

Command ==>
```

The title at the top of the menu shows either `SINGLE` or `DOUBLE BUF`, depending on whether the board is single or double buffered. The main menu selections are described in the following paragraphs.

A

If you type **a**, the *auto test* is invoked, which executes all the tests available for this diagnostic. 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 automatic diagnostic repeats until it reaches the specified number of passes, then returns to the main menu. The specific tests executed in Auto Mode are described in the section that follows, and under "Auto Register Test Details" at the end if this chapter.

M

The *Manual* selection from the Main Menu enables the manual mode of operation and brings up the Manual Menu. A prompt will ask whether or not to use the default settings. The *manual test* allows you to test each part of the hardware. If the Auto test discovers an error, you may want to re-run the specific test from the Manual Test menu.

C

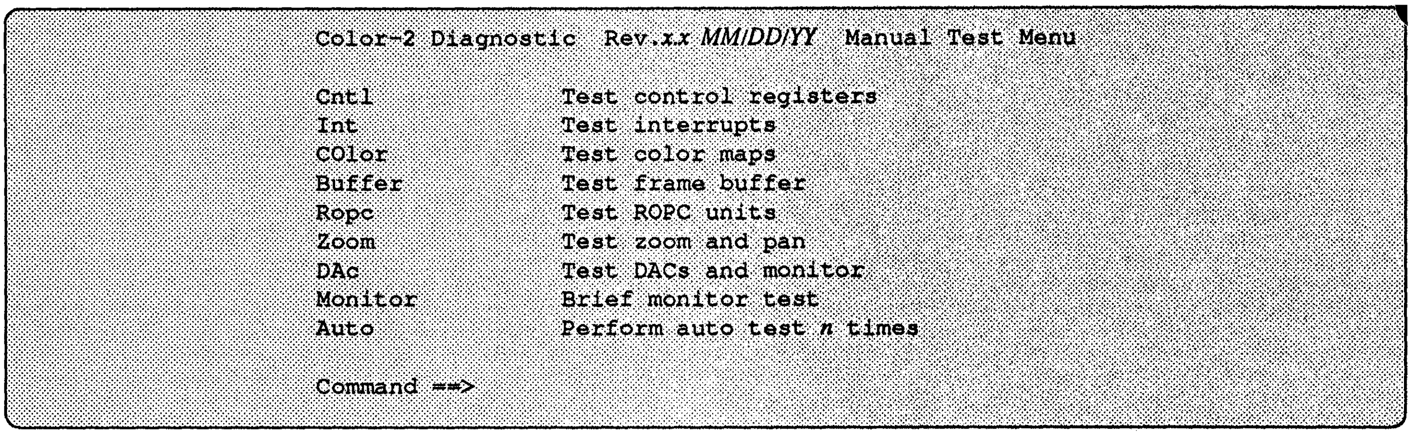
The *Change* option from the Main Menu changes the default board type setting. The Color-2 diagnostic can be changed to the Color-3 or Color-5 diagnostic and back again. The diagnostic determines the correct board type upon initializing the board. This menu is useful for manual testing.

S

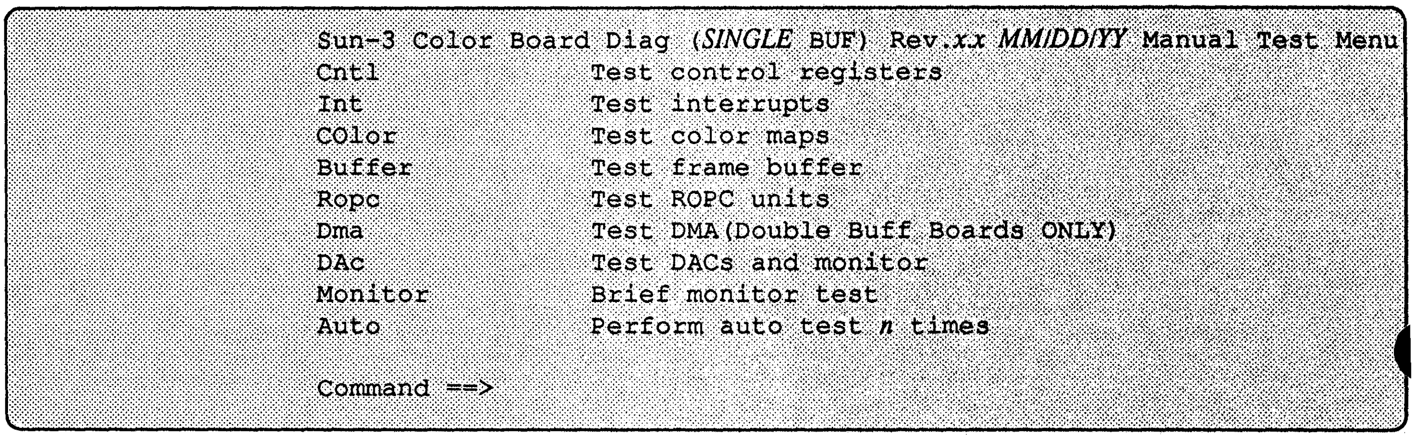
The *Switch* option from the Main Menu switches the current default setting of the buffer type. Single buffer can be switched to double buffer and back again. This option is only valid for Sun Color-3 and Color-5 Graphics boards, and is useful for manual testing.

Manual Test Menu

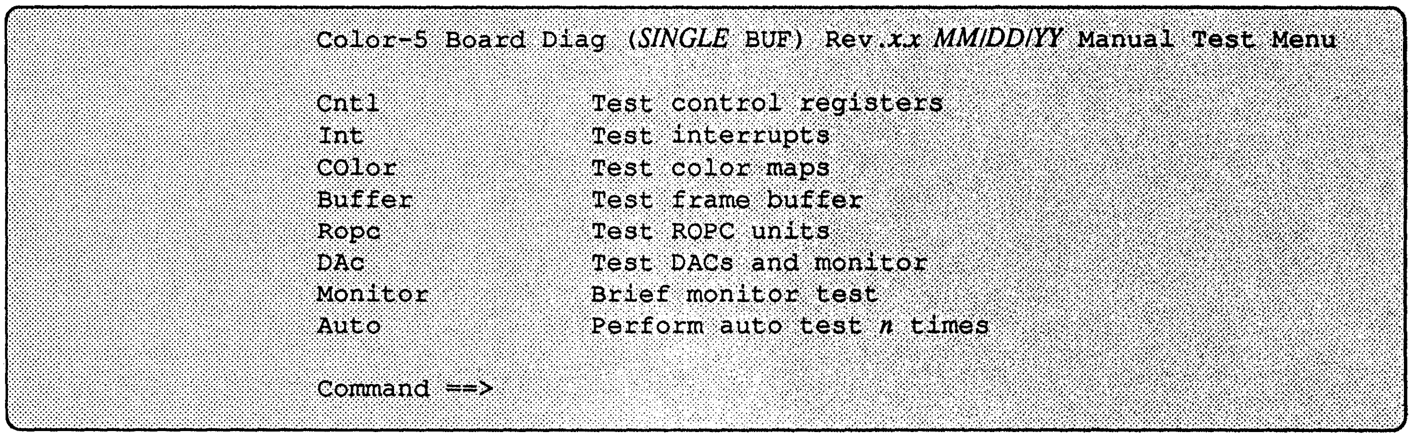
Selecting `Manual` from the main menu displays the Manual Test Menu options. Depending on the type of board installed, either the Color-2, Color-3 or Color-5 manual test menu will be displayed as shown below in the following examples: You may select any item on the menu or press [Esc] to back up to the main menu.



```
Color-2 Diagnostic  Rev.x.x MM/DD/YY  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 n times

Command ==>
```



```
Sun-3 Color Board Diag (SINGLE BUF) Rev.x.x MM/DD/YY 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(Double Buff Boards ONLY)
DAc                 Test DACs and monitor
Monitor             Brief monitor test
Auto                Perform auto test n times

Command ==>
```



```
Color-5 Board Diag (SINGLE BUF) Rev.x.x MM/DD/YY Manual Test Menu

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

Command ==>
```

The following text describes the commands that appear on the Manual Test Menus.

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

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

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

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

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

`Dma`
: Type `d` to select the *DMA window test.* Both pattern write/read and visual tests are executed in this test. If double buffering is installed, a DMA test menu is displayed. Note that this menu selection is only valid for Color-3 and boards.

`Zoom`
: The *Zoom* option brings up the zoom and pan test and sub-menu. This menu selection is only valid for Color-2 boards.

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

`Monitor`
: Type `m` to select the *auto monitor test*, from the Manual Menu. This test produces several different patterns on the screen, to test the convergence, linearity, focus, and other monitor parameters. Use this test to isolate a problem to the output DACs or the display monitor. There is no menu for this item.

`Auto` *n times*
: Type `a` to select the *auto test* from the Manual Menu. It will test all parts of the hardware automatically, once. There is no sub-menu for this test. This test is similar to the auto test selection on the main menu. For detailed descriptions of the tests executed when you select Auto, refer to Section 11.8 near the end of this chapter.

As shown, there is only one difference between the color-2, color-3 and color-5 manual test menus: the DMA and ZOOM tests.

Sub-Menus

The following text describes each sub-menu available from the test menus.

Register Test 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.

Since the registers on the color-2, color-3 and color-5 boards are different, different register test menus appear, depending on the type of board under test. Examples of the register test menus follow:

```
[Color-2] Diagnostic  Revx.x MM/DD/YY 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

Command ==>
```

```
[Color-3] Diagnostic(SINGLE BUF)Rev:x.x MM/DD/YY 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 =>
```

```
[Color-5] Diagnostic(SINGLE BUF)Rev:x.x MM/DD/YY Register Test
Menu

Status           Status register
Perplane         Per_Plane register
Frame            Read-only Frame Count register
Double           Double-buffering register
Interrupt        Interrupt register
BIDreg           ROPMode Register
SR2reg           2nd Status (GP2) Register
IV2reg           2nd Interrupt Vector (GP2) Register (8-bit)

Command =>
```

The following paragraphs describe the control register choices shown in the previous examples.

`S`

The *Status* option executes the status register test and brings up the register test sub-menu. Both the color-2 and color-3 status registers are identical, and only the first nine bits (D0 - D8) are used.

`P`

The *Per_Plane* option executes the per-plane mask register test and brings up a register test sub-menu. This register restricts frame buffer access to the selected bit-planes of the frame buffer memory.

NOTE *The following four commands are available from the Sun-2 Register Test Menu only.*

`W`

The *Word* option executes the word-pan register test and brings up the register test sub-menu. The word-pan register is one of four registers containing the pan and zoom information. The selection is only valid for the Color-2 board.

`PI`

The *Pixel* option executes the pixel-pan register test and brings up a register test sub-menu. The pixel-pan register is one of four registers containing the pan and zoom information. This register sets the origin of the region being displayed. The selection is only valid for the Color-2 board.

`L`

The *Line* option executes the line offset and zoom register test and brings up a register test sub-menu. The zoom register is one of four registers containing the pan and zoom information. 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. The selection is only valid for the Color-2 board.

`V`

The *Variable* option executes the variable zoom register test and brings up the register test sub-menu. The variable zoom register is one of four registers containing the pan and zoom information. This register specifies the line number on the screen that is the lower limit of the zoom register. The selection is only valid for the Color-2 board.

`F`

The *Frame* option executes the frame count register test and brings up the register test sub-menu. The frame count register is read-only. The selection is only valid for the Color-3 and Color-5 board.

`B`

The Color-3 board *Base* option executes the DMA window base register test and brings up the register test sub-menu. The base register points to the beginning of the DMA window and is write-only.

`W`

The *Width* option executes the DMA window width register test and brings up the register test sub-menu. The DMA width register contains a value that is 1/16th the width of the DMA window, and it is write-only. The selection is only valid for the Color-3 board.

`D`

The *Double* option executes the double-buffering register test and brings up the register test sub-menu. The double-buffering register is sixteen bits wide and is cleared on bus reset. The selection is only valid for the Color-3 and Color-5 board.

`I`

The *Interrupt* option executes the interrupt register test and brings up the register test sub-menu. This register contains an eight-bit interrupt vector. The color board sends this vector to the VMEbus during a vectored interrupt. The selection is only valid for the Color-3 and Color-5 board.

NOTE *The next four Register Test menu choices are available for Color-5 boards only.*

BID

The *BIDreg* option executes the Board ID Register Test and brings up a register test sub-menu. The board ID register is an 8-bit, read-only register. The CG5 board ID value will always be 0x01. This menu selection is valid only for the CG5 board.

ROP

The *ROPreg* option executes the ROPmode register test and brings up a register test sub-menu. This register allows write access to the three ROP-mode bits of the status register. Bits 0-2 and 6-15 of the ROPmode register are not used. Bits 3-5 are the same as bits 3-5 of the status register. This menu selection is valid only for the CG5 board.

SR2

The *SR2reg* option executes the second status register test and brings up a register test sub-menu. The second status register contains information about the P2 bus (bit 0) and the GP2 (bits 14-15). Bits 1-13 are not used. This menu selection is valid only for the CG5 board.

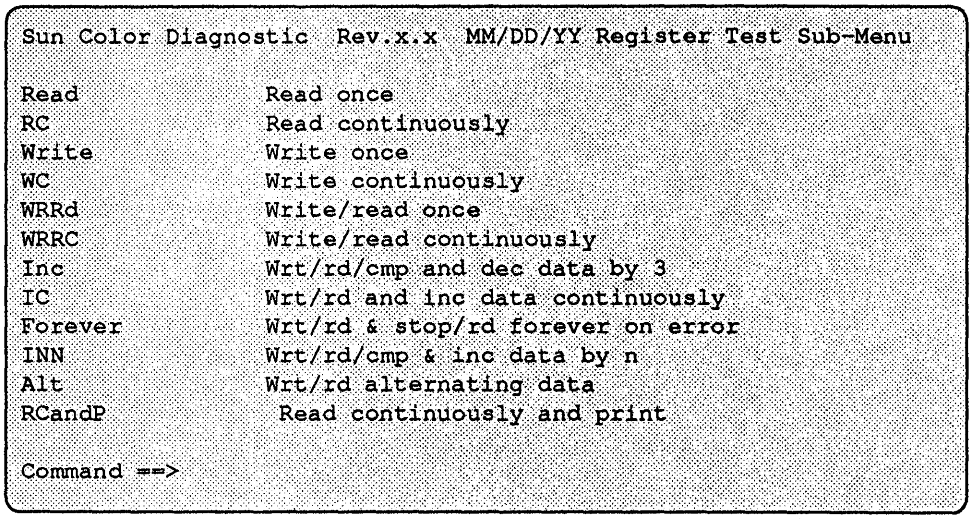
IV2

The *IV2reg* option executes the GP2 interrup vector register test and brings up a register sub-menu. The GP2 interrupt vector register is an 8-bit register. It holds the interrupt vector for the GP2 interrupt. This menu selection is valid only for the CG5 board.

Register Test Sub-Menu

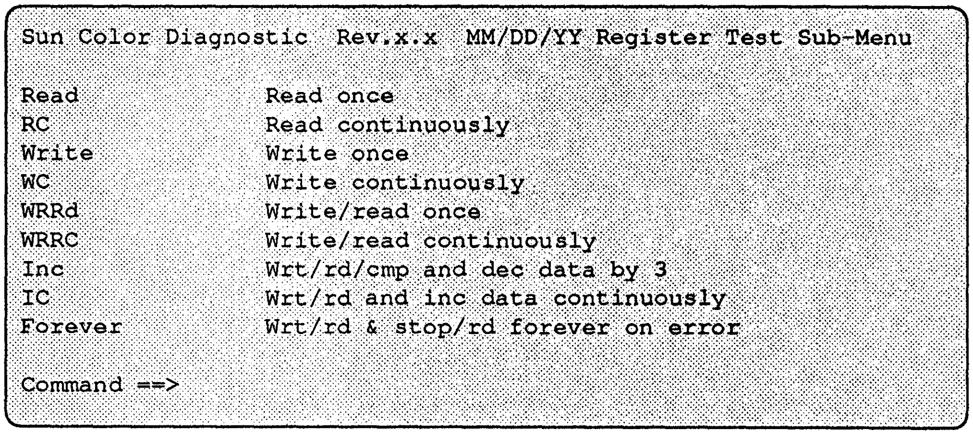
After selecting the register you want to test, the following sub-menus give you various ways to test it. Each previously described register selection produces one of the two following Register Test Sub-Menus. The sub-menu for 8-bit registers will only appear for the board ID and the GP2 interrupt registers on the Color-5 board.

For 16-Bit Registers



```
Sun Color Diagnostic  Rev.x.x  MM/DD/YY 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
RCandP               Read continuously and print

Command ==>
```

For 8-Bit Registers — CG5 Board



```
Sun Color Diagnostic  Rev.x.x  MM/DD/YY 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

Command ==>
```

The following paragraphs describe the Register Test Sub-Menu selections.

R

The *Read once* option reads the indicated register once, and prints its hexadecimal contents on the terminal screen. This command reads indicated register repeatedly. It prints the following message at the beginning of the cycle:

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

The value named 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. Every register on the Color-2 board can be read. The DMA, base and width registers on the Color-3 and Color-5 board are write-only.

RC

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

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

W

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

```
Enter Datum(hex)
```

Every register on the Color-2 board can be written. The Frame Count register on the Color-3 and Color-5 board is read-only.

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, as described above for the Write Once option. It then writes the indicated register repeatedly. At the end of 0x10000 writes, the test prints the following message:

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

If q has been entered, 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, as described above for the Write Once option. It then writes the indicated register once, using the value of the argument given, and immediately reads the value of the register back and prints it on the screen. The message printed is

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

Every register on the color-2 board can be read and written. The DMA, Base and Width registers on the color-3 board are write-only. The Frame Count register on the color-3 board is read-only.

WRRC

The *Write, then Read continuously* command prompts for a value to write, as previously described for the Write Once option. 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.
```

The value 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. Every register on the color-2 board can be read and written. The DMA, Base and Width registers on the color-3 board are write-only. The Frame Count register on the color-3 board is read-only.

I

The *Inc* option writes then reads the register continuously while changing the value of the data written. This *Write, Read, Compare, and Decrement* option 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 rdl,rd2,rd3
```

When the value decrements below zero, the test ends.

IC

The *IC* option writes then reads the register continuously while changing the value of the data written, and also halts on error. This *Write, Read, and Increment* option 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.

F

The *Forever* option writes then reads the register continuously while changing the value of the data written, and does not halt on error. This option is known as the *Write, Read, and Read on Error* option and 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.

NOTE *The next two options do not appear on the sub-menu for testing the CG5 8-bit registers.*

INN

The *INN* option writes then reads the register continuously while incrementing the value of the data written. This command is known as the *Write, Read, Compare and Increment by n* command and 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 *Alt* option writes then reads the register continuously while alternating the value of the data written. This command, known as 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.
```

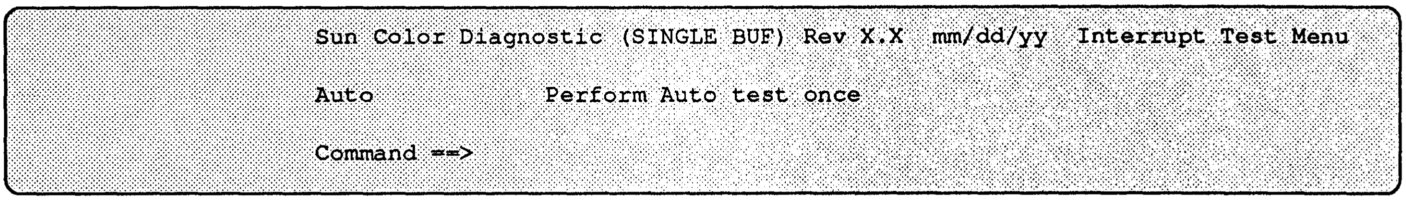
It then reads the keyboard to see if the letter q has been typed. If it has, the test quits. If it hasn't, the test repeats the cycle.

RCP

The *Read Continuously and Print* option reads continuously and then prints.

Interrupt Test Menu

Type *i* from the Manual Test menu to select the interrupt test menu. The `SINGLE BUF` flag will only be present on the Color-3 and Color-5 test menus. Only one option is available as shown below:



```
Sun Color Diagnostic (SINGLE BUF) Rev X.X  mm/dd/yy  Interrupt Test Menu

Auto            Perform Auto test once

Command ==>
```

`Auto`

The *Auto* option performs the interrupt test once, then returns to the main menu. The interrupt test causes an interrupt and waits for several vertical retraces before confirming that the interrupt bit in the status register toggles.

Color Map Test Menu

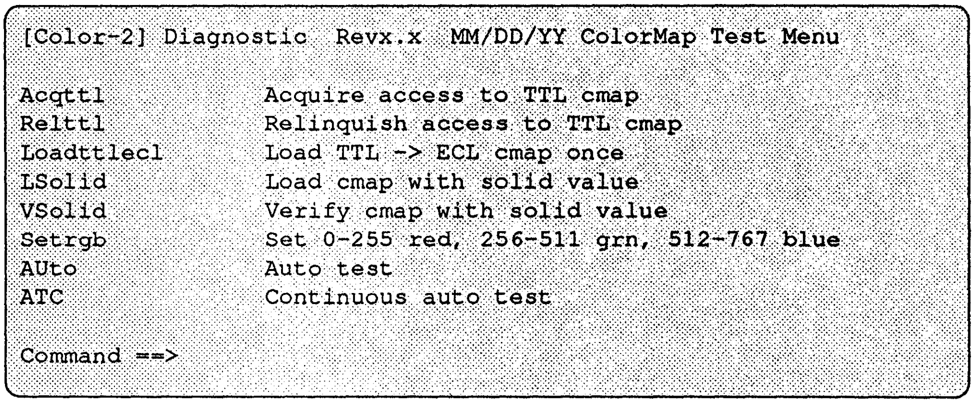
Selecting **`COlor`** from the Manual Menu displays the Color Map Test Menu options. This test checks the Sun-3 color map by loading it with different values and patterns, then verifying that the values in the map are correct. Select the color map values by choosing them yourself, or have the values selected automatically. These tests can be set to run one time or continuously.

To display the image in the 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 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

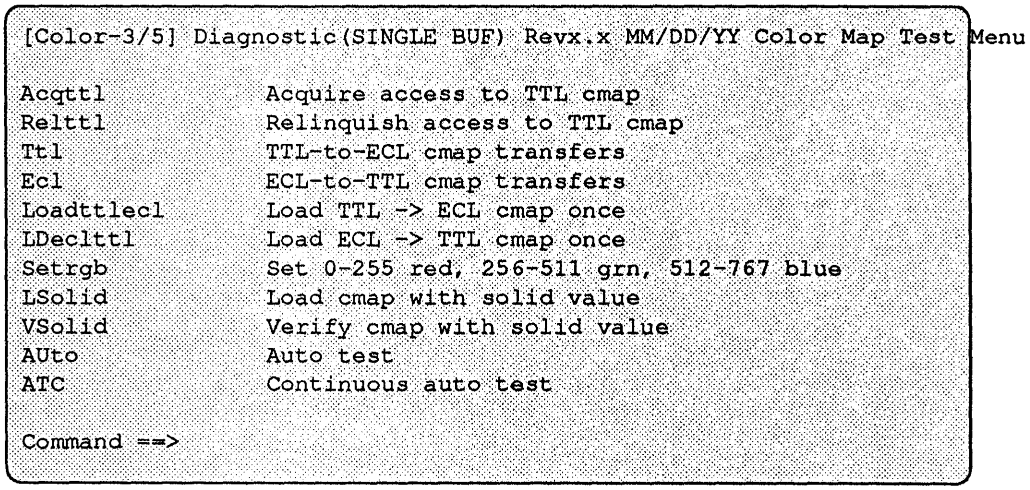
```
Load TTL -> ECL cmap
```

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



```
[Color-2] Diagnostic  Revx.x  MM/DD/YY ColorMap Test Menu

Acqttl           Acquire access to TTL cmap
Relttl           Relinquish access to TTL cmap
Loadttlecl       Load TTL -> ECL cmap once
LSolid           Load cmap with solid value
VSolid           Verify cmap with solid value
Setrgb           Set 0-255 red, 256-511 grn, 512-767 blue
AUto             Auto test
ATC              Continuous auto test

Command ==>
```

- or -



```
[Color-3/5] Diagnostic(SINGLE BUF) Revx.x MM/DD/YY 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
LDeclttl         Load ECL -> TTL cmap once
Setrgb           Set 0-255 red, 256-511 grn, 512-767 blue
LSolid           Load cmap with solid value
VSolid           Verify cmap with solid value
AUto             Auto test
ATC              Continuous auto test

Command ==>
```

All the options in the Color-2 Color Map Test Menu are also available in the Color-3 and Color-5 Menus. The Color-3/5 Color Map Test Menus have three additional options that are not available for Color-2 board testing. The following text describes each option on the Color-3 and Color-5 menus, in the order shown in the menu example above.

Acqttl

The *TTL Acquire* option acquires access to the TTL color map. The color map needs to be accessed before it can be loaded or set and needs to be released before the changes can be seen on the screen.

R

The *Relttl* option releases access to the TTL color map. The color map needs to be accessed before it can be loaded or set and needs to be released before the changes can be seen on the screen.

T

The *Ttl* option enables data transfer from the TTL to the ECL color map. This selection is only valid for Color-3 and Color-5 boards.

E

The *Ecl* option enables data transfer from the ECL to the TTL color map. This selection is only valid for Color-3 and Color-5 boards.

L

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

LD

The *LDeclttl* option transfers the data from the ECL color map to the TTL color map once. This selection is only valid for Color-3 and Color-5 boards.

S

The *Setrgb* option sets the color map with a default color sequence. This process loads the color map with three linear ramps of Red, Green, and Blue. The 256 entries will appear on the screen as a 16x16 grid of squares. The first 85 squares will be red, increasing in intensity from left to right, from top to bottom. The next will be green and last will be blue, also increasing in intensity.

LS

The *LSolid* option loads the color map with a solid value. You are prompted for the color map entry, which can be 0 to 255, and for the color intensities. Each of the 256 entries in the color map can be individually set with a color. These entries can be verified with the `VS` command, described next.

VS

The *VSolid* option verifies the color map. You are prompted for the color map entry, which can be 0 to 255, and for the color intensities. Each of the 256 entries in the color map can be individually set with a color.

AUto

The *auto test* option performs the auto color map test once.

ATC

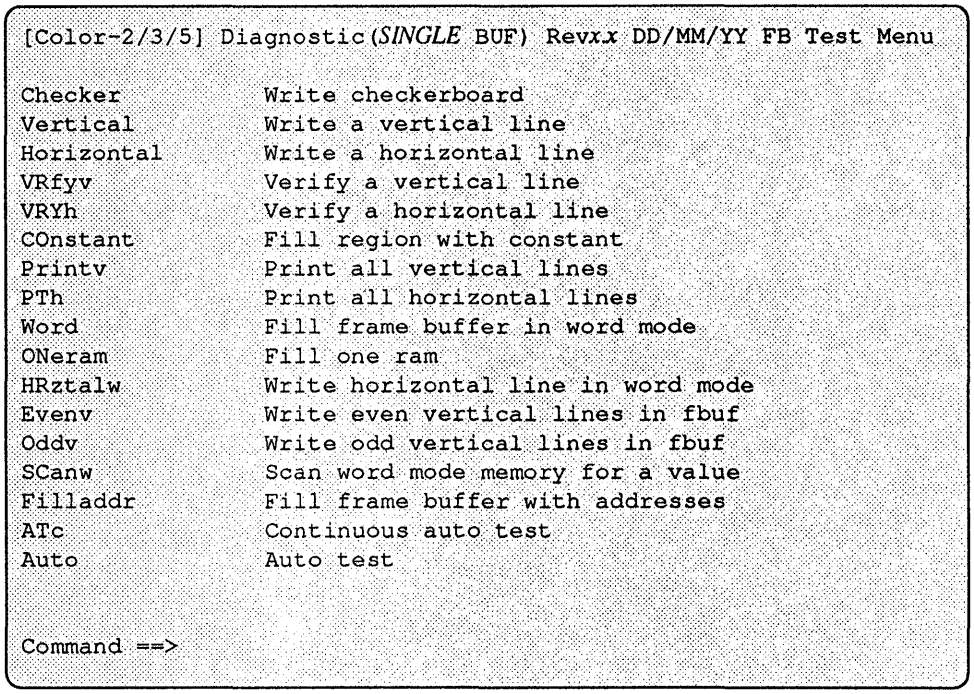
The *continuous auto test* command performs the auto color map test until the letter `q` is typed on the keyboard.

Frame Buffer Test Menu

Selecting **Buffer** from the Manual menu displays the Frame Buffer Test Menu options. The menu is the same for Color-2, Color-3 and Color-5 boards.

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 continuously.



```
[Color-2/3/5] Diagnostic(SINGLE BUF) Revxx DD/MM/YY 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
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
ATc              Continuous auto test
Auto             Auto test


Command ==>
```

The following paragraphs describe the Frame Buffer Test Menu selections.

C

The *Checker* option writes a checkerboard pattern to the screen. The checkerboard is a 16x16 matrix showing the 256 colors in the color map. The boxes can be thought of as being numbered, starting with zero, increasing from left to right, then top to bottom.

V

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

H

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

VR

The *VRfyv* option 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*

VRY

The *VRY* command verifies that the line displayed by the `horizontal line` command was drawn correctly. The test prompts you to enter 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*

CO

The *COnstant* option draws a rectangular region of constant color on the screen. You are asked for the x and y coordinates of the rectangle's upper left corner, along with its height (dy), width (dx), and color.

P

The *Printv* option fills the screen with every possible vertical line, each in a different color.

PT

The *PTH* option fills the screen with every possible horizontal line, each in a different color.

W

The *Word* option fills the frame buffer in word mode. The *word mode* command prompts for a 32 bit value. It then writes this value into every location in the frame buffer.

ON

The *ONe* option calls the fill one RAM test and menu, described on the following page.

HR

The *HRztalw* option writes a horizontal line in word mode. 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.

E

The *Evenv* option writes every other vertical line on the screen (even lines) with the color you are prompted to enter.

O

The *Oddv* option writes every other vertical line on the screen (odd lines) with the color that you are prompted to enter.

SC

The *SCan* option scans word mode memory for the value you have been prompted to enter. It then scans the buffer, in word mode, looking for that value. Every time it encounters the value, it prints:

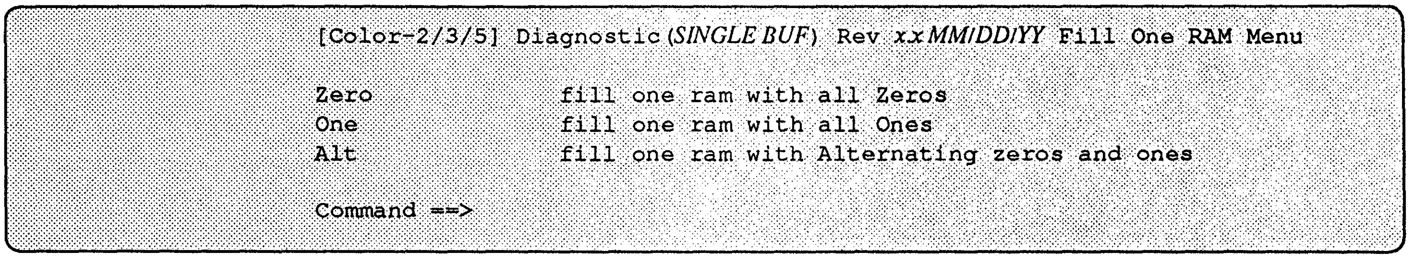
```
Data matches value at address location
```

F

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

ATc

The *ATc* option executes the continuous auto test. 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.



```
[Color-2/3/5] Diagnostic (SINGLE BUF) Rev xx MM/DD/YY Fill One RAM Menu

Zero            fill one ram with all Zeros
One             fill one ram with all Ones
Alt             fill one ram with Alternating zeros and ones

Command ==>
```

Z

The **Zero** option fills one ram will all zeros.

O

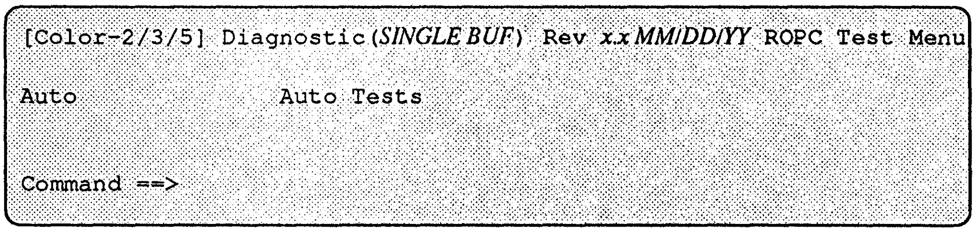
The **One** option fills one ram will all ones.

A

The **Alt** option fills one ram will alternating zeros and ones.

ROPC Test Menu

Type **r** from the Manual Test Menu to bring up the ROPC test menu, which is the same for Color-2 and Color-3 boards.



```
[Color-2/3/5] Diagnostic (SINGLE BUF) Rev xx MM/DD/YY ROPC Test Menu

Auto            Auto Tests


Command ==>
```

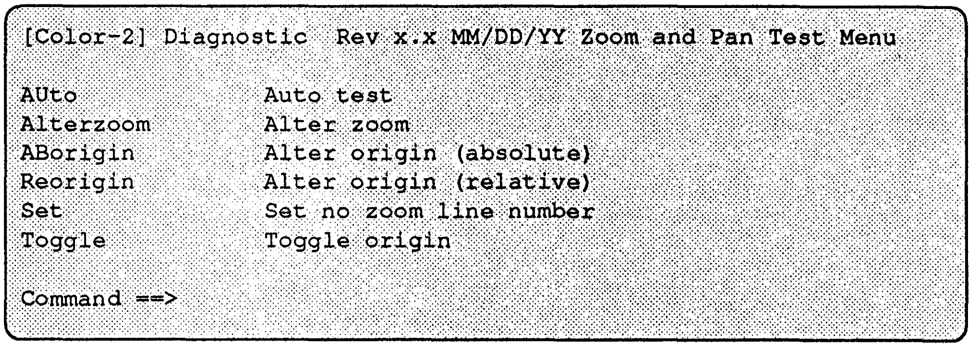
A

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

NOTE *During testing, colored horizontal lines may appear on the top of the display. This is a normal effect of the ROPC test.*

Zoom and Pan Test Menu

Selecting **Zoom** from the Manual menu displays the Zoom and Pan Test Menu options. The menu is only available for Color-2 boards.



```
[Color-2] Diagnostic  Rev x.x MM/DD/YY Zoom and Pan Test Menu

AUto                Auto test
Alterzoom           Alter zoom
ABorigin            Alter origin (absolute)
Reorigin            Alter origin (relative)
Set                 Set no zoom line number
Toggle              Toggle origin

Command ==>
```

AU

The *AUto* option executes the zoom and pan auto test. This tests the zoom and pan circuits for the color-2 board, and pans through zoom levels 0 to 7. The panning sequence is: right-left-up-down-diagonal.

A

The *Alterzoom* option changes the zoom of the screen. You are prompted for the zoom level (0 to 8).

AB

The *ABorigin* option changes absolute origin of the screen, which is normally in the upper left-hand corner. You are prompted for the new origin.

R

The *Reorigin* option changes relative origin of the screen, which is normally in the upper left-hand corner. You are prompted for the new origin.

S

The *Set* option sets a no-zoom horizontal line on the screen. Anything below the line is not zoomed. You are prompted for the new no-zoom line.

T

The *Toggle* option makes the screen toggle between two origins. You are prompted for the two origins.

DAC Test Menu

Type **da** from the Manual Test Menu 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.

```
[Color-2/3] Diagnostic(SINGLE BUF) Rev xx MM/DD/YY 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 ==>
```

The following paragraphs describe the DAC Test Menu choices.

R

The *Rhramp* option displays a horizontal Red ramp. This test checks the linearity of the red DAC.

G

The *Ghramp* option displays a horizontal green ramp. This test checks the linearity of the green DAC.

B

The *Bhramp* option displays a horizontal Blue ramp. This test checks the linearity of the blue DAC.

W

The *Whramp* option displays a horizontal White ramp. This test checks the linearity of all three DACs working together.

RV

The *RVramp* option displays a vertical Red ramp.

GV

The *GVramp* option displays a vertical Green ramp.

BV

The *BVramp* displays a vertical Blue ramp.

WV

The *WVramp* option displays a vertical White ramp.

RGBw

The *RGBw* option sequentially displays horizontal Red, Green, Blue, and

White ramp images.

BOR
The *BORder* option displays a White border around the screen. This checks the beam deflection circuitry.

A
The *Alt* option displays one color, then slowly overlays another color.

AU
AUto option executes the continuous DAC auto test, and runs through the tests listed above until you enter `q` is on the keyboard.

S
The *Stab* option executes the screen stability test, and displays a gray pattern that produces consistent shading and color on the entire screen.

11.8. Auto Test

Selecting **`Auto`** from the Manual menu (shown at the beginning of this chapter) executes the Auto test, which is similar to the Auto test from the Main Menu. Following are descriptions of these tests.

DAC Tests

This test is only executed once and is useful for adjusting the color monitor. These tests REQUIRE visual verification by the user. Therfore, NO error messages are ever printed during the DAC tests. The following tests are executed in the order of presentation:

Single Horizontal Ramp
The frame buffer is loaded with a single horizontal ramp as defined in the Single Horizontal Ramp Test Values table.

Table 11-1 *Single Horizontal Ramp Test Values*

Column	*pixel value*
0	31
1	30
:	:
:	:
62	1
63	0
64	1
65	2
:	:
:	:
572	254
573	255
574	254
575	253
:	:
:	:
1081	1
1082	0
1083	1
1084	2
:	:
:	:
1150	34
1151	35

Red Ramp Monotonicity

The red intensity is greatest at the center and least at the edges. The frame buffer still has the values from the previous test. The RED color map is loaded with the ramp function in the table. The GREEN and BLUE color map are set to zero (off). (see "Red Ramp Monotonicity" table)

Table 11-2 *Red Ramp Monotonicity*

Red Color Map Value	0, 1, . . ., 254, 255 0,1, . . ., 254, 255

Green Ramp Monotonicity

The green intensity is greatest at the center and least at the edges. The frame buffer still has the values from the previous test. The RED color map is loaded with zero (off), the GREEN color map has the values from the above table, and BLUE color map is set to zero (off).

Blue Ramp Monotonicity

The blue intensity is greatest at the center and least at the edges. The frame buffer still has the values from the previous test. The RED color map is loaded with zero (off), the GREEN color map is set to off, and BLUE color map has the values from the Red Ramp table.

White Ramp Monotonicity

The white intensity is greatest at the center and least at the edges. The frame buffer still has the values from the previous test. The RED, GREEN, and BLUE color map has the values from the Red Ramp table.

Stable Gray Pattern

This test checks DAC output in a stable gray pattern. The color map is loaded with the following functions. The result is that the first color is black, the last color is white, and all other colors are red. Every 32-bit word address (in word mode) in the frame buffer is loaded with 0x55555555 (refer to the Stable Gray Pattern Values table.)

Table 11-3 *Stable Gray Pattern Values*

Color Map	*Values*
Red	0,0xFF,0xFF, . . .,0xFF,0xFF
Green	0,0, . . ., 0, 0xFF
Blue	0,0, . . ., 0, 0xFF

Draw Borders

This test verifies that all screen borders are visible. This setup of color map table and frame buffer draws a white line along each border of the screen, which had turned black before the drawing. The color map is loaded with the monochrome ramp function (refer to the Draw Borders Test Color Map Values table.)

Table 11-4 *Draw Borders Test Color Map Values*

Color Map	*Values*
Red	0,1,, . . ., 254, 255
Green	0,1, . . ., 254, 255
Blue	0,1, . . ., 254, 255

The frame buffer (FB) is cleared except the following lines are set on (white). (x1, y1) are the starting points of the line, and (x2, y2) are the ending points. (see "Draw Borders Test FB Values" table)

Table 11-5 *Draw Borders Test FB Values*

Line	*x1*	*y1*	*x2*	*y2*
Top Line	0	0	1150	0
Right Side	1150	0	1150	899
Bottom Line	1150	899	0	899
Left Side	0	899	0	0

11.9. Interrupt Tests

The interrupt circuitry check is done with the steps shown in the table below. A dash (-) in place of an error code indicates NO error checking. "Wait" under iterations means that the diagnostic will wait until the test condition occurs (refer to the Interrupt Test Sequence table.)

Table 11-6 *Interrupt Test Sequence*

Sequence	*Description*	*Iterations*	*Error Code*
1	Set up interrupt handler, Int Reg = 0x0054	1	-
2	Check if retrace bit in status register toggles	2000000	6
3	If Color-3 or 5, Check if wait bit in db reg set/cleared	1	33
4	Wait for end of retrace	wait	-
5	Enable interrupts	1	-
6	Wait for retrace to begin	wait	-
7	Wait for end of retrace	wait	-
8	Check if interrupt occurred	1	6
9	Turn off all interrupts	1	-

NOTE *The remaining tests loop the number of times set by the* PASS= *variable.*

11.10. Auto Register Test Details

This section only applies to the Register tests executed during the auto test from the *.Main*Menu

NOTE *The Color-5 OR the Color-3 OR the Color-2 test below will run only on the board with the same name.*

The registers shown below are tested by writing, reading and comparing with pre-determined values.

NOTE *The value written to the registers over the on-board bus will be held by the bus for a lengthy period. Because of this undesirable effect, on the Color-3 and Color-5 graphics boards only, the bus is precharged to a different level by writing to a dummy register, which is an unused location. The value written to the dummy register is 0x5555 before the write to the actual register, and 0xAAAA after writing to the registers.*

Color-5 Auto Register Test

This test is for the Color-5 graphics board only. The registers below are tested by writing, reading and comparing with pre-determined values. The following table documents the values written to each register, the decrement count step in each loop, the value the loop stops at, value loaded into register after the test, and the possible error code.

Table 11-7 *Color-5 Auto Register Test Values*

Register	*Initial Value*	*Increment*	*End Value*	*Interations*	*Value After Test*	*Error*
Per-Plane Mask	0x00FF	-3	0	84	0x00FF	1
Interrupt Vector	0x00FF	-3	0	84	0x0054	29
Status	0x003F	-3	0	20	0x0001	0
Double Buffer	0xF900	-3	0	21,248	0x0000	37
Retrace bit Status	(see below)	-	-	1	-	33
Frame Count	(see below)	-	-	1	-	34
Board ID	0x0001	-	-	1	-	41
ROPMODE	0x0038	-3	0	18	0x0000	42
2nd Status	(see below)	-	-	1	-	43
GP2 Interrupt	0x00FF	-3	0	84	0x0000	44

After these register are tested, the retrace bit in the status register is tested for toggling. The possible error code is 6. The wait bit in the double buffer register is tested as follows: set then cleared. The double buffer register is cleared to zero after test. The possible error code is 33.

The Frame Count register is tested ONLY if the other tests produced NO error. It checks that the frame count register increments. The possible error code is 34.

Bit 14 (GP2 interrupt enable/disable) of the 2nd Status register is first read. The bit is then toggled and the register is read again.

Color-3 Auto Register Test

This test is for the Color-3 graphics board only. The registers below are tested by writing, reading and comparing with pre-determined values. The following table documents the values written to each register, the decrement count step in each loop, the value the loop stops at, value loaded into register after the test, and the possible error code.

After these register are tested, the retrace bit in the status register is tested for toggling. The possible error code is 6. The wait bit in the double buffer register is set, then cleared. The double buffer register is cleared to zero after test. The possible error code is 33.

Table 11-8 *Color-3 Auto Register Test Values*

Register	*Initial Value*	*Increment*	*End Value*	*Interations*	*Value After Test*	*Error*
Per-Plane Mask	0x00FF	-3	0	84	0x00FF	1
Interrupt Vector	0x00FF	-3	0	84	0x0054	29
Status	0x003F	-3	0	20	0x0001	0
Double Buffer	0xF900	-3	0	21,248	0x0000	37
Retrace bit Status	(see below)	-	-	1	-	33
Frame Count	(see below)	-	-	1	-	34

The Frame Count register is tested ONLY if the other tests produced NO error. It checks that the frame count register increments. The possible error code is 34.

Color-2 Auto Register Test

This test is for the Color-2 graphics board only. The registers below are tested by writing, reading and comparing with pre-determined values. The following table documents the values written to each register, the decrement count in each loop, the value the loop stops at, value loaded into register after the test, and the possible error code.

Table 11-9 *Color-2 Auto Register Test Values*

Register	*Initial Value*	*Increment*	*End Value*	*Interations*	*Value After Test*	*Error*
Status	0x003F	-3	0	20	0x0001	0
Per-Plane Mask	0x00FF	-3	0	84	0x00FF	1
Word Pan	0xFFFF	-3	0	21,845	0x0000	2
Zoom and Offset	0x00FF	-3	0	84	0x0000	3
Pixel Pan	0x00FF	-3	0	84	0x0000	4
Variable Zoom	0x00FF	-3	0	84	0x00FF	5
Interrupt Vector	0x00FF	-3	0	84	0x0054	29

11.11. Auto Color Map Test

The color map tables consist of a high-speed ECL look-up table used during video display, and a TTL shadow color look-up table that can be accessed at any time by the host software. The test algorithms for the two look-up tables are the same.

First a 16x16 grid checker-board pattern is written to the frame buffer. The checker-board pattern is explained as follows:

```
for (initial=0, outer_loop=0; outer_loop < 16; initial+=16, outer_loop++)
for (loop=0; loop < 56; loop ++)
for (pattern=initial; pattern < 16+initial; pattern++)

consecutive 72 pixels are loaded with pattern;
```

If the screen size is 1152x900, the two top and two bottom rows of squares contain one more extra pixel-row than the other rows of squares. This is because the screen size does not divide evenly into the 16x16 grid. This removes the problem of having extra pixel rows left on the bottom of the screen.

The size of each square is 72x56 pixels. The size of the squares on the two top rows and two bottom rows is 72x57 pixels.

The auto color map test is performed three times. The test consists of a series of memory tests, followed by two iterations of color loading and verification.

The series of memory tests are address unique, modified galpat, and modified surround disturb tests, and are performed to test the Color Map Tables. For detailed information about these algorithms, see the description under Frame Buffer Tests.

The color loading and verification test consists of seven (7) color patterns loaded in the color map and automatically verified, followed by complementary color data loaded in the color map and verified. The color maps are loaded and verified automatically (refer to the Auto Color Map Test Values table.)

Table 11-10 *Auto Color map Test Values*

Type of Display	*Red*	*Green*	*Blue*
Solid	0xAA	0x55	0xCC
Solid	0x00	0xFF	0xC3
Solid	0xC3	0x28	0xB7
Ramp	0x00-0xFF	0x00	0x00
Ramp	0x00	0x00-0xFF	0x00
Ramp	0x00	0x00	0x00-0xFF
Ramp	0x00-0xFF	0x00-0xFF	0x00-0xFF

The following two sets of patterns loaded into color map consist of complementary data for the same color.

```
red [i] = 0x55, green [i] = 0xC3, blue [i] = 0x28
red [i] = 0xAA, green [i] = 0x3C, blue [i] = 0xD7
```

The test loads even locations of color map with the first set of data, and odd locations with the second set of data. It then writes every two locations with complementary data: locations zero and one with first set of data and locations two and three with second set of data, and so on. It then writes every 4, 8, 16, 32, 64, and 128 locations with complementary data. There are eight sets of writes, reads, and comparisons for the color map as described above.

11.12. Load Colors

After the Color Map test, the color map is reset by loading in "nice" colors. The 256 color map entries are divided into four sections of 64 entries each, and loaded with the following quad-ramp function increasing intensity colors: red, green, blue, white (refer to the Load Colors table.)

Table 11-11 *Load Colors*

Color Map Entry	*Color Map Entry (hex)*	*Red Value*	*Green Value*	*Blue Value*
0	0x00	0	0	0
1	0x01	4	0	0
2	0x02	8	0	0
3	0x03	12	0	0
:	:	:	:	:
61	0x3D	244	0	0
62	0x3E	248	0	0
63	0xeF	252	0	0
64	0x40	0	0	0
65	0x41	0	4	0
66	0x42	0	8	0
67	0x43	0	12	0
:	:	:	:	:
125	0x7D	0	244	0
126	0x7E	0	248	0
127	0x7F	0	252	0
128	0x80	0	0	0
129	0x81	0	0	4
130	0x82	0	0	8
131	0x83	0	0	12
:	:	:	:	:
125	0xBD	0	0	244
126	0xBE	0	0	248
191	0xBF	0	0	252
128	0xC0	0	0	0
129	0xC1	4	4	4
130	0xC2	8	8	8
131	0xC3	12	12	12
:	:	:	:	:
253	0xFD	244	244	244
254	0xFE	248	248	248
255	0xFF	252	252	252

11.13. Frame Buffer Tests: Word Mode and Pixel Mode

These tests check the one 1 MB frame buffer DRAM in both word and pixel modes. If double buffering (an optional extra 1 MB DRAM) exists, we apply the same algorithms that test the regular frame buffer. The frame buffer test does a series of tests in word mode, and then repeats the same set of tests in pixel mode. The possible error message code is 8 or 9.

Constant Pattern Test

This test fills the frame buffer with constant 32-bit word patterns, then reads them back for comparison. Each constant pattern is applied to the whole frame buffer in one pass. The sequence of constant patterns is described below, along with an overview of all the Frame Buffer tests (refer to the FB Constant Patterns table.)

The "Sequence Numbers" in the FB Contant Patterns table are printed to the screen during testing.

Table 11-12 *FB Constant Patterns*

Color-2			*Color-3/5*		
Sequence Number	*Test Name*	*Data*	*Sequence Number*	*Test Name*	*Data*
1	FB	0x00000000	0	FB	0x00000000
2	FB	0xFFFF0000	1	FB	0xFFFFFFFF
3	FB	0xFFFFFFFF	2	FB	0x55555555
4	FB	0xAA55CC33	3	FB	0xAAAAAAAA
5	FB	0x00FF3355	4	FB	0x66666666
6	FB	0xAA33CCEF	5	FB	0x99999999
			6	FB	0x18181818
			7	FB	0xE7E7E7E7
			8	FB	0x71717171
			9	FB	0x8E8E8E8E
			a	FB	0xC3C3C3C3
			b	FB	0x3C3C3C3C
7	Unique		c	Unique	
8	Random		d	Random	
			e	galpat	
			f	disturb	
			10	refresh	
9	C-word		11	C-word	
A	C-byte		12	C-byte	

Address Uniqueness Test

This test fills each 32 bit word with its address, then reads and compares data read with the data written. This is test seven for Color-2 and test "C" for Color-3 and Color-5.

Random data test

This test checks the frame buffer with the random data test. The following random number generator is used in this test. The first random number is set to 433. Each random number after this is generated with: r = r + 4*r + 17623. For more information on this random number generator, refer to Knuth, "The Art Of Computer Programming", VOL 2, Chapter 3.

NOTE *Color-3 and Color-5 boards run the modified galpat, surround disturb, refresh logic tests, and two checker tests. Color-2 boards run the two checker tests.*

Address Test (Modified Galpat)

This is test "E", for one Mbyte memory. Because the 32-bit-word address is used here, we treat this memory as a 256K, 32-bit- word DRAM. This memory is divided into three groups (refer to the Address Test (galpat) table and the Galpat Test Algorithm table.)

Table 11-13 *Address Test (galpat)*

Sequence	*Description*
First	32-bit word address 0, 3, 6, 9, ...etc, and the last word address of the frame buffer.
Second	32-bit word address 1, 4, 7, 10, ...etc.
Third	32-bit word address 2, 5, 8, 11, ...etc.

Table 11-14 *Galpat test Algorithm*

Sequence	*Description*
1	Clear all DRAM by writing zero's to all frame buffer
2	Fill first group with 0xFFFFFFFF, then read all frame buffer back for comparison
3	Fill second group with 0xFFFFFFFF, then read all frame buffer back for comparison.
4	Fill third group with 0xFFFFFFFF, then read all frame buffer back for comparison. Now every bit of frame buffer should be turned on.
5	Fill first group with 0x00000000, then read all frame buffer back for comparison.
6	Fill second group with 0x00000000, then read all frame buffer back for comparison.
7	Fill third group with 0x00000000, then read all frame buffer back for comparison. Now every bit of frame buffer should be turned off.

Address test (Modified Surround Disturb)

This test writes 0xFF to 12 critical locations of pre-cleared frame buffer (refer to the Surround Disturb Test Addresses and Surround Disturb Algorithm tables)

Table 11-15 *Surround Disturb Test Addresses*

Number	*Memory Address*	*Data Written*	*Iterations*
1	0x00000	0xFF	256
2	0xFFFF0	0xFF	256
3	0x55550	0xFF	256
4	0xAAAA0	0xFF	256
5	0x66660	0xFF	256
6	0x99990	0xFF	256
7	0x18180	0xFF	256
8	0xE7E70	0xFF	256
9	0x71710	0xFF	256
10	0x8E8E0	0xFF	256
11	0xC3C30	0xFF	256
12	0x3C3C0	0xFF	256

Table 11-16 *Surround Disturb Algorithm*

Sequence	*Description*
1	Clear all frame buffer.
2	Keep writing 0xFF to one of the above locations for 256 times.
3	Read whole frame buffer back for comparison.
4	Go to step 1. The sequence of locations to be written with 0xFF is exactly the same as the above table of address.

Because this test writes all addresses with zeroes, with the exception of one of the above locations, the entire screen is turned dark except for one pixel—the byte address written with 0xFF— which is turned bright. While this test is running, you can see that one bright spot moves to 12 different locations on a dark screen.

Refresh Logic Test

This test is to check the frame buffer DRAM refresh logic (refer to the Refresh Logic Algorithm table.)

Table 11-17 *Refresh Logic Algorithm*

Sequence	*Description*
1	Clear all frame buffer.
2	Write even 32-bit word addresses with 0x0 and odd 32-bit addresses with 0xFFFFFFFF.
3	Wait for a while.
4	Read all frame buffer back for comparison.

Checker Test (32-bit Word Patterns)

This test writes the entire frame buffer with checkerboard patterns — one pattern and its complement — then reads them back for comparison. The first pass is to write every even word address with one pattern, and every odd word address with its complement. The second pass applies the same pattern to every even two-word address, word address 0, 1, 4, 5, etc, and its complement to every odd two-word address. Each segment, with the same pattern in the size of two-to-the-nth memory space, doubles its size after each pass until the size reaches one-half Mbyte. There are 18 passes for a complete test. Note that we use the same pattern for each pass in one test (refer to the Checker Test (word) Patterns table.)

Table 11-18 *Checker Test (word) Patterns*

Sequence	*Pattern Written*
1	0xA53C5AC3
2	0x43215678
3	0xF0FF00AA
4	0x87345520
5	0x24689753
6	0x34F76206

Checker Test (Byte Patterns)

This is test "A" for Color-2, and test "11" for Color-3 and Color-5. This test is the same as the previous one except that byte patterns to byte locations are applied here (refer to the Checker Test (byte) Patterns table.)

Table 11-19 *Checker Test (byte) Patterns*

Sequence	*Pattern Written*
1	0xF0
2	0x81

NOTE *If any errors occur in these tests (0 through A for Color-2 and 0 through f for Color-3 and Color-5) a cumulative error message is displayed. If no errors occurred,* NoErrors *is displayed. All DMA tests are executed for double-buffered Color-3 boards ONLY.*

DMA Window Test

The double buffering memory option includes circuitry that maps read and write accesses in the offset range 0x320000 to approximately 0x3FFFFE to an address defined by a pixel-mode address counter. If the double buffering is on the board, this test will be executed automatically. The DMA base register test is shown. The error code for this test is 35 (refer to the DMA Window Test table.)

Table 11-20 *DMA Window Test*

Sequence	*Description*
1	Write 0xFF to DMA width register.
2	Through DMA base register, Write 32-bit address uniqueness data to every 16 pixel in frame buffer.
3	Read 32-bit data from every 16 pixel of frame buffer in pixel mode for comparison.
4	Through DMA base register, read 32-bit data from DMA address space for comparison.

DMA Width Register Test

The write-only DMA width register controls the width of the DMA window. In this test, the value in the DMA width register starts with 2, which means the actual window size is (2+1) * 16 in pixels. The first line in this window is filled in the frame buffer, starting from the upper left corner, through DMA space with the pattern equal to the value in DMA width register plus one.

Then, in the same address mode, the same patter is written to the next four locations to frame buffer. If the hardware is good, those last four writes should be in the first four positions in the second line of the window defined above, rather than showing up in the following four positions after the window boundary in the same line. The data is then read back for comparison. After this, the value in DMA width register is incremented by one. This procedure continues until the actual window width is equal to or greater than the width of the screen. The error code for this test is 36.

DMA Window Visual Demo

First, the color map is initialized as the one shown just before the auto frame buffer test. 35 is written to the DMA width register, which will make the actual window width

$$(35+1)*16=1152/2$$

half the screen width, and the value 16146 is written to the DMA base register, which makes the window appear approximately screen center. There are 448 lines in one window. Each pixel in this window is written with the same constant pattern. After each pass, the pattern is incremented by one. The pattern range is from 0 to 255. No errors are reported during this test.

11.14. Second DAC Auto Test

The DAC Test is executed again after the DMA Tests.

11.15. Auto Zoom and Pan Test

The Zoom tests are run for Color-2 boards ONLY.

For each zoom factor, origin moves down and up, to lower-right and back, to right, to lower-left and back, and left to the original starting point. Then it tests that the lower "N" lines do not zoom when the rest does. This test uses a 16x16 square grid checkerboard, and verifies the results.

11.16. ROPC Tests

The following series of tests are repeated for each ROPC plane, 0 through 7, in word mode, and then all planes in parallel in pixel mode. The ROPC tests include: Dst, Src1, Src2, Pattern, Mask1, Mask2, Shift, Function, Width, Op_count and Flag registers (refer to the ROPC Tests table.)

Table 11-21 *ROPC Tests*

ROPC Register	*Initial Value*	*Increment Value*	*Final Value*	*Interations*	*Error Codes*
Dest	0xFFFF	-3	0	21,845	10-17,20
Src1	0xFFFF	-3	0	21,845	10-17,20
Src2	0xFFFF	-3	0	21,845	10-17,20
Pat	0xFFFF	-3	0	21,845	10-17,20
Msk1	0xFFFF	-3	0	21,845	10-17,20
Msk2	0xFFFF	-3	0	21,845	10-17,20
SftVal	0x010F	-3	0	89	10-17,20
Func	0xFFFF	-3	0	21,845	10-17,20
Width	0xFFFF	-3	0	21,845	10-17,20
OpCnt	0xFFFF	-3	0	21,845	10-17,20
Flag	0xFFFF	-3	0	21,845	10-17,20

11.17. Random Data Function Test

This test writes random patterns to the src1, src2, dst, pattern and function registers. All possible shift amounts are tried. The results of the operation are read back from the frame buffer, and compared with computed results.

11.18. Implicit word mode Tests

For each bit plane, Dst Mode 0,2,4,6 is tested. For each addressing mode, 800 patterns are written and the results are read back.

11.19. Pixel Mode Per-Plane Masking Test

This test checks the mask register for all bit combinations by the first pixel, in pixel mode. For each mask bit combination "M", the following two tests are performed (refer to the Pixel Mode Per-Plane Masking Test table.)

Table 11-22 *Pixel Mode Per-Plane Masking Test*

Sequence	*Description*
1	0xFF is written through mask = M and read back through mask = 0xFF
2	0xFF is written through mask = 0xFF and read back through mask = M.

11.20. Word Mode Per-Plane Masking Test

This test checks the plane mask register in word-mode.

11.21. Error messages

The following error messages are displayed when an error is detected during any of the tests. At the beginning of each error message, a test number is displayed to indicate which test failed. The following table summarizes the error code numbers and associated messages.

Table 11-23 *Color Diagnostic Error Code Table*

Number	*Test*	*Number*	*Test*
0	Status Register	23	Frame Buffer Word Memory Plane 2
1	Plane Mask Register	24	Frame Buffer Word Memory Plane 3
2	Word Pan Register	25	Frame Buffer Word Memory Plane 4
3	Line Offset and Zoom Register	26	Frame Buffer Word Memory Plane 5
4	Pixel Pan register	27	Frame Buffer Word Memory Plane 6
5	Variable Zoom Register	28	Frame Buffer Word Memory Plane 7
6	Interrupts	29	Interrupt Vector Register
7	Shadow Color Map	30	ECL Color Map
8	Frame Buffer Word-Mode	31	Frame Buffer Word-Memory(set B)
9	Frame Buffer Pixel-Mode	32	Frame Buffer Pixel-Memory(set B)
10	ROPC Plane 0	33	Double Buffer Wait Bit
11	ROPC Plane 1	34	Frame Count Register
12	ROPC Plane 2	35	DMA Base Loading
13	ROPC Plane 3	36	DMA Width Loading
14	ROPC Plane 4	37	Double Buffer Register
15	ROPC Plane 5	38	DMA Base Register
16	ROPC Plane 6	39	DMA Width Register
17	ROPC Plane 7	40	Byte Write to ROPC
18	Pix-Mode Plane Masking	41	Board ID Register
19	Word-mode Plane Masking	42	ROPMode Register
20	ROPC Pixel Memory	43	2nd Status Register
21	Frame Buffer Word Memory Plane 0	44	GP2 Interrupt Vector Register
22	Frame Buffer Word Memory Plane 1		

The location of the faulty hardware shown in an error message is relative. The address given 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.

Register Test Error Messages

If an error occurs during any register test, an the error message will look something like this:

```
#N - rrrrr Register - Addr 0x0000, Wr 0x0000, Rd 0x0000.
```

The paragraphs that follow explain what could appear in place of the italics in the example above.

N is the error code number from the error code table. This number could be 0,1,2,3,4,5,29,34,37,38, or 39.

rrrrr
is the register name, such as status register, zoom register, etc.

0x0000
is the address, data written, and data read.

In addition, other tests can give specific error messages about the state of the register as follows:

```
0 - Status Register - Retrace bit never toggles.

33 - Double Buffer Register - Wait bit won't set.

33 - Double Buffer Regstr - Wait bit doesn't clear.

34 - Frame Count Register - doesn't increment.
```

Interrupt Test Error Messages

Possible interrupt test error messages are as follows:

```
6 - Interrupt Test - Retrace bit in Status Register Never Toggles.

33 - Interrupt Test - Double Buffer Register, Wait bit won't set.

33 - Interrupt Test - Double Buffer Register, Wait bit doesn't clear properly.

6 - Interrupt Test - No interrupt when Expected.
```

Color Map Test Error Messages

Possible color map test error messages are as follows:

```
#N - rrrrr - color Entry 0x0000. Read 0x0000.  Compare w/ 0x0000. Xor 0x0000.
```

The paragraphs that follow explain what could appear in place of the italics in the example above.

N is the error code number from the error code table.

rrrrr
is the error code statement from the error code table.

color
is either RED, GREEN, or BLUE.

11.22. Glossary

DAC
Digital to Analog Converter

DMA
Direct Memory Access.

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

ROPC
Raster Operations Chip.

FB Frame Buffer

12

CG6 Graphics Accelerator Board Diagnostic

12

CG6 Graphics Accelerator Board Diagnostic

12.1. Introduction

CG6 is a low-cost graphics accelerator board with the following capabilities:

- floating point 3D wireframe imaging
- 2D flat shaded polygons
- anti-aliased text/vectors
- multiple resolution support

The board consists of these major components: transformation engine and cursor ASIC chip, frame buffer controller ASIC chip, DAC chip, and video RAM. CG6 does not require any firmware to support its operation. The board can be accessed as a dumb color frame buffer, or as a graphics accelerator.

The CG6 Diagnostic is divided into the following functional areas:

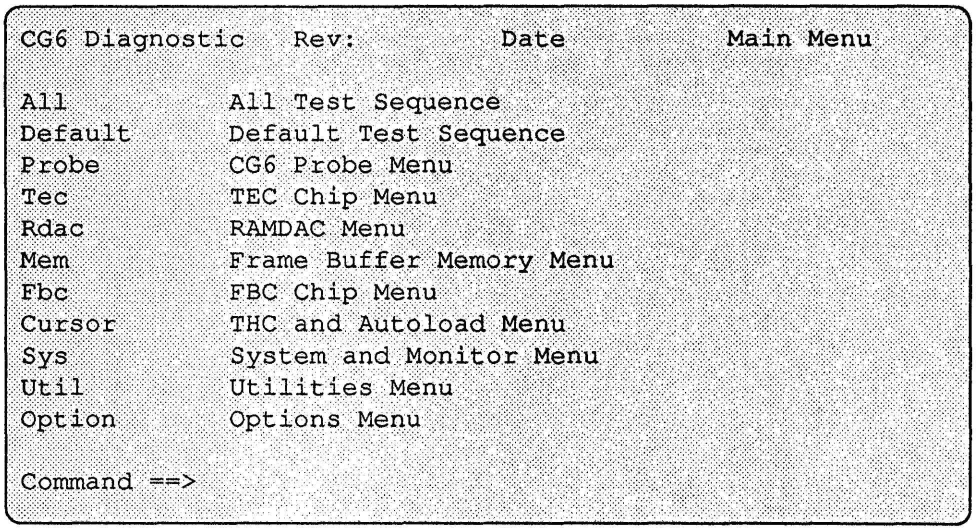
- P4 bus interface.
- TEC chip access and functional verification.
- FBC chip access and functional verification.
- DAC chip access and functional verification.
- Frame Buffer Memory.
- Hardware Cursor.
- System level and Monitor.

The CG6 diagnostic individually tests each functional section of the CG6 board. Tests are automated through menu selections. Each test has several parameters or options that can be entered from the command line. A help option is available for each test.

Before running this diagnostic, refer to *Chapter 2* for information on how to boot the SunDiagnostic Executive. You will select the CG6 test from the Diagnostics Menu.

12.2. The Main Menu

The Main Menu provides these choices. Enter the letter(s) shown in uppercase on the left to make a selection.



```
CG6 Diagnostic   Rev:          Date           Main Menu

All            All Test Sequence
Default        Default Test Sequence
Probe          CG6 Probe Menu
Tec            TEC Chip Menu
Rdac           RAMDAC Menu
Mem            Frame Buffer Memory Menu
Fbc            FBC Chip Menu
Cursor         THC and Autoload Menu
Sys            System and Monitor Menu
Util           Utilities Menu
Option         Options Menu

Command ==>
```

NOTE *When selecting tests, follow the menu sequence from the CG6 Probe Menu through the System and Monitor Menu.*

A *pass=*

Selecting **A** from the Main Menu executes the *All* test sequence. This option executes the *all* test option contained in each submenu. Each test is described in the section describing the corresponding menu.

The initial global counter is set to zero for the *All* test sequence. This global counter records the number of errors that occur for each test. At the conclusion of the *All* test sequence, the total number of errors is displayed. You may specify the number of iterations a test is executed, after the *pass=* parameter.

D

Selecting *Default* from the Main Menu executes a *quick confidence-level check* of the CG6 board to verify its functionality.

The remaining menu options present submenus. When an option is selected, a new menu is displayed. Type Esc followed by a Return to return to the main menu. These submenus (main menu options) are described in the following text:

Table 12-1 *CG6 Submenus*

Option	*Description*
Probe	CG6 Probe
Tec	TEC Chip Tests
Rdac	RAMDAC Tests
Mem	Frame Buffer Memory Tests
Fbc	FBC Chip Tests
Cursor	THC and Autoload Tests
Sys	System and Monitor Tests
Util	Utility Tests
Option	Options Menu

12.3. CG6 Probe Menu

Selecting **Probe** from the Main Menu displays this menu:

```
CG6 Diagnostic    Rev:      Date        CG6 Probe Menu

All           All Test Sequence
Default       Default Test Sequence
FHCRead       Hardware Config Read Probe
FHCWrite      Hardware Config Write Probe
THCRead       Hardware Cursor Read Probe
THCWrite      Hardware Cursor Write Probe
FBCRead       FB Controller Read Probe
FBCWrite      FB Controller Write Probe
TECRead       Xform Engine Read Probe
TECWrite      Xform Engine Write Probe
DFBRead       Frame Buffer Read Probe
DFBWrite      Frame Buffer Write Probe
ROMread       ROM Read Probe
?             Display the Help Menu

Command ==>
```

By default, all the tests in the Probe Menu are set to `pass=1` and `scope=0`, meaning that each test will execute once and there will be no scope loop, unless those parameters are set. Enter the number of test passes required after `pass=`. To enable a scope loop, enter the**scope=1**on

The tests in this menu check the interface between the CPU and the CG6. The CG6 addresses are multiplexed down to the FBC, TEC, DAC, DFB, and HWC chip selects. If an acknowledge is not returned by the CG6 upon request of a bus cycle, a CPU bus error is generated. How this bus error is processed depends on the kind of inCPUpresent The Probe Menu tests are briefly described in the text that follows.

All
All executes each option in the Probe Menu.

Default
Default also executes each option in the probe menu.

FHCR *pass= scope=[0,1]*
The *Hardware Configuration Read Probe* performs a longword read of the FHC register. This register contains the board ID in the high-order byte. The CG6 board ID is 0x60000000. If a 6 is not detected, the board is not accessible.

FHCW *pass= scope=[0,1]*
The *Hardware Configuration Write Probe* writes the value 0 to the FHC register. The written value is not checked for its validity. The test is interested only in the fact that a bus error does not occur.

THCR *pass= scope=[0,1]*
The *Hardware Cursor Read Probe* performs a longword read at address 0x900 of the THC.

THCW *pass= scope=[0,1]*
The *Hardware Cursor Write Probe* writes the value 0 to address 0x900 of the THC.

FBCR *pass= scope=[0,1]*
The *FB Controller Read Probe* performs a longword read at address 0x80 of the FBC.

FBCW *pass= scope=[0,1]*
The *FB Controller Write Probe* writes the value 0 to address 0x80 of the FBC.

TECR *pass= scope=[0,1]*
The *Transform Engine Read Probe* performs a longword read at address 0x100 of the TEC.

TECW
The Transform Engine Write Probe writes the value 0 to address 0x100 of the TEC.

DFBR *pass= scope=[0,1]*
The *Frame Buffer Read Probe* performs a longword read at address 0x0 of the dumb frame buffer memory.

DFBW *pass= scope=[0,1]*
The *Frame Buffer Write Probe* writes the value 0 to address 0x0 of the dumb frame buffer memory.

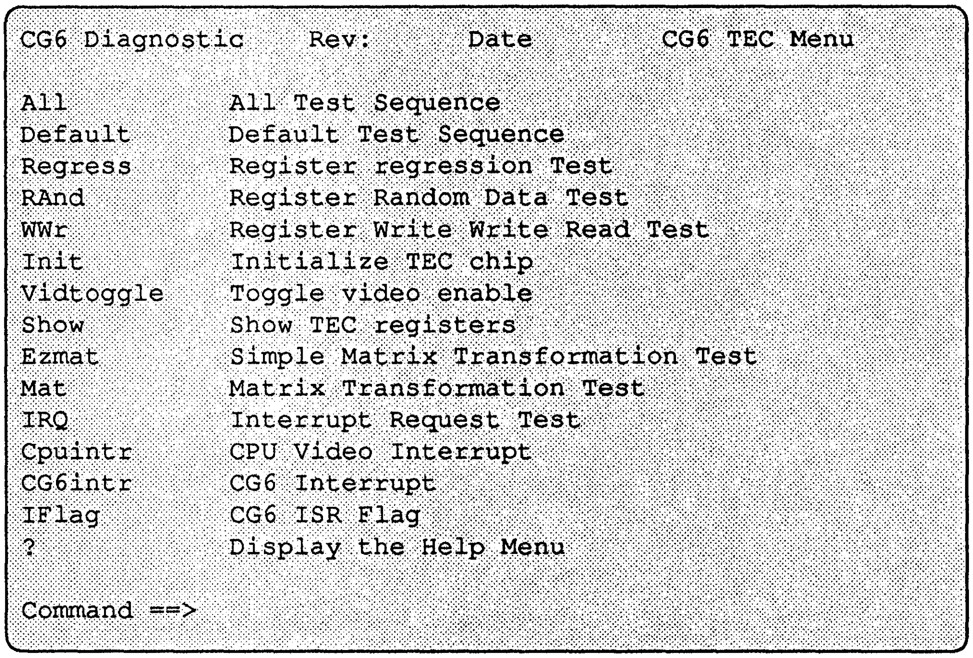
ROM *pass= scope=[0,1]*
The *ROM Read Probe* performs a longword read at address 0x0 in ROM on the CG6 board. The ROM is read-only memory.

?
Entering a question displays the Probe Help Menu. Command-line

parameters for each read or write test include selecting a maximum pass count and/or execution of a scope loop by setting scope=1. You must enter Esc to exit from the help menu.

12.4. TEC Menu

Selecting **TEC** from the Main Menu displays the TEC Menu.



```
CG6 Diagnostic     Rev:      Date         CG6 TEC Menu

All            All Test Sequence
Default        Default Test Sequence
Regress        Register regression Test
RAnd           Register Random Data Test
WWr            Register Write Write Read Test
Init           Initialize TEC chip
Vidtoggle      Toggle video enable
Show           Show TEC registers
Ezmat          Simple Matrix Transformation Test
Mat            Matrix Transformation Test
IRQ            Interrupt Request Test
Cpuintr        CPU Video Interrupt
CG6intr        CG6 Interrupt
IFlag          CG6 ISR Flag
?              Display the Help Menu

Command ==>
```

The tests in this menu exercise the path between the CPU and the TEC chip. Functional tests of the TEC chip include matrix transformation operations and video enable/disable. This menu also contains the initialization of video monitor syncs, hardware cursor control through the THC control section, and uploading of the FBC chip. Tests for the FBC chip functions are contained in the THC and Autoload submenu, and the FBC submenu, described later. The following section describes TEC menu choices.

A

All executes each test shown in the TEC menu.

D

Choosing *Default* executes a minimum set of tests to verify the functionality of the TEC.

R *pass=*

The *Register Regression Test* performs a walking 1's regression test of all the TEC read/write registers. By default, *pass=* is set to one.

RA *pass= seed=*

The *Register Ramdom Data Test* writes random data to all the read/write TEC registers. The registers are then read back for data comparison. A seed value may be specified (*seed=*) to generate a random pattern. By default, *pass=* is set to one, and *seed=0.*

WWr *off= data= pass=*

The *Register Write-Write-Read Test* writes the specified data (*data=*)to a location specified by the offset (*off=*) parameter while writing to a location in the TEC. Next, data is read from the specified offset location for verification. Default parameter values are:

```
off=0x19c
data=0x55555555
pass=1
```

I *res=[1 - 6]*

The *Initialize TEC chip* selection initializes the TEC and THC registers to the following known values:

```
1 = 1024 x 768
2 = 1024 x 1024
3 = 1152 x 900
4 = 1152 x 870
5 = 1600 x 1280
```

The *res=* parameter selects the register to be initialized.

V

The *Toggle Video Enable* test toggles the video bit in the TEC MISC register. Video is enabled when the bit is set to 1 and disabled if it is set to 0.

S

The *Show* TEC Menu option displays the contents of the TEC and THC registers listed in the following table.

Table 12-2 *TEC and THC Registers*

Address	Register Name	Value
0x000	TEC_MV_MATRIX	0x0
0x004	TEC_CLIPCHECK	0x0
0x008	TEC_VDC_MATRIX	0x0
0x800	THC_HCHS	0x00010009
0x804	THC_HCHSDVBS	0x00570000
0x808	THC_HCHD	0x0015005d
0x80C	THC_HCVS	0x00010005
0x810	THC_HCVD	0x002403a8
0x814	THC_HCREFRESH	0x0000016b
0x818	THC_HCMISC	0x0000148f
	THC_HCMISC	0x0000048f

E *pass=*

The *Simple Matrix Transformation Test* uses the TEC registers to multiply a 1 x 4 matrix to a 4 x 4 matrix. The result is compared with the precalculated value.

M *pass=*

The *Matrix Transformation Test* performs a rigorous check of the

TEC chip with different combinations of matrix transformations.

IRQ *num=*

The *Interrupt Request Test* performs a hardware test on the interrupt circuitry. The number of interrupts can be selected by setting *num=* to a decimal value.

The default is `num=100`.

C *C 0 or C 1*

The *CPU Video Interrupt* test checks the current value of the CPU interrupt bit when the parameters 0 or 1 are used.

C 0 clears the CPU interrupt bit.

C 1 enables the CPU to accept frame-buffer interrupt requests for vertical blanking.

CG6 *C 0 or C 1*

The *CG6 Interrupt* test checks the current value of the CG6 interrupt bit when the parameters 0 or 1 are used. This bit is located in the TEC miscellaneous register.

C 0 clears the TEC miscellaneous register interrupt bit.

C 1 enables it.

IF *C 0 or C 1*

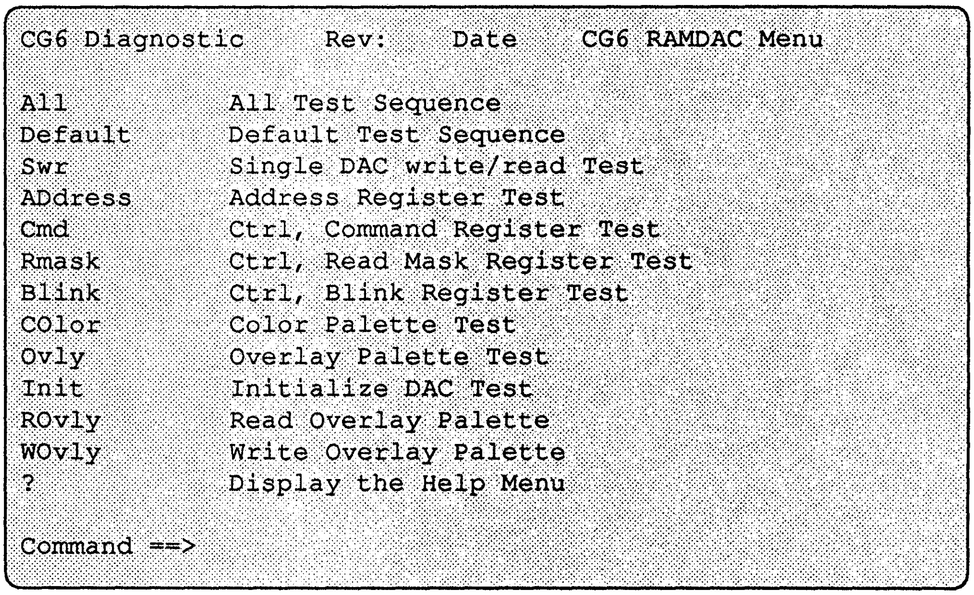
The *CG6 ISR Flag* test, chosen without a parameter of 0 or 1, displays the current value of the interrupt service routine flag. This flag is set when a frame-buffer interrupt is serviced by the CPU. **C 0** clears the interrupt service routine flag. **C 1** sets the interrupt service routine flag to 1.

?

Entering a question mark displays the TEC Help Menu. You must enter [Esc] to exit from the help menu.

12.5. RDAC Menu

Selecting **RDAC** from the Main Menu displays the RDAC Menu.



```
CG6 Diagnostic     Rev:    Date    CG6 RAMDAC Menu

All            All Test Sequence
Default        Default Test Sequence
Swr            Single DAC write/read Test
ADdress        Address Register Test
Cmd            Ctrl, Command Register Test
Rmask          Ctrl, Read Mask Register Test
Blink          Ctrl, Blink Register Test
COlor          Color Palette Test
Ovly           Overlay Palette Test
Init           Initialize DAC Test
ROvly          Read Overlay Palette
WOvly          Write Overlay Palette
?              Display the Help Menu

Command ==>
```

A

All executes each test on the RAMDAC menu.

D

The *Default* command executes a minimum set of tests to verify the functionality of RAMDAC.

S *pass= off= data=*

The *Single DAC write/read test* performs a single write/read/compare to the address register of the RAMDAC.

Default values are:

```
pass=1
off=0x0
data=0x55000000
```

AD *pass=*

The *Address Register Test* performs a walking-ones regression of write/read/compare to the address register of the RAMDAC.

The default number of passes is one.

C *pass=*

The *Control, Command Register Test* performs a walking-ones regression of write/read/compare to the RAMDAC command register.

The default number of passes is one.

Rmask *pass=*

The *Control, Read Mask Register Test* performs a walking-ones regression of write/read/compare to the RAMDAC read mask register.

The default number of passes is one.

`Blink` *pass=*

The *Control, Blink Register Test* performs a walking-ones regression of write/read/compare to the RAMDAC blink register.

The default number of passes is one.

`CO` *pass= test=[1,2,3]*

The *Color Pallette Test* performs one of three tests: *address, constant,* or *random pattern.* Each test checks the RAMDAC color pallet.

For example, entering

```
co test=1
```

selects the *address* test.

```
co test=2
```

selects the *constant* test.

```
co test=3
```

selects the *random* test.

`O` *pass=*

The *Read Overlay Pallette* test checks the DAC overlay registers that have been written-to and verified.

The default number of passes is one.

`I` This test *initializes* the DAC registers and loads the default color map. In addition, the control register is written-to for verification.

`RO` *pass=*

The *Read Overlay Pallette* test reads the content value of the overlay registers located in the DAC.

`WO` *pass=*

The *Write Overlay Pallette* test interactively writes the content value of the overlay registers located in DAC

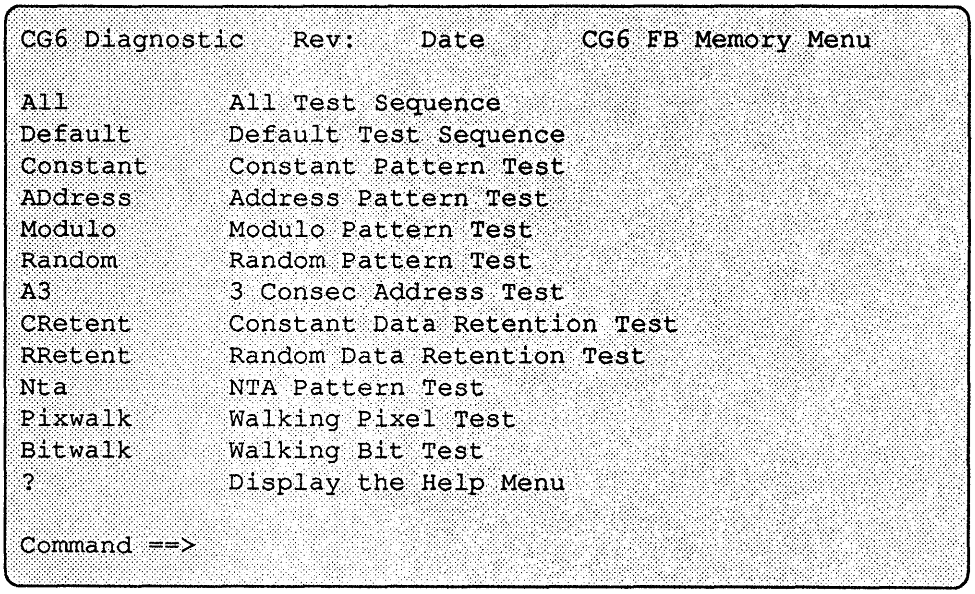
The default number of passes is one.

`?`

A question mark displays the CG6 RAMDAC Help Menu. You must enter [Esc] to exit from the help menu.

12.6. FB Memory Menu

Selecting **M** from the main menu brings up the Frame Buffer Memory Menu. Enter the letter(s) shown in uppercase on the left to execute the various test sequences.



```
CG6 Diagnostic   Rev:    Date      CG6 FB Memory Menu

All          All Test Sequence
Default      Default Test Sequence
Constant     Constant Pattern Test
ADdress      Address Pattern Test
Modulo       Modulo Pattern Test
Random       Random Pattern Test
A3           3 Consec Address Test
CRetent      Constant Data Retention Test
RRetent      Random Data Retention Test
Nta          NTA Pattern Test
Pixwalk      Walking Pixel Test
Bitwalk      Walking Bit Test
?            Display the Help Menu

Command ==>
```

Frame Buffer Memory Menu choice descriptions follow.

All executes each Frame Buffer Memory test.

D

Default executes a minimum set of tests to check the functionality of Frame Buffer memory.

C *off= xfer=[8,16,32] len= data= pass=*

The *Constant Pattern Test* writes a constant data pattern into the frame buffer memory, then reads and compares the written data.

off= is the offset frame buffer address you may specify.

xfer= specifies whether you want the data transfer to be in 8-, 16-, or 32-bit mode.

len= specifies the number of bytes to be written. The default is

```
len=100000
```

which is the hexadecimal equivalent of one megabyte.

data= may be any hexadecimal pattern.

AD *xfer=[8,16,32] comp=[0,1] len= pass=*

The *Address Pattern Test* performs an address-equal-data test in the frame buffer memory. The parameters are the same as those for the Constant Pattern Test, *comp=* defaults to 0.

M *off= len= data= mod=[2 - 5] pass=*

The *Modulo Pattern Test* performs a modulo data pattern test in frame buffer memory. Default parameters are

```
off=0x0
len=0x100000
data=0x555555555
mod=2
pass=1
```

Setting *mod=* to a value of 2 through 5 results in various sequences, as described below.

mod=2

If data is 0x5555aaaa, the complement of that data will be 0xaaaa5555. 0x5555aaaa will be written to the location specified by the *addr=* parameter, and 0xaaaa5555 will be written to the next location.

mod=3

If this parameter is selected, the data pattern will be written to the selected address and to the following address. Then, the complement of the data pattern will be written to the next location.

mod=4

If this parameter is selected, the data pattern will be written to the selected address and to the next two locations. Then the complement of the data pattern will be written to the fourth location.

mod=5

If this parameter is selected, the data pattern will be written to the selected address and to the next three locations. Then the complement of the data pattern will be written to the fifth location.

R *off= xfer=[8,16,32] len= seed= pass=*

The *Random Pattern Test* writes, reads, and compares random data to the frame buffer memory location selected with *off=*, in 8-, 16-, or 32-bit transfer mode.

Entering **r xref=8** selects the random pattern test with an 8-bit transfer mode. Entering 16 or 32 selects the corresponding transfer mode.

len= selects the number of bytes to be written. The default is one megabyte.

seed= may be an integer value from which random patterns are generated. The default is `seed=0`.

The default number of passes is one.

A3

The *Three Consecutive Address Test* writes specific patterns of access cycles into frame buffer memory. The pattern is write-write-read, read-write, and then read to three consecutive addresses. The cycle pattern is executed by incrementing the initial address.

N The *NTA Pattern Test* clears the screen and fills the display with a solid color. This procedure is repeated with different color combinations.

P *off= len= fillbg= data= pass=*

The *Walking Pixel Test* writes its value and verifies it. The background is filled with the *fillbg=* value and verified. The default "fill background" value is zero, which results in a black background. Values from 0x0 to 0xffffffff may be entered after *fillbg=*.

The other parameters behave the same as described for previous tests. Defaults are:

```
off=0x0
len=0x100000
fillbg=0
data=0xffff ffff
pass=1
```

B *off= len= bit= pass=*

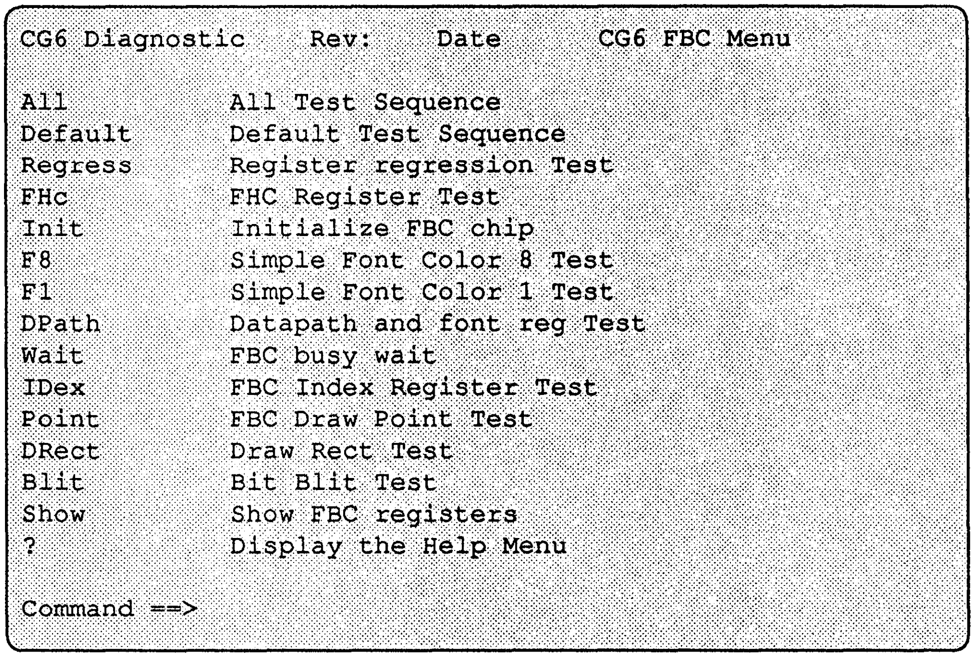
The *Walking Bit Test* positions a white pixel on a black background when the parameter *bit=1* is set. When *bit=0* is entered, a black pixel is positioned on a white background.

Defaults are:

```
off=0x0
len=1000000
bit=1
pass=1
```

12.7. FBC Menu

Selecting **F** from the Main Menu displays the FBC Chip Menu.



```
CG6 Diagnostic    Rev:    Date      CG6 FBC Menu

All          All Test Sequence
Default      Default Test Sequence
Regress      Register regression Test
FHc          FHC Register Test
Init         Initialize FBC chip
F8           Simple Font Color 8 Test
F1           Simple Font Color 1 Test
DPath        Datapath and font reg Test
Wait         FBC busy wait
IDex         FBC Index Register Test
Point        FBC Draw Point Test
DRect        Draw Rect Test
Blit         Bit Blit Test
Show         Show FBC registers
?            Display the Help Menu

Command ==>
```

A

All executes each test in the FBC menu.

D *Default* executes a minimum set of tests to verify the functionality of FBC.

R *pass=*

The *Register Regression Test* performs a walking ones and zeroes on all the FBC read/write registers. Verification is completed by writing As and 5s to the registers. The default number of passes is one.

FH *pass=*

The *FHC Register Test* checks the functionality of the FHC register. This is accomplished by checking the modify flag status. If the flag changes in modification due to x and y, then verification is complete. The default number of passes is one.

I

This *initializes* the FBC registers.

F8 *pass=*

The *Simple Font Color 8 Test* checks the Font registers of the FBC for 8-plane mode. The default number of passes is one.

F1 *pass=*

The *Simple Font Color 1 Test* checks the FBC Font register for 1-plane mode.

DP *pass=*

The *Datapath and Font Register Test* checks ROP and font operations without plane and pixel masks. It test the operations again after modifying the mask. The default number of passes is one.

W The *FBC Busy Wait* test reads the FBC status register and displays a time-out message if the FBC does not respond within an interval of time.

ID *pass=*

The *FBC Index Register Test* uses the index registers to draw points and lines. The default number of passes is one.

P *xval= yval= mask= data= pass= All=[0,1]*

The *FBC Draw Point* test writes one pixel to a location provided by the *xval=* and *yval=* values. If you do not enter options, mask enters a plane-mask value for the pixel. If *all=1*, data and address lines are checked.

Default values are:

```
xval=0x0
yval=0x0
mask=0xff
data=0xff
pass=1
all=1
```

DR *pass=*

The *Draw Rectangle Test* verifies the resolution plane of the monitor by drawing rectangles. The default number of passes is one.

B *pass=*

The *Bit Blit Test* moves the source and destination registers of a drawn rectangle in order to check FBC functionality. The default number of passes is one.

S

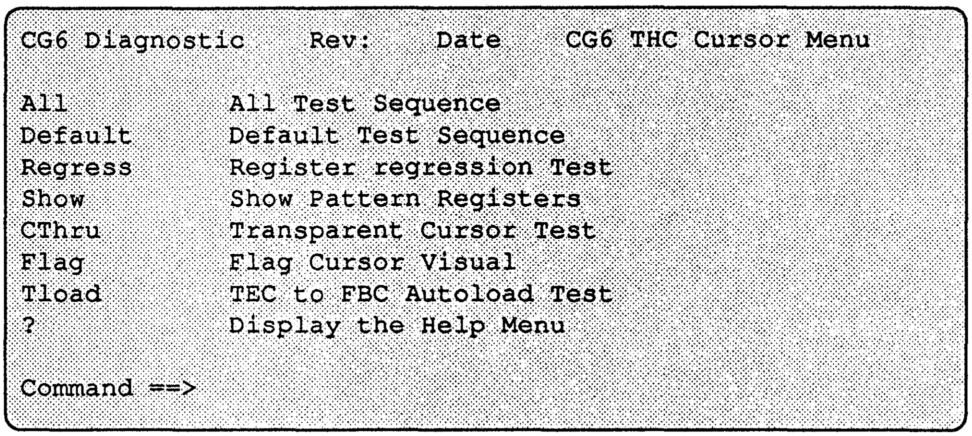
The *Show FBC Registers* option displays the value in the FBCregisters.

?

A question mark displays the FBC Help menu. You must enter Esc to exit from the help menu.

12.8. THC and Autoload Menu

Selecting `Cursor` from the Main Menu displays the THC and Autoload Menu.



```
CG6 Diagnostic    Rev:    Date    CG6 THC Cursor Menu

All          All Test Sequence
Default      Default Test Sequence
Regress      Register regression Test
Show         Show Pattern Registers
CThru        Transparent Cursor Test
Flag         Flag Cursor Visual
Tload        TEC to FBC Autoload Test
?            Display the Help Menu

Command ==>
```

A

All executes each THC Cursor Menu test.

D

Default executes a set of tests that verify the functionality of the THC and Autoload menu.

R *pass=*

The *Register Regression Test* checks the cursor functionality. It verifies the address, plane, A, and B registers, using walking ones. The default number of passes is one.

S *noname*

The *Show Pattern Registers* test displays the pattern registers with names listed. If you add the *noname* option, cursor register values are shown sequentially without names.

CT The *Transparent Cursor Test* fills the frame buffer memory with modulo 5 values.

F *pass=*

The *Flag Cursor Visual* test loads a cursor and moves it, using the TEC hardware configuration address register. The default number of passes is one.

T *pass=*

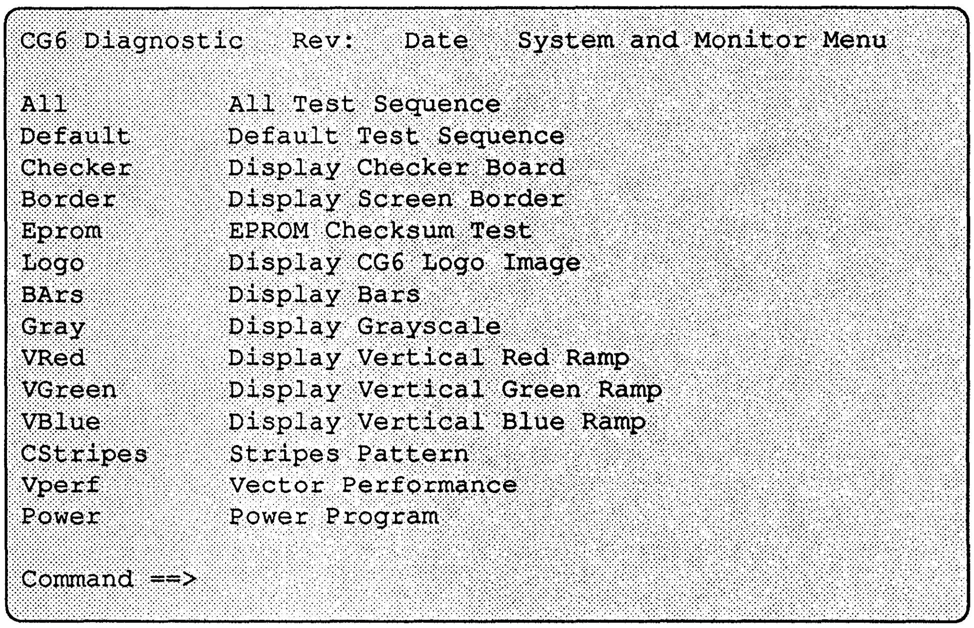
The *TEC to FBC Autoload Test* loads the FBC, using the TEC registers for verification. The default number of passes is one.

?

? displays the THC Cursor Help Menu. You must enter [Esc] to exit from the help menu.

12.9. System and Monitor Menu

Selecting `Sys` from the Main Menu displays the System and Monitor Menu.



```
CG6 Diagnostic    Rev:   Date   System and Monitor Menu

All           All Test Sequence
Default       Default Test Sequence
Checker       Display Checker Board
Border        Display Screen Border
Eprom         EPROM Checksum Test
Logo          Display CG6 Logo Image
BArs          Display Bars
Gray          Display Grayscale
VRed          Display Vertical Red Ramp
VGreen        Display Vertical Green Ramp
VBlue         Display Vertical Blue Ramp
CStripes      Stripes Pattern
Vperf         Vector Performance
Power         Power Program

Command ==>
```

A

All executes each option in the displayed menu.

D

Default executes a minimum set of tests that verify the functionality of System and Monitor menu.

C

The *Display Checker Board* test draws color squares on the monitor that vary in color intensity as each square is drawn from left to right and top to bottom. These color values are read back from the frame buffer and verified.

B

Border tests the resolution plane of a monitor by drawing a border.

E

The *EPROM Checksum Test* reads data from the PROM, calculates the checksum, and verifies the checksum by reading it from the last two bytes of the PROM.

CG6

This test draws CG6 logo after reading it from the PROM.

BA

The *Display Bars* option draws horizontal bars at the top and bottom of the video display, using the DAC register.

G The *Display Grayscale* option draws gray horizontal lines that increase in intensity as the lines approach the center of the video display.

VR

The *Display Vertical Red Ramp* option draws red horizontal lines that increase in intensity as the lines approach the center of the video display.

VG

The *Display Vertical Green Ramp* option draws green horizontal lines that increase in intensity as lines approach the center of the video display.

VB

The *Display Vertical Blue Ramp* option draws red horizontal lines that increase in intensity as lines approach the center of the video display.

CS

The *Stripes Pattern* option draws a pattern of stripes on the screen which is used to identify bad video memory.

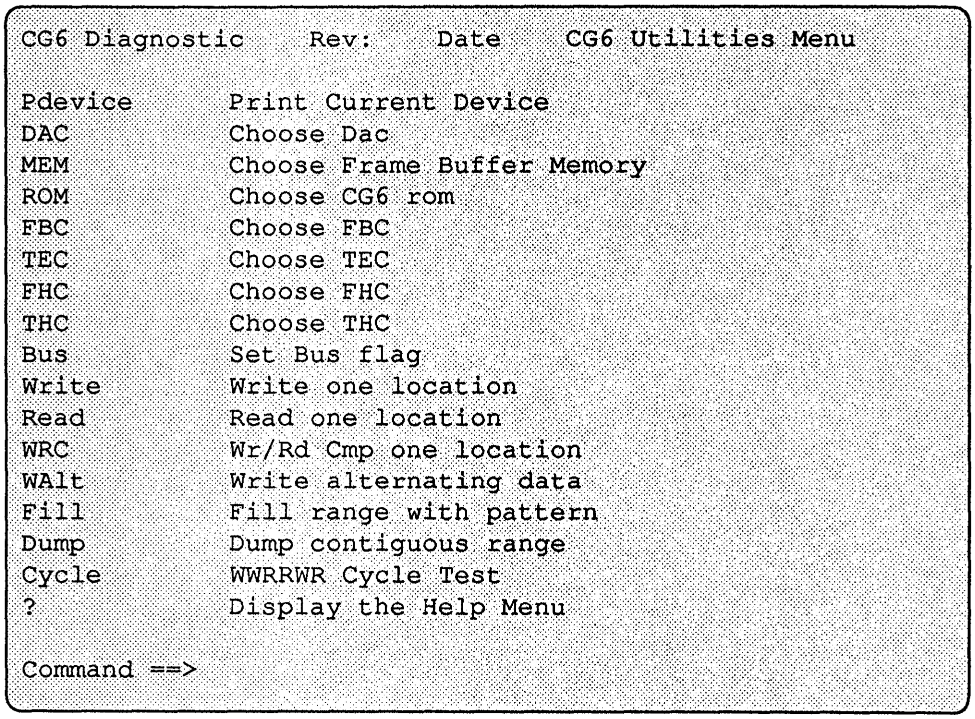
V

The *Vector Performance* test measures CG6 board performance by the number of vectors drawn per second.

P The *Power Program* option tests the blit function, which determines FBC, TEC, and DAC functionality.

12.10. Utilities Menu

Selecting `Util` from the Main Menu displays the Utilities Menu.



```
CG6 Diagnostic    Rev:    Date    CG6 Utilities Menu

Pdevice      Print Current Device
DAC          Choose Dac
MEM          Choose Frame Buffer Memory
ROM          Choose CG6 rom
FBC          Choose FBC
TEC          Choose TEC
FHC          Choose FHC
THC          Choose THC
Bus          Set Bus flag
Write        Write one location
Read         Read one location
WRC          Wr/Rd Cmp one location
WAlt         Write alternating data
Fill         Fill range with pattern
Dump         Dump contiguous range
Cycle        WWRRWR Cycle Test
?            Display the Help Menu

Command ==>
```

In this menu, the DAC, MEM, ROM, FBC, TEC, FHC, and THC items are entered to select a device. The subsequent choices, such as `Bus`, `Write`, and so on, are then performed on the selected device.

P The *Print Current Device* option prints the physical and virtual addresses of the current device, and labels the transfer type (8-, 16-, or 32- bit.)

DAC

DAC selects the Digital-to-analog converter device.

MEM

MEM selects the frame buffer memory.

ROM

ROM selects the CG6 board Read Only Memory.

FBC

FBC selects the FBC registers.

TEC

FHC selects the FHC registers.

THC

THC selects the THC registers.

B *set or reset*

The *Set Bus Flag* test determines the status of the bus error flag. If you add the parameter *set*, the bus error flag is initialized. *reset* clears the error flag condition.

W *Off= Data= Xfer= Scope= Pass=*

The *Write One Location* test provides writing access to any register or memory location. If a device is selected, the offset (*Off=*), transfer (*Xfer=*), and *Data=* parameters may be utilized. Default values are:

```
off=0x0
data=0x12345678
xfer=32
scope=0
pass=1
```

R *Off= Xfer= Scope= Pass=*

The *Read One Location* option provides the scoping facility to read any location in any of the transfer modes. Default parameters are:

```
off=0x0
xfer=32
scope=0
pass=1
```

WRC *Off= Data= Scope= Pass=*

The *Write, Read, Compare One Location* test does a write, read, and compare to a specified location along with scoping procedures. Parameter defaults are:

```
off=0x0
data=0x12345678
scope=0
pass=1
```

W *d1= d2= Xfer= Scope=*

The *Write Alternating Data* test writes alternate data patterns to a specified address location. This test is useful when switching data lines in the scope mode. Default parameters are:

```
d1=0xaaaaaaaa
d2=0x55555555
xfer=32
scope=0
```

F *Off= Xfer= Len= Data=*

The *Fill Range With Pattern* test starts at the specified offset which is filled with data of the transfer type and length specified. Default parameters are:

```
off=0x0
xfer=32
len=0x100000
data=0x5a5a5a5a
```

D *Off= Xfer= Len=*

The *Dump Contiguous Range* test dumps data from the specified offset address to the video monitor using the given transfer type and specified length. Default parameter values are:

```
off=0x0
xfer=32
len=0x100000
```

C *a0= a1= a2= d0= d1= d2=*

The *Write-Write-Read-Read-Write-Read Cycle Test* writes to the virtual address specified by parameters *a0=*, *a1=*, and *a2=* with hexadecimal values provided by parameters *d0=*, *d1=*, and *d2=*. The default values are:

```
d0=5a5a5a5a
d1=a5a5a5a5
d2=5a5a5a5b
```

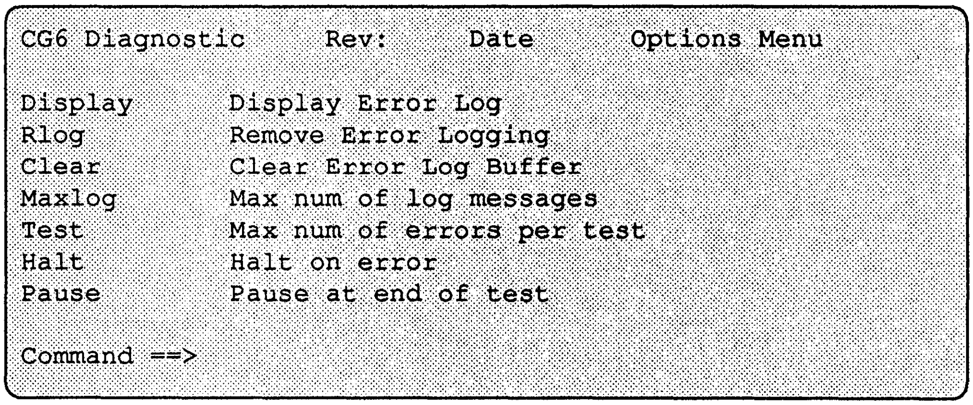
By default, `a0` is initialized to the beginning of the frame buffer memory and `a1` and `a2` are the next two consecutive locations.

?

A question mark displays the Utilities Help Menu. You must enter [Esc] to exit from the help menu.

12.11. Option Menu

Selecting `Option` from the Main Menu displays the Options Menu.



```
CG6 Diagnostic     Rev:     Date      Options Menu

Display       Display Error Log
Rlog          Remove Error Logging
Clear         Clear Error Log Buffer
Maxlog        Max num of log messages
Test          Max num of errors per test
Halt          Halt on error
Pause         Pause at end of test

Command ==>
```

D The *Display Error Log* option shows all of the errors saved in the error log buffer. The Esc key exits this display.

R

The *Remove Error Logging* option either enables or disables error-logging capability. The default enables error logging.

Clear

The *Clear Error Log Buffer* option flushes the error log buffer.

M *n=*

The *Maximum number of log messages* option determines the total number of error messages that can be logged. The decimal number you enter after **n=** is the maximum number of error messages to be logged. Default is `n=100`.

T *n=*

The *Maximum number of errors per test* option specifies the number of errors that can be logged for an individual test. Enter the decimal number after `n=`. The total number of errors cannot exceed the number set with the `max num of log messages` option. The default is `n=10`.

H The *Halt on Error* option determines whether or not a test will stop when an error occurs. The default is to enable the "halt on error" option.

P The *Pause at end of test* option enables or disables the pause at the end of a test. The default is to disabled the pause.

12.12. Error Messages

This section depicts error messages that may appear on the screen during the the tests listed.

PROBE Error Messages

FHCRead

```
***** FHCRead Error: BUS ERROR occurred.
```

FHCWrite

```
***** FHCWrite Error: BUS ERROR occurred.
```

THCRead

```
***** THCRead Error: BUS ERROR occurred.
```

THCWrite

```
***** THCWrite Error: BUS ERROR occurred.
```

FBCRead

```
***** FBCRead Error: BUS ERROR occurred.
```

FBCWrite

```
***** FBCWrite Error: BUS ERROR occurred.
```

TECRead

```
***** TECRead Error: BUS ERROR occurred.
```

TECWrite

```
***** TECWrite Error: BUS ERROR occurred.
```

DFBRead

```
***** DFBRead Error: BUS ERROR occurred.
```

DFBWrite

```
***** DFBWrite Error: BUS ERROR occurred.
```

ROMread

```
***** ROMread  Error: BUS ERROR occurred.
```

TEC Error Messages

Regress

```
tec regress Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

RAnd

```
tec random Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

Wwr

```
***** TEC WWR Error: off(0x ), exp(0x ) obs(0x )
```

Init

```
Illegal resolution res= . res=[1-6]
```

Vidtoggle

No Error Message

Show

If registers have not been initialized then following message is displayed:

```
WARNING TEC/THC has not been initialized
```

Ezmat

```
tec ezmat Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

Mat

```
TEC Mat test Error: off(0x )
```

```
exp(0x ), obs(0x ), xor(0x )
```

IRQ

```
***** Tec IRQ Test: IRQ wait exceeded.
***** Tec IRQ test: errant IRQ received
```

Cpuintr

If the interrupt service routine is not installed then this message is displayed:

```
***** ERROR in irq install
UNIX: cannot install isr
```

CG6intr

No Error Message

IFlag

No Error Message

DAC Error Messages

Swr

```
***** DAC swr Error: off( ), exp( ) rcv( ) xor( )
```

ADdress

```
***** DAC addr reg Error: addr( ), exp( ) rcv( ) xor( )
```

Cmd

```
***** DAC ctrl reg Error: addr( ), exp( ) rcv( ) xor( )
```

Rmask

```
***** DAC ctrl reg Error: addr( ), exp( ) rcv( ) xor( )
```

Blink

```
***** DAC ctrl reg Error: addr( ), exp( ) rcv( ) xor( )
```

COlor

The following messages are displayed if an error is found for the respective color:

```
***** Dac Color Test: addr=0x , red, exp(0x), rcv(0x )
***** Dac Color Test: addr=0x , green, exp(0x), rcv(0x )
***** Dac Color Test: addr=0x , blue, exp(0x), rcv(0x )
```

Ovly

For all Colors this message is displayed:

```
***** DAC ovly Error: addr( ), exp( ) rcv( ) xor( )
```

Init

No Error Message

ROvly

No Error Message

WOvly

No Error Message

MEM Error Messages

```
Constant Pattern xfer=8 Test Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
   Constant Pattern xfer=16 Test Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
   Constant Pattern xfer=32 Test Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

ADdress

```
Address Pattern Test Passes = , Errors =
Address Pattern Test Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

Modulo

```
Modulo Pattern Test Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

Random

```
Random Pattern Test Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

A3

```
***** Mem Cycle Test Error: Off=0x

d0=0x 0x d1=0x 0x d2=0x 0x
```

CRetent

```
Constant Data Retention Test Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

RRetent

```
Random Data Retention Test Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

Nta

```
NTA pattern (x.y) Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

where x can be a value from 1 to 9 and y can be a value of 1 or 2

Pixwalk

```
Walking Pixel Test Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

Bitwalk

```
Walking Bit Test Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

FBC Error Messages

Regress

If an error occurs in the I/O (read/write) the following message is displayed:

```
FBC regress Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

Else this is displayed:

```
   FBC regress Error: off(0x )

wrote (0x ) exp(0x ), obs(0x ), xor(0x )
```

FHc

If an error occurs in the I/O (read/write) the following message is displayed:

```
   FHC reg test Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

Else this is displayed:

```
   FHC reg test Error: off(0x )

wrote (0x ) exp(0x ), obs(0x ), xor(0x )
```

Init

No Error Message

F8

```
Illegal address offset 0x

wrc Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

F1

```
***** Error: dst[ ]= , exp[ ]= obs[ ]=
```

DPath

```
***** Error: dst[ ]= , exp[ ]= obs[ ]=
```

Wait

If the FBC does not respond in a specified time the following message is displayed:

```
FBC timeout
```

IDex

The following error message is displayed for x, y, Line, and Point tests:

```
***** IDex point Error: exp(0x ), obs(0x )
***** IDex line Error: exp(0x ), obs(0x )
```

DRect

One of the following errors will be displayed for DRect test:

```
***** draw wait error.  status = 0x
***** blit status error. Could not launch draw
```

Blit

```
***** Blit Test Error: Could not initialize frame
buffer
***** Blit Test Error:off(0x )

exp(0x ), obs(0x ), xor(0x )
```

Show

No Error Message.

CURSOR Error Messages

Regress"

```
THC Cursor Regress Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

Show

No Error Message

CThru

No Error Message

Flag

No Error Message

Tload

No Error Message

SYS Error Messages

Checker

```
***** Checker Error: Checker did not verify
```

Border

No Error Message

Eprom

If the CPU is not a Sun4 workstation, then the following message is displayed:

```
EPROM CHECKSUM Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

Else this error is displayed:

```
SBUS EPROM CHECKSUM Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

CG6

No Error Message

BArs

No Error Message

`Gray, VRed, VGreen, VBlue`

No Error Message

CStripes

Resolution does not have width=1152 and stripes cannot be executed.

Vperf

No Error Message

Power

No Error Message

UTILITIES Menu Error Messages

```
Pdevice, DAC, MEM, ROM, FBC, TEC, FHC, THC
Illegal device.  Recheck the code!
llegal Xfer size. valid Xfer=[8,16,32] Return
```

Bus

No Error Message

Write

```
walt Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

Read

No Error Message

WRC

```
wrc Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

WAlt

```
walt Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

Fill, Dump

No Error Message

Cycle

If the address is not aligned, the following message is displayed:

```
Warning, unaligned vaddr: a0=0x , a1=0x , a2=
Cannot execute.
addr3test Error: off(0x )

exp(0x ), obs(0x ), xor(0x )
```

13

CG8 Diagnostics

13

CG8 Diagnostics

13.1. Introduction

The CG8 board is a low-cost frame buffer that can display 24-bit color with a resolution of 1152 x 900. This frame buffer interfaces with a P4 daughter board that can be installed in the Sun-3/60, Sun-3/80, Sun-3/460, Sun-4/110, and Sun SPARC system 330

The CG8 diagnostic tests the following sections of the board: overlay plane memory, enable plane memory, frame buffer memory, RAMDAC, and interface circuits. The CG8 Diagnostic ensures that the CG8 frame buffer works correctly.

NOTE *This diagnostic can be executed while the CG8 board functions as a console device. A serial terminal or a dual frame buffer configuration can be used for testing the CG8.*

To troubleshoot the CG8, you must use a secondary display device, such as a serial terminal or a second frame buffer. You can, however, use a single-headed system to verify that the board is functional. To do so, enter the Exec Environment Menu. From this menu, enter:

```
info=0
```

Then exit from the Environment Menu and go to the Diagnostics Menu. From this menu, select the CG8 Menu. This sequence of actions enables the diagnostics to run on a single-headed system (with the CG8 board functioning as a console device).

Before running this diagnostic, refer to *Chapter 2* for information on booting the SunDiagnostic Executive. When you are ready to run the diagnostic, select **CG8** from the Diagnostics Menu.

13.2. Parameter Definitions

This section lists a brief description of each parameter used with the CG8 diagnostics. If a test accepts parameters, they can be entered when you select a test to execute. Otherwise, they are ignored.

address=
The *address* parameter selects random addresses to be generated along with random data.

pattern=
The *pattern* parameter selects the data pattern that is written into memory for comparison.

pass=

The *pass* parameter selects a decimal number that indicates the number of cycles used to execute the test.

scope

The *scope* parameter enables scope looping when a failure occurs.

stop

The *stop* parameter is an integer value that represents the number of test failures that should occur before terminating the current test.

data=

The *data* parameter is a value that is written and compared against the default.

offset=

The *offset* parameter is the starting location in memory to be tested. The default value is 0.

memory=

The *memory* parameter designates the memory being tested. It has a default value of 0. See the following table for alternate value settings.

Table 13-1 *Memory Designations*

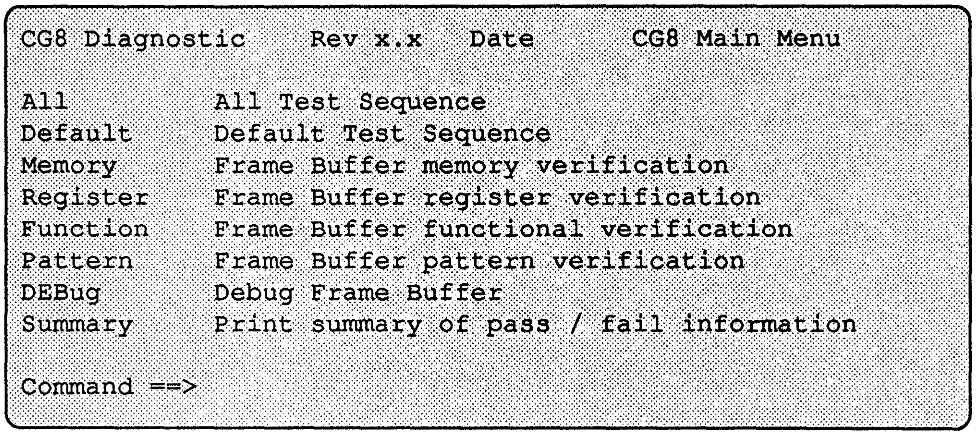
Value	*Setting*
0	all three memories
1	enable memory
2	overlay memory
3	frame buffer memory

sequence=

The *sequence* parameter determines the direction of the test's execution, either *address up*, the default, or *address down.*

13.3. The Main Menu

The Main Menu provides these choices. To make a selection, enter the name of the selection or the upper case characters of that menu selection (an abbreviation). Parameter options are discussed in the previous section.



```
CG8 Diagnostic    Rev x.x   Date      CG8 Main Menu

All         All Test Sequence
Default     Default Test Sequence
Memory      Frame Buffer memory verification
Register    Frame Buffer register verification
Function    Frame Buffer functional verification
Pattern     Frame Buffer pattern verification
DEBug       Debug Frame Buffer
Summary     Print summary of pass / fail information

Command ==>
```

a *pass= scope stop=*

The *all test sequence* option of the Main Menu executes all memory verification tests, register tests, and function tests. See the section entitled *Parameter Definitions* for an explanation of these terms.

Note Execute the *Initialize Frame Buffer* Debug Menu option before running the *All Test Sequence.*

d *pass= scope stop=*

Option **d** selects a *Default Test Sequence.* This option executes a default set of memory tests, register tests, and function tests.

m

The *frame buffer memory verification* Main Menu test displays the Memory Menu. Once selected, the Memory Menu prompts you for a memory selection. The Memory Menu is described in the section entitled *Memory Menu.*

r

The *frame buffer register verification* option of the Main Menu displays the Register Menu. Once selected, the Register Menu prompts you for a register subtest selection. The Register Menu is described in the section entitled *Register Menu.*

f

Selecting *frame buffer functional verification* from the Main Menu displays the Functional Menu. Once selected, the Functional Menu prompts you for a functional subtest selection. The Functional Menu is described in the section titled *Functional Menu.*

p

Selecting *frame buffer pattern verification* from the Main Menu displays the Pattern Menu. The Pattern Menu prompts you for a pattern subtest selection. The Pattern Menu is described in the section titled *Pattern Menu.*

deb

Selecting *debug frame buffer* from the Main Menu displays the Debug Menu. The Debug Menu prompts you or a debug subtest selection. The Debug Menu is described in the section titled *Debug Menu.*

s

Selecting *print summary* from the Main Menu displays a summary of PASS/FAIL information on all previously executed tests. The display prints the group, register, memory, and function. Included in this summary is the actual number of passes, fails, and loops. If a failure occurs, four asterisks print to the side of the subtest name. The following screen display is an example of a failed test:

```
Register

Test                 # Passes   # Fails    # Loops
----------------     --------   --------   --------
Prom checksum           0          1          1      ****

Command ==>
```

13.4. Memory Menu

This menu displays all memory-related tests. To execute all of the tests, select **All**.

```
CG8 Diagnostic   Rev:  1.1    Date     Memory Menu

All            All Test Sequence
Default        Default Test Sequence

Constant       Constant pattern test
Walk           Walking zero/one pattern test
WEdge          Wedge zero/one pattern test
ADdress        Address pattern test
Random         Random pattern test
Mats           Mats pattern test
Nta            NTA pattern test
B              Buchanan Memory pattern test
Size           Verify different data sizes into memory.
Initialize     Initialize frame buffer.

Command ==>
```

a *pass= scope stop=*

The *all test sequence* option executes the following: patterns, all zeros, all ones, checkerboard, inverse checkerboard, walking ones, walking zeros, wedge ones, wedge zeros, address-up sequence, address-down sequence, random address, random data, three passes of mats tests, and eight passes of NTA tests. All memories are also tested.

d *pass= scope stop=*

The *default test sequence* option executes a walking ones pattern and a mats pattern in all memories.

c *pass= scope stop= data= offset= memory=*

The *constant pattern test* option executes a constant pattern memory test. This test stores a given pattern throughout enable memory, overlay memory, and frame buffer memory. The memory is compared against the pattern to determine if any errors have occurred.

w *pass= scope stop= data= offset= memory=*

The *walking zero/one pattern test* option executes a walking-pattern memory test. This test walks a zero or a one through a 32-bit pattern in memory. The test verifies the pattern is correct in memory. The following two tables show examples of walking patterns:

Table 13-2 *Walking One Pattern*

Location	*Pattern*
n	0001
n+1	0010
n+2	0100
n+3	1000
: : :	: : : :
: : :	: : : :

Table 13-3 *Walking Zero Pattern*

Location	*Pattern*
n	1110
n+1	1101
n+2	1011
n+3	0111
: : :	: : : :
: : :	: : : :

we *pass= scope stop= data= offset= memory=*

The *wedge zero/one pattern test* option executes a wedge memory pattern test. The test writes either a wedge zero or a wedge one pattern into memory, then compares the written memory with the wedge pattern. The following are examples of wedge patterns:

Table 13-4 *Wedge One Pattern*

Location	*Pattern*
n	0001
n+1	0011
n+2	0111
n+3	1111
:::	::::
:::	::::

Table 13-5 *Wedge Zero Pattern*

Location	*Pattern*
n	1110
n+1	1100
n+2	1000
n+3	0000
:::	::::
:::	::::

ad *pass= scope stop= sequence= offset= memory=*

The *address pattern test* option verifies the memory address function. A sequence is stored starting at the beginning or at the end of memory. After writing to memory, the test verifies the memory contents. The following tables show examples of address patterns:

Table 13-6 *Address-Up Pattern*

Location	*Pattern*
n	0001
n+1	0010
n+2	0011
n+3	0100
:::	::::
:::	::::

Table 13-7 *Address-Down Pattern*

Location	*Pattern*
n	0100
n+1	0011
n+2	0010
n+3	0001
:::	::::
:::	::::

r *pass= scope stop= address= memory=*

The *random pattern test* option selects a random data pattern with an optional random address to be written to and verified. By default, only random data is generated across the entire address range. However, by adding the *address parameter,* the test checks only random addresses.

m *pass= scope stop= offset= pattern= memory=*

Th *mats pattern test* option mats a test that is written to memory, then verified.

n *pass= scope stop= memory=*

This option selects the *NTA pattern test,* which is executed in memory. The NTA pattern test determines pattern-sensitive faults and coupling faults in memory arrays. The following are examples of NTA patterns:

Initialize memory to zeroes.

pass 1

Starting at bottom of memory, change all zeroes to ones. Readback ones starting from the top of memory.

pass 2

Starting at bottom of memory, change all ones to zeroes. Readback zeroes starting from the top of memory.

pass 3

Starting from the top of memory, change all zeroes to ones. Readback ones starting at the bottom of memory.

pass 4

Starting from the top of memory, change all ones to zeroes. Readback zeroes starting at the bottom of memory.

pass 5

Starting at the bottom of memory, change all zeroes to ones to zeroes. Readback zeroes from the top of memory.

pass 6

Starting at the top of memory, change all zeroes to ones to zeroes. Readback zeroes from the bottom of memory.

Reset memory to ones.

pass 7
Starting at the bottom of memory, change all ones to zeroes to ones. Readback ones from the top of memory.

pass 8
Starting at the top of memory, change all ones to zeroes to ones. Read back ones from the bottom of memory.

b *pass= scope stop= data= offset= memory=*
This option selects and executes the *Buchanan memory test.* This test writes to a memory location, then immediately reads it. The read location is specified in the command line. This sequence is done on the entire memory array. The pattern verifies the refresh function of both the refresh state machine and the RAMDAC read state.

s *pass= scope stop= data= offset= memory=*
Option **s** tests to *verify different data sizes into memory.* It verifies that memory can be written to and read from in different data sizes. Memory is tested with data sizes of 8, 16, and 32 bits.

13.5. Register Menu

The Register Menu allows access to all register subitems. There are four main register tests: *Access, Crosstalk, BRead,* and *BWrite.*

```
CG8 Diagnostic    Rev:  1.1    Date    Register Menu

All              All Test Sequence

ACcess           Register read / write
Crosstalk        Register Crosstalk
Prom             Verify Prom Checksum
BRead            Test tight read cycles.
BWrite           Test tight write cycles.
Initialize       Initialize frame buffer

Command ==>
```

a *pass= scope stop=*

Selecting the *all test sequence* option executes a read/write test on both the access register and the bit crosstalk register.

ac *pass= scope stop=*

Option *access* executes read/write pattern tests to all frame buffer registers. Each bit is toggled ON and then OFF to determine accessibility. This test is not a crosstalk test.

c *pass= stop=*

Selecting the *register crosstalk* option executes a crosstalk test on all frame buffer registers and look-up table memory. This test does a bit crosstalk on all registers by setting a bit in the target register, while verifying the remaining registers for corruption. If corruption exists, the test flags the corrupt register. The register access test should pass completely before executing this test. This action prevents other errors from occurring.

p *pass=*

The *verify PROM checksum,* option performs a checksum comparison to verify the PROM. The last word of the PROM contains the verifying checksum comparison.

br *pass=*

Option *BRead* executes the burst read test. This test exercises the memory state machine by pulling data from the state machine in rapid sessions. This routine is optimized to run continuous read instructions back-to-back.

bw *pass=*

Option *BWrite* executes the burst write test. The burst write test exercises the memory while it pushes data to the state machine as fast as the host machine can send it. This routine is optimized to run continuous write instructions.

13.6. Function Menu

The Functional Menu lists all available functional tests.

```
CG8 Diagnostic    Rev:  1.1   Date    Function Menu

All               All Test Sequence
Default           Default Test Sequence
Ramdac            Verify ramdac functional
INTerrupt         Verify interrupt function
Mask              Verify read mask control
Overlay/enable    Overlay/enable Test sequence
Initialize        Initialize the frame buffer

Command ==>
```

a *pass= scope*

This option, *all test sequence,* executes all frame buffer functional tests. This selection verifies the RAMDAC functional tests, P4 vertical retrace interrupt, blink mode, read mask, overlay, and enable.

d *pass= scope*

The *default test sequence* performs a quick verification by executing the RAMDAC and overlay/enable tests.

r *pass= scope stop=*

Option **r** executes the *verify RAMDAC functional* test for the basic color frame buffer. This test places RAMDAC in test mode, then verifies the look-up table addressing *from the frame buffer memory* and *to MDAC.* Data is looped back before it enters the DAC.

int *pass= scope stop=*

This option tests the *verify interrupt function.*

m *pass= scope stop=*

The *verify read mask control* option verifies the read mask function in all RAMDAC look-up tables. The read mask is verified by writing all ones into color frame buffer memory. The same sequence pattern is executed in the color look-up table. By looping back, the DAC data can be compared to verify correct operation.

o *pass= scope stop=*

Option *overlay/enable test sequence* tests up to the DAC, while the RAMDAC is in test mode. This selection tests the *color palette addressing from overlay memory, enable memory,* and *overlay read mask.*

Overlay color addressing is verified by writing a specific pattern into the color look-up table. The overlay memory is written with three different patterns. Overlay 2, however, cannot be used because of the gating of enable planes.

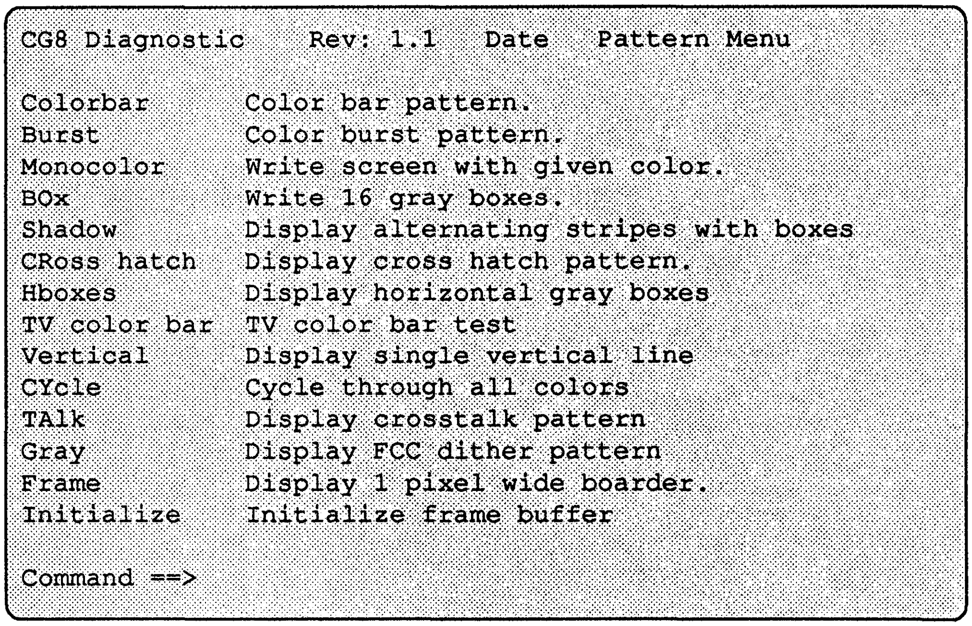
Data is looped back from the input of the RAMDAC to verify correct operation. Overlay read mask is verified by writing all ones into overlay memory while disabling overlay planes. This action creates different color data sent to the RAMDAC. Data is verified by looping back RAMDAC data to the CPU for comparison.

`int` *pass= scope stop=*

This option, *initialize the frame buffer*, tests that function.

13.7. Pattern Menu

The Pattern Menu allows access to four patterns: colorbar, burst, monocolor, and box. These tests are manual tests only. Examples are visual, DVM, and scope.



```
CG8 Diagnostic    Rev: 1.1   Date   Pattern Menu

Colorbar       Color bar pattern.
Burst          Color burst pattern.
Monocolor      Write screen with given color.
BOx            Write 16 gray boxes.
Shadow         Display alternating stripes with boxes
CRoss hatch    Display cross hatch pattern.
Hboxes         Display horizontal gray boxes
TV color bar   TV color bar test
Vertical       Display single vertical line
CYcle          Cycle through all colors
TAlk           Display crosstalk pattern
Gray           Display FCC dither pattern
Frame          Display 1 pixel wide boarder.
Initialize     Initialize frame buffer

Command ==>
```

`c`

The *color bar pattern* option executes a visual test that displays a color bar pattern on the monitor.

`b`

Selecting the *color burst pattern* option executes a visual single-color burst test. The operator of the test is expected to distinguish among the various colors displayed.

`m` *red= green= blue=*

The *monocolor* option writes one color to the screen to allow scope measurement of the Brooktree RAMDAC output level.

The colors *red, green, and blue* are values between 0 and 255. Each specific value writes to an individual RAMDAC color palette.

`bo`

Selecting the *box* option displays 16 different gray boxes on the screen. The initial gray value starts at six per cent and goes to ten per cent. Each gray value is incremented by six per cent.

s

The *shadow* option displays a shadow pattern. The pattern consists of alternating white and black vertical lines for the background. One white box is displayed in the upper middle of the screen, and one black box is displayed in the lower middle of the screen. This pattern is primarily used to verify the ring at the left and right edges of the two boxes. The pattern is displayed in overlay and enable memory only.

cr *xstart= ystart= width= vertical*

This option *displays a crosshatch pattern.* A series of evenly spaced vertical and horizontal lines are displayed on the screen. This pattern verifies correct data mapping between the memory and the RAMDACs. The crosshatch pattern is only displayed in overlay and enable memory.

xstart is the *beginning x pixel location* to start the vertical lines. *ystart* indicates the *starting pixel* for the horizontal lines. *width* denotes the *space between* consecutive lines in pixels. *vertical* displays only vertical lines. If not invoked, horizontal and vertical lines are available.

h

This option, *display horizontal gray boxes,* displays six boxes with perpendicular lines to the background. This is a video memory to RAMDAC- mapping pattern.

tv

Option *TV color bar test* displays the TV color pattern for color addressing. This pattern verifies RAMDAC synchronization by placing solid colors next to each other. If synchronization is lost, some color bars will have a black gap between them. This pattern uses color memory only.

v *column*

This option *displays a single vertical line.* This pattern verifies that the number of pixel clocks is correct. The test places a green vertical line at a user-specified horizontal location. *Column* is an integer value indicating the horizontal offset of the vertical line.

cy

The *cycle through all colors* option cycles through all primary and secondary colors. The RAMDACs are tested by this pattern to ensure that the DACs can change to the appropriate colors.

ta *half*

The *display crosstalk pattern* option displays the video crosstalk pattern. Wide stripes are placed into color memory, while overlay and enable memory are filled with ones. The result is a white screen without vertical abnormalities.

When *half* is present, this test places only the vertical bars in the lower half of color memory.

g

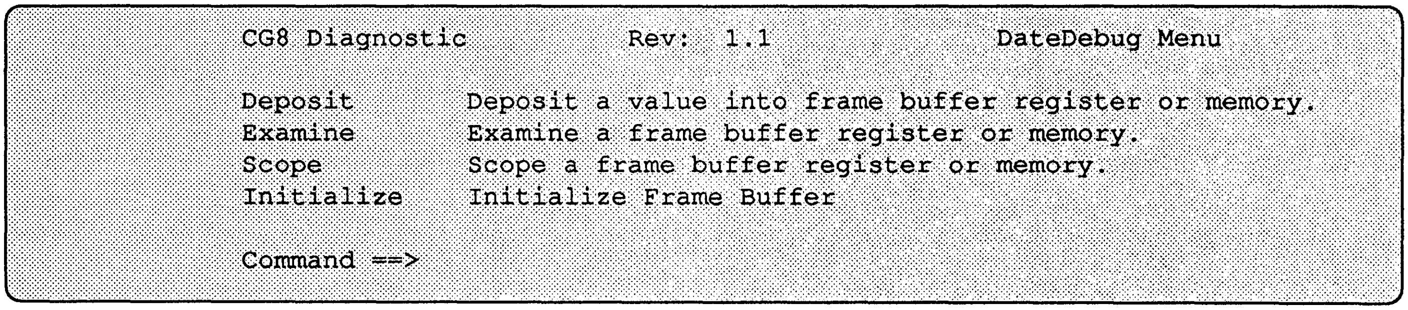
The *display FCC dither pattern* option specifies the pattern used by FCC testing. Alternating pixels are toggled to give a maximum frequency. This pattern uses all memories.

f

This option *displays a one pixel-wide boarder.* This display verifies video memory to RAMDAC mapping. The pattern uses overlay memory.

13.8. Debug Menu

The Debug Menu allows access to three functions: **deposit, examine, and scope.** Debug and checkout of the CG8 frame buffer receives access to all registers and memory locations.



```
CG8 Diagnostic          Rev:  1.1              DateDebug Menu

Deposit       Deposit a value into frame buffer register or memory.
Examine       Examine a frame buffer register or memory.
Scope         Scope a frame buffer register or memory.
Initialize    Initialize Frame Buffer

Command ==>
```

d *address data*

The *deposit* option allows data to be written to any frame buffer register. Use this option to debug both code and faulty hardware in the manufacturing environment. *Deposit* writes only in the longword data format used by the P4 interface.

Both addresses and data can be specified in either hex 0x, decimal, or octal 0. Address values can be specified in single or multiples. Multiple addresses are separated with a colon(:). The following is an example of a separated address:

```
address1:address2
```

The data value must be separated from the address with a space. The following is an example of *deposit:*

```
d 0x400000:0x40000C 123
```

The previous example writes "123" to locations "0x400000-0x40000C". The following example writes "0xCCCCFFFF" into address "0x500000":

```
d 0x500000 0xCCCCFFFF
```

e *address*

The *examine frame buffer register or memory* option examines any frame buffer address. It is similar to *Deposit* because you can examine code. *Examine* reads memory in longword format.

Address can be either a range of addresses or a single address. The syntax is *address1:address2* for a range of addresses. Address can be any base denoted by the 'C' extensions `0x - hex, 0 - octal`. There is no designation for decimal.

s *address data= r address*

The *scope* option allows continuous reading and writing (scoping) to any address on the frame buffer. Scoping is done on a longword boundary, where *address* is the actual address to scope. *Data* should be specified only when in enabled writing mode. To specify a write address, prefix the address with a **w.** For example, scoping at address "0x5e000" with data of "0x1234567" would be:

```
sc w0x5e000 0x1234567
```

Also, both reading and writing can be specified on the same line:

```
sc r0x5e000 w0x5f000 0x12345 r0x50000
```

If the `r` and `w` are omitted, a read from the address occurs followed by a write with the same data. Scope looping on one location is recommended.

i

Option *initialize frame buffer* executes a menu command, which initializes the CG8 frame buffer. All frame buffer memory and look-up tables are reset to zero. All registers are defaulted to disable screen display. Use this option before entering the debug option and before running the **All** option on the Main Menu.

13.9. Memory Map for Debugging

The following table should be used as a debugging aid for the CG8 board. It indicates the physical address and location in memory according to engineering specifications.

Memory Map		
0x200000	0x200008	RAMDAC
0x300000	0x300004	P4 Register
0x380000	0x3C0000	PROM
0x400000	0x41FA40	Overlay Memory
0x600000	0x61FA40	Enable Memory
0x800000	0xBF4800	Color Memory

14

CG9 Color Graphics Diagnostic

14

CG9 Color Graphics Diagnostic

14.1. Introduction to the CG9

The CG9 is another addition to the Color Graphics family of frame buffer boards, capable of displaying 24-bit color with a resolution of 1152 x 900. It is a VME slave device with a P2 interface. The CG9 enhances the capabilities of the GP2 (Graphics Processor2) in RGB (red, green, blue) shading and graphics acceleration.

14.2. What This Chapter Contains

This chapter describes the CG9 Diagnostic. Each diagnostic test menu is shown, followed by a description of each command. The end of the chapter contains a glossary.

14.3. Overview of the Diagnostic

The CG9 Diagnostic tests the following sections of the CG9 frame buffer board:

- Control space registers
- Overlay memory
- Enable memory
- RGB memory
- Double buffering
- BITBLT
- RAMDACs

The parameters of each test are given default values upon execution. Online help is provided. The CG9 Diagnostic also generates and stores error messages for later retrieval.

14.4. Hardware Requirements

To run the CG9 Diagnostic, the system being tested must have the following:

- A CPU board with VME interface
- A CG9 frame buffer
- A color monitor
- A keyboard
- A boot device (local disk, local tape, or remote disk over Ethernet)

14.5. User Interface

The user interface of the CG9 Diagnostic adheres to the standards of the Exec menu system. Each menu option may be selected by typing the letter or letters displayed in upper case in the column on the left side of the menu.

Additional parameters and options may be specified on the command line. For a complete discussion of the Exec command line syntax, refer to Chapter 2, "Using the SunDiagnostic Executive."

A Main Menu and four submenus are provided. The options on the submenus provide for testing the functional components separately or in a continuous sequence. All and Quick Test Sequences are available.

To move back one level in the menu hierarchy, press [ESC] then press [RETURN]. To leave the CG9 Diagnostic and return to the Exec, press [ESC] then press [RETURN] when the Main Menu is displayed.

To display online Help for any menu option, enter the following on the command line:

```
? option_name
```

14.6. Starting the Diagnostic

NOTE *Before you run this diagnositc, you should set the* `info=` *environment variable to zero:*

```
info=0
```

from the Exec's Environment Menu. This setting suppresses all informational messages. If this variable is set to any other value, the diagnostic will send messages to the frame buffer under test.

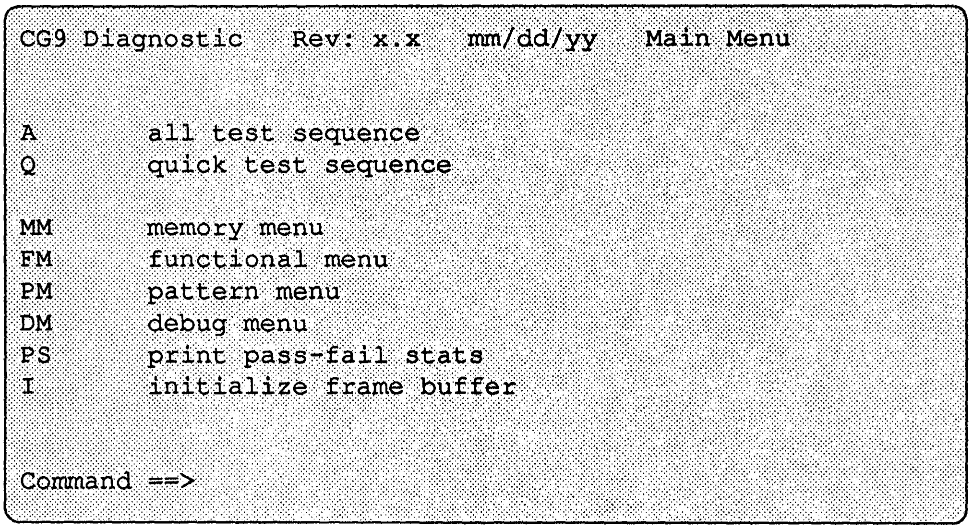
For information on starting the Exec, refer to Chapter 2, "Using the SunDiagnostic Executive." After you have started the Exec, choose the CG9 Diagnostic from the Diagnostics Menu.

14.7. The Diagnostic Menus

This section of the chapter provides a modular description of the CG9 Diagnostic, beginning with the Main Menu and working down through the options available on each of the submenus.

Main Menu

The Main Menu, which displays when you start the CG9 Diagnostic, provides access to the submenus of the individual tests:



```
CG9 Diagnostic   Rev: x.x   mm/dd/yy   Main Menu


A       all test sequence
Q       quick test sequence

MM      memory menu
FM      functional menu
PM      pattern menu
DM      debug menu
PS      print pass-fail stats
I       initialize frame buffer


Command ==>
```

The All and Quick Test Sequence options on the Main Menu take optional environmental variables. If you specify values for these variables on the command line, those values override the settings of the Exec global environmental variables for that sequence of tests. If you do not specify values on the command line, the default values are used. The meanings of the variables, and their default values, are shown in the following table:

Variable	*Description*	*Default*
`Pass`	Pass count.	1
`STop`	Stop on error count.	1
`SCope`	Scope loop on error initiated if this variable is entered on the command line.	No looping

All Test Sequence

A *p= st= sc*

The *All Test Sequence* option on the Main Menu executes an exhaustive test sequence that fully exercises the functional blocks of the CG9. This command causes the All Test Sequence to be executed in each of the three test submenus.

Quick Test Sequence

Q *p= st= sc*

The *Quick Test Sequence* option on the Main Menu executes a brief series of tests for quick functional verification. This command causes the Quick Test Sequence to be executed in the first two test submenus.

Memory Menu

MM

The *Memory Menu* option on the Main Menu displays a submenu that allows you to test the functionality of the CG9 memory and RAMDAC registers. The options on this menu are described in detail in the section entitled "Memory Menu," later in this chapter.

Functional Menu

FM

The *Functional Menu* option on the Main Menu displays a submenu that allows you to run a series of functional tests on the components of the CG9. The options on this menu are described in detail in the section entitled "Function Menu," later in this chapter.

Pattern Menu

PM

The *Pattern Menu* option on the Main Menu displays a submenu that allows you to check the display of various patterns. The options on this menu are described in detail in the section entitled "Pattern Menu," later in this chapter.

Debug Menu

DM

The *Debug Menu* option on the Main Menu displays a submenu that allows you to edit and display the contents of the CG9 frame buffer and perform specific debugging operations. The options on this menu are described in detail in the section entitled "Debug Menu," later in this chapter.

Print Pass-fail Stats

PS

The *Print Pass-fail Stats* option on the Main Menu displays the pass-fail statistics of all commands executed during the current session. As shown in the following sample, this option displays the menu name, the command name, the number of passes, and the number of errors.

```
MENU        COMMAND      PASSES      ERRORS
-----------------------------------------
memory        co           1           1
memory        wa           1           0
function      sb           1           0
```

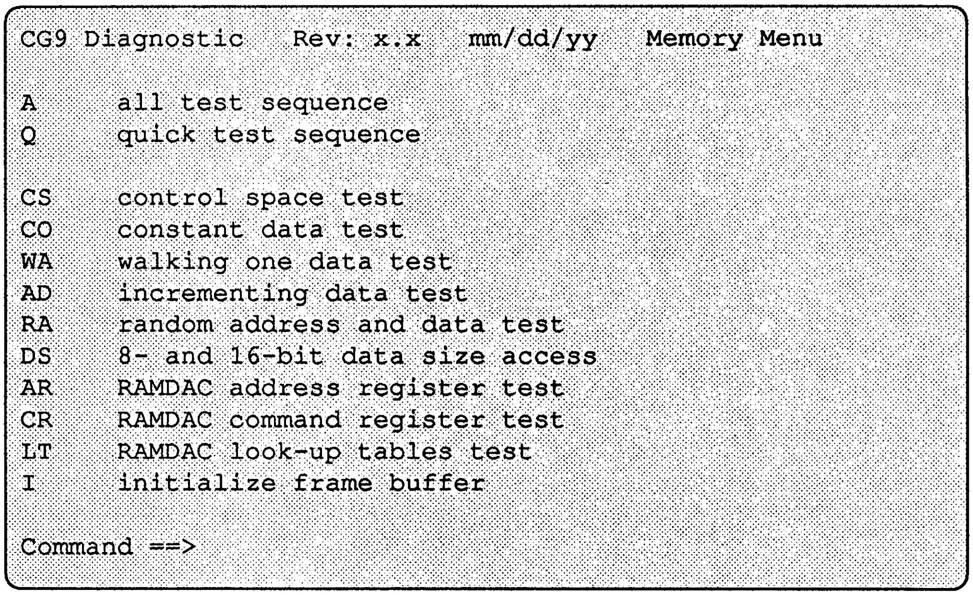
Initialize Frame Buffer

I

The *Initialize Frame Buffer* option on the Main Menu initializes the CG9 frame buffer to its default state.

Memory Menu

When you choose MM from the Main Menu, the Memory Menu is displayed:



```
CG9 Diagnostic    Rev: x.x    mm/dd/yy    Memory Menu

A      all test sequence
Q      quick test sequence

CS     control space test
CO     constant data test
WA     walking one data test
AD     incrementing data test
RA     random address and data test
DS     8- and 16-bit data size access
AR     RAMDAC address register test
CR     RAMDAC command register test
LT     RAMDAC look-up tables test
I      initialize frame buffer

Command ==>
```

The options on this menu test the memory and the RAMDAC registers. The environmental variables and other parameters that you may specify on the command line are summarized in the following table:

Variable	*Description*	*Default*
`Pass`	Pass count.	1
`STop`	Stop on error count.	1
`SCope`	Scope loop on error initiated if this variable is entered on the command line.	No looping
`Block`	Type of memory block. Overlay memory (1) Enable memory (2) RGB memory (3)	3
`Data`	Hex data pattern.	55555555

All Test Sequence

A

The *All Test Sequence* option on the Memory Menu executes all of the tests on the menu in order. The Constant Data Test executes with the following data patterns on all memory blocks (overlay, enable, and RGB):

```
00000000
55555555
AAAAAAAA
FFFFFFFF
```

The Walking One Data Test, Incrementing Data Test, and Random Address and Data Test also execute on all three memory blocks.

Quick Test Sequence

Q

The *Quick Test Sequence* option on the Memory Menu executes the following tests, using the default parameter values:

- Control Space Test
- Constant Data Test
- 8- and 16-bit Data Size Access
- RAMDAC Address Register Test
- RAMDAC Command Register Test
- RAMDAC

Control Space Test

CS *p= st= sc*

The *Control Space Test* option on the Memory Menu performs a write-read-compare on the CG9 interrupt vector, the GP2 interrupt vector, the Plane Mask Register, and the Double-buffer Register using the following data patterns:

```
00000000
55555555
AAAAAAAA
FFFFFFFF
```

Constant Data Test

CO *p= st= sc b= d=*

The *Constant Data Test* option on the Memory Menu does a write-read-compare with a constant data pattern.

Walking One Data Test

WA *p= st= sc b=*

The *Walking One Data Test* option on the Memory Menu does a write-read-compare with a walking one-bit data pattern.

Incrementing Data Test

AD *p= st= sc b=*

The *Incrementing Data Test* option on the Memory Menu writes an incrementing data pattern from the beginning to the end of memory. It then reads and compares the data from the end to the beginning of memory.

Random Address and Data Test

RA *p= st= sc b=*

The *Random Address and Data Test* option on the Memory Menu writes a random data pattern to a random address. It then reads the pattern back and compares.

8- and 16-bit Data Size Access

DS *p= st= sc b=*

The *8- and 16-bit Data Size Access* option on the Memory Menu writes, then reads and compares a single 8-bit pattern at offset 1, and a 16-bit pattern at offset 2.

RAMDAC Address Register Test

AR *p= st= sc*

The *RAMDAC Address Register Test* option on the Memory Menu writes an incrementing pattern to the RAMDAC Address Register. The pattern is then read and compared to the write pattern.

RAMDAC Command Register Test

CR *p= st= sc*

The *RAMDAC Command Register Test* option on the Memory Menu writes an incrementing pattern to the RAMDAC Command and Mask Registers. The pattern is then read and compared to the write pattern.

RAMDAC Look-up Tables Test

LT *p= st= sc*

The *RAMDAC Look-up Tables Test* option on the Memory Menu writes an incrementing pattern to the RAMDAC Look-up Tables. The pattern is then read and compared to the write pattern.

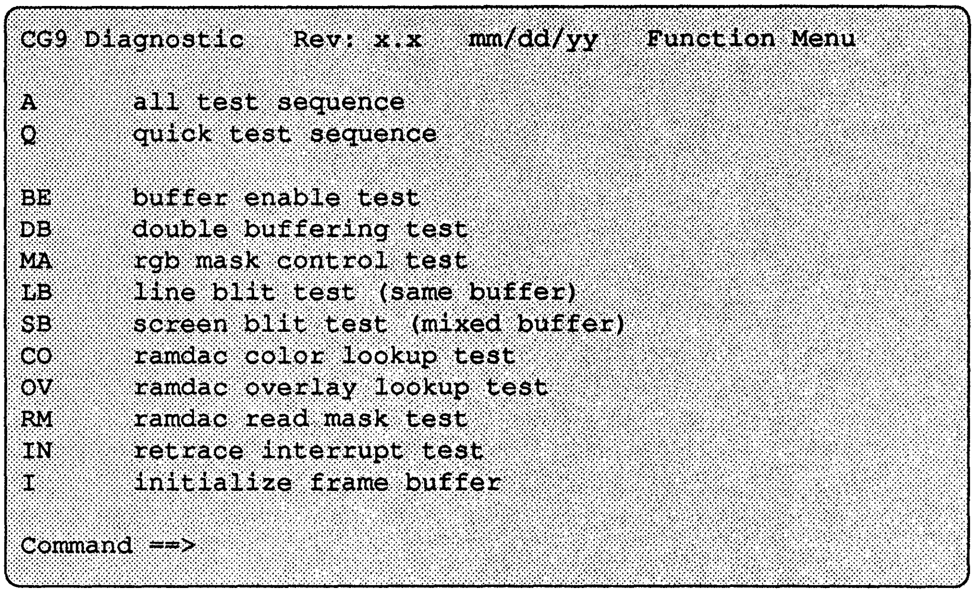
Initialize Frame Buffer

I

The *Initialize Frame Buffer* option on the Memory Menu initializes the CG9 frame buffer to its default state.

Function Menu

When you choose **FM** from the Main Menu, the Function Menu is displayed:



```
CG9 Diagnostic   Rev: x.x   mm/dd/yy   Function Menu

A       all test sequence
Q       quick test sequence

BE      buffer enable test
DB      double buffering test
MA      rgb mask control test
LB      line blit test (same buffer)
SB      screen blit test (mixed buffer)
CO      ramdac color lookup test
OV      ramdac overlay lookup test
RM      ramdac read mask test
IN      retrace interrupt test
I       initialize frame buffer

Command ==>
```

The options on this menu provide tests to verify all functional components of the CG9 frame buffer. The environmental variables and other parameters that you may specify on the command line are summarized in the following table:

Variable	*Description*	*Default*
`Pass`	Pass count.	1
`STop`	Stop on error count.	1
`SCope`	Scope loop on error initiated if this variable is entered on the command line.	No looping
`Data`	Hex data pattern.	FF8001

All Test Sequence

A

The *All Test Sequence* option on the Function Menu executes all of the tests on the menu in order, using the default parameter values. The Screen Blit Test (Mixed Buffer) option is executed twice.

Quick Test Sequence

Q

The *Quick Test Sequence* option on the Function Menu executes the following tests, using the default parameter values:

- Buffer Enable Test
- RGB Mask Control Test
- RAMDAC Color Lookup Test
- Line Blit Test (Same Buffer)
- Retrace Interrupt Test

Buffer Enable Test

BE *p= st= sc*

The *Buffer Enable Test* option on the Function Menu verifies double-buffer enable functionality. The first four locations of RGB memory are cleared, including bit 25 from the display mode plane. The buffer enable bit in the Double Buffer R/W Register is set to 1, enabling double buffering. The first four locations are then written to again, excluding bit 25. The buffer enable bit should set the 25th bit.

Double Buffering Test

DB *p= st= sc*

The *Double Buffering Test* option on the Function Menu verifies double-buffer functionality. First buffer A is filled with all zeros, then buffer B is filled with all ones. All 24 bits are then read back and verified. The process is repeated using all 5's in buffer A, and all A's in buffer B.

RGB Mask Control Test

MA *p= st= sc*

The *RGB Mask Control Test* option on the Function Menu verifies 12/24 bit plane masking in RGB memory. Write and read mask operations are tested in both 12- and 24-bit mode.

Line Blit Test (Same Buffer)

LB *p= st=*

The *Line Blit Test (Same Buffer)* option on the Function Menu performs the functional test for bit-block transfers that use one 1024-pixel line of an incrementing data pattern. The test is performed in 12-bit buffer A, 12-bit buffer B, and 24-bit modes. The Debug Menu enables you to perform a manual BITBLT of one horizontal line, or of one rectangle.

Screen Blit Test (Mixed Buffer)

SB *p= st= sc*

The *Screen Blit Test (Mixed Buffer)* option on the Function Menu performs the functional test for bit-block transfers. The RGB memory is sectioned into 12- and 24-bit quadrants. The display is configured as follows:

24	12
12	24

The upper half is filled with a pattern incremented by 0x030201. The lower half is filled with a pattern decremented by 0x030201. Next the test swaps the upper and lower halves using BITBLT. Data is then verified. The Debug Menu provides the capability to perform a manual BITBLT of one horizontal line, or of one rectangle.

RAMDAC Color Look-up Test

CO *p= st= sc d=*

The *RAMDAC Color Look-up Test* option on the Function Menu verifies color look-up table addressing from frame buffer memory. RGB memory is filled with a constant data pattern. The RAMDACs are placed in test mode, which loops back the indexed color value to be verified.

RAMDAC Overlay Look-up Test

OV *p= st= sc*

The *RAMDAC Overlay Look-up Test* option on the Function Menu verifies color look-up table addressing from overlay/enable memory. The RAMDACs are placed in test mode. All four possible colors (three overlay and one transparent) are then looped back.

RAMDAC Read Mask Test

RM *p= st= sc*

The *RAMDAC Read Mask Test* option on the Function Menu verifies color look-up table masking. The RAMDACs are placed in test mode. The RAMDAC Read Mask Register is then used to mask table look-up.

Retrace Interrupt Test

IN *p= st= sc*

The *Retrace Interrupt Test* option on the Function Menu verifies the vertical retrace interrupt.

Initialize Frame Buffer

I

The *Initialize Frame Buffer* option on the Function Menu initializes the CG9 frame buffer to its default state.

Pattern Menu

When you choose **PM** from the Main Menu, the Pattern Menu is displayed:

```
CG9 Diagnostic   Rev: x.x   mm/dd/yy   Pattern Menu

A      all test sequence

BU     rotate through 16 burst patterns
MO     display one overlay color
BX     16 gray level boxes
SH     alternating stripes with boxes
CR     cross hatch pattern
TV     TV color bar
CY     cycle through colors
TA     crosstalk color memory with overlay
GR     FCC dither pattern (gray stripes)
FR     one pixel wide border
I      initialize frame buffer

Command ==>
```

The options on this menu allow you to verify the pattern display capability of the CG9. All of the tests on this menu require visual verification. Optional parameters that you may enter on the command line are described together with their associated commands.

All Test Sequence

A

The *All Test Sequence* option on the Pattern Menu executes all of the tests on the menu in order.

Rotate Through 16 Burst Patterns

BU

The *Rotate Through 16 Burst Patterns* option on the Pattern Menu rotates through 16 colors of ramp patterns in RGB.

Display One Overlay Color

MO *r= g= b=*

The *Display One Overlay Color* option on the Pattern Menu displays one color on the screen to allow scope measurement of the RAMDAC. The parameters are summarized in the following table:

Parameter	*Description*
`Red`	Hex value (0–FF) specifying intensity of red.
`Green`	Hex value (0–FF) specifying intensity of green.
`Blue`	Hex value (0–FF) specifying intensity of blue.

16 Gray Level Boxes

BX

The *16 Gray Level Boxes* option on the Pattern Menu displays 16 different gray boxes. Gray levels start at 6% and increment in 6% steps to 100%.

Alternating Stripes With Boxes

SH

The *Alternating Stripes With Boxes* option on the Pattern Menu displays a shadow pattern. The background of the pattern is composed of alternating white and black vertical lines. The foreground includes one white box in the upper middle of the screen and one black box in the lower middle of the screen. This pattern is used primarily to verify ringing (the rippling effect) at the left and right edges of the two boxes. The pattern is displayed in overlay memory and enable memory only.

Cross Hatch Pattern

CR *x= y= w= v*

The *Cross Hatch Pattern* option on the Pattern Menu displays the crosshatch pattern, a series of evenly spaced vertical and horizontal lines. This pattern verifies correct data mapping between the memory and RAMDACs. The crosshatch pattern is displayed in overlay memory and enable memory only. The parameters are summarized in the following table:

Parameter	*Description*
`Xstart`	Starting pixel location for vertical lines.
`Ystart`	Starting pixel location for horizontal lines.
`Width`	Number of pixels between consecutive lines.
`Vertical`	Vertical lines only when parameter is present.

TV Color Bar

TV

The *TV Color Bar* option on the Pattern Menu displays the TV color pattern for color addressing. This pattern verifies RAMDAC synchronization by placing solid colors beside each other. If synchronization is lost, black gaps will appear between some color bars. This pattern uses color memory only.

Cycle Through Colors

CY

The *Cycle Through Colors* option on the Pattern Menu cycles through all primary and secondary colors. The RAMDACs are tested by this pattern, verifying that the DACs can change to produce the different colors.

Crosstalk Color Memory With Overlay

TA *h*

The *Crosstalk Color Memory With Overlay* option on the Pattern Menu displays the video crosstalk pattern. Wide stripes are placed into color memory while overlay memory and enable memory are filled with ones. The result should be a white screen without vertical abnormality. The parameter is described in this table:

Parameter	*Description*
`Half`	Vertical bars in the lower half of memory only when parameter is present.

FCC Dither Pattern (Gray Stripes)

GR

The *FCC Dither Pattern (Gray Stripes)* option on the Pattern Menu displays the pattern used in FCC testing. Alternating pixels are toggled to produce a maximum frequency. All memories are used for this test.

One Pixel Wide Border

FR

The *One Pixel Wide Border* option on the Pattern Menu displays a one-pixel-wide border around the screen. This display verifies video memory to RAMDAC mapping. The pattern uses overlay memory.

Initialize Frame Buffer

I

The *Initialize Frame Buffer* option on the Pattern Menu initializes the CG9 frame buffer to its default state.

Debug Menu

When you choose **DM** from the Main Menu, the Debug Menu is displayed:

```
CG9 Diagnostic   Rev: x.x   mm/dd/yy   Debug Menu

FI      fill frame buffer location(s)
ED      edit frame buffer location(s)
DU      dump block of frame buffer data
PA      display RAMDAC overlay or color palette
SC      scope frame buffer location
WR      write-quick-read scope test
BL      bit-blt horizontal line
BS      bit-blt line scope
BR      bit-blt rectangle
I       initialize frame buffer

Command ==>
```

The options on this menu allow you to perform operations for specific debugging purposes. The environmental variables and other parameters that you may specify on the command line are summarized in the following table:

Variable	*Description*	*Default*
`Block`	Type of memory block. Control space memory (0) Overlay memory (1) Enable memory (2) RGB memory (3)	3
`Offset`	Hex byte offset in memory from which operation starts.	0
`Count`	Hex number of long words to write or write and read.	Block size
`Data`	Hex data pattern.	0
`DM`	Writes also made to display mode memory when this parameter is present.	No writes to display mode memory

Other parameters that are specific to certain options are described together with those options.

Fill Frame Buffer Location(s)

`FI` *b= o= c= d= dm*

The *Fill Frame Buffer Location(s)* option on the Debug Menu fills a contiguous block of frame buffer memory.

Edit Frame Buffer Location(s)

ED *b= o= dm*

The *Edit Frame Buffer Location(s)* option on the Debug Menu allows any frame buffer location to be written and read. It displays the physical address followed by the long data word at that address. You can then type one of the following:

- New data to be written to that address,
- [RETURN] to see the next long word,
- [-] to see the previous long word, or
- [q] to quit.

Dump Block of Frame Buffer Data

DU *b= o=*

The *Dump Block of Frame Buffer Data* option on the Debug Menu dumps a block of frame buffer data to the screen. It then prompts for a new offset to dump. You can then type one of the following:

- The new offset,
- [RETURN] to dump the next contiguous block, or
- [q] to quit.

Display RAMDAC Overlay or Color Palette

PA *o*

The *Display RAMDAC Overlay or Color Palette* option on the Debug Menu displays the contents of the RAMDAC overlay or of the color palette RAMs. After 16 color palette entries have been displayed, the test asks if you want to continue. You can then type one of the following:

- [RETURN] to display more color palette information, or
- [q] to quit.

The parameter is described in this table:

Parameter	*Description*	*Default*
`Overlay`	Indicates overlay palette when parameter is present.	Color palette

Scope Frame Buffer Location

SC *b= o= d= w dm*

The *Scope Frame Buffer Location* option on the Debug Menu allows tight scope looping on any frame buffer location. Twenty back-to-back write or read cycles are run continuously. To stop the test, press any key. The parameter specific to this option is described in the following table:

Parameter	*Description*	*Default*
`Write`	Indicates write scope when parameter is present.	Read scope

Write-Quick-Read Scope Test

WR *b= o= c= d= dm*

The *Write-Quick-Read Scope Test* option on the Debug Menu performs a write-quick-read on any frame buffer location. The write is followed immediately by a read. This operation is performed in a continuous loop over the area of memory that you specify. To stop the test, press any key. The default data pattern for this test is 55555555.

Bit-blt Horizontal Line

BL *x1= y1= x2= y2= c= d*

The *Bit-blt Horizontal Line* option on the Debug Menu performs a BITBLT of a horizontal line as long as 1024 pixels. The parameters specific to this option are summarized in the following table:

Parameter	*Description*	*Default*
`X1`	Decimal horizontal screen coordinate of the source line's left end.	8
`Y1`	Decimal vertical screen coordinate of the source line's left end.	0
`X2`	Decimal horizontal screen coordinate of the destination line's left end.	0
`Y2`	Decimal vertical screen coordinate of the destination line's left end.	0
`Count`	Decimal number of pixels to move.	8
`Down`	Indicates count-down BITBLT when parameter is present.	Count-up

Bit-blt Line Scope

BS *x1= y1= x2= y2= c=*

The *Bit-blt Line Scope* option on the Debug Menu performs a scope loop on the horizontal line BITBLT function. To stop the test, press any key. The parameters specific to this option are summarized in the following table:

Parameter	*Description*	*Default*
`X1`	Decimal horizontal screen coordinate of the source line's left end.	8
`Y1`	Decimal vertical screen coordinate of the source line's left end.	0
`X2`	Decimal horizontal screen coordinate of the destination line's left end.	0
`Y2`	Decimal vertical screen coordinate of the destination line's left end.	0
`Count`	Decimal number of pixels to move.	8

Bit-blt Rectangle

BR *x1= y1= x2= y2= w= h= m d*

The *Bit-blt Rectangle* option on the Debug Menu performs a BITBLT of a rectangle. You can either create a rectangle for BITBLT or use the current contents of RGB memory. The parameters specific to this option are summarized in the following table:

Parameter	*Description*	*Default*
`X1`	Decimal horizontal coordinate of the source rectangle's upper-left corner.	60
`Y1`	Decimal vertical coordinate of the source rectangle's upper-left corner.	0
`X2`	Decimal horizontal coordinate of the destination rectangle's upper-left corner.	0
`Y2`	Decimal vertical coordinate of the destination rectangle's upper-left corner.	0
`Width`	Decimal number of pixels specifying rectangle's width.	8
`Height`	Decimal number of pixels specifying rectangle's height.	8
`Make`	Indicates to make rectangle when parameter is present.	Use contents of RGB memory
`Down`	Indicates count-down BITBLT when parameter is present.	Count-up

Initialize Frame Buffer

I

The *Initialize Frame Buffer* option on the Pattern Menu initializes the CG9 frame buffer to its default state.

14.8. Glossary

BITBLT BIT-BLock Transfer. The process of moving a block from one location on screen to another.

RAMDAC Random Access Memory Digital to Analog Converter.

15

Graphics Processor1 Diagnostic

15

Graphics Processor1 Diagnostic

15.1. Introduction

The Sun GP1 (Graphics Processor 1) and GB (Graphics Buffer Board) are full-sized VME boards intended for use in any Sun color workstation that has a VMEbus. They may also be used in a monochrome workstation that supports grayscale imaging and has a modified color frame buffer board for that purpose.

The GP1 and GB only work in certain adjacent slots of a Sun workstation. For details, see the appropriate Cardcage Backplane Configuration and Slot Assignment document.

15.2. What This Chapter Contains

This chapter describes the GP1 Diagnostic. Each diagnostic test menu is shown, followed by a description of each command. The end of the chapter contains a list of error messages and their interpretations.

15.3. Overview of the Diagnostic

The GP1 Diagnostic consists of a series of tests that run automatically with no user intervention, except the scope loop tests, which present submenus of options.

The diagnostic does not name failing parts of the board. It displays actual and expected values of each test. See the "Error Messages" section for tips on interpreting the values to locate hardware faults.

15.4. User Interface

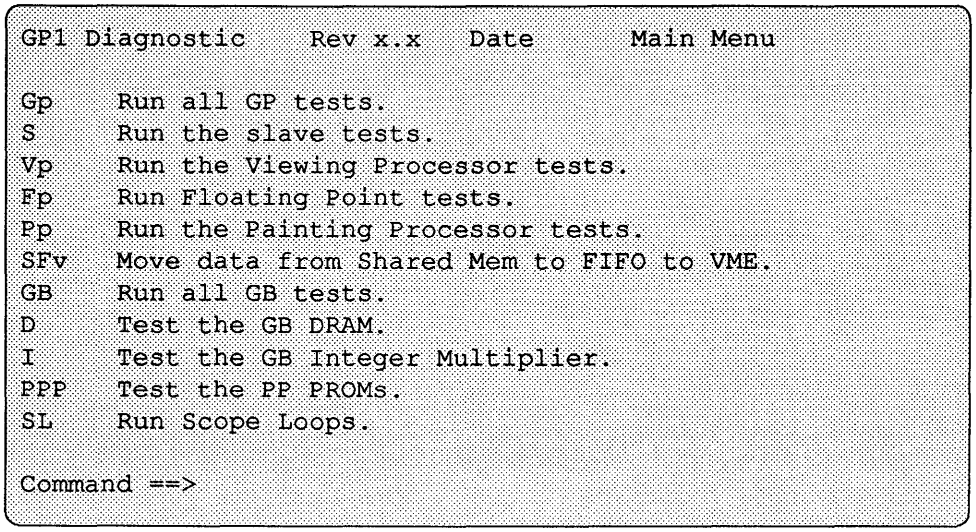
The user interface of the GP1 Diagnostic adheres to the standards of the Exec menu system. Each menu option may be selected by typing the letter or letters displayed in upper case in the column on the left side of the menu.

15.5. Starting the Diagnostic

For information on starting the Exec, refer to Chapter 2, "Using the SunDiagnostic Executive." After you have started the Exec, choose the GP1 Diagnostic from the Diagnostics Menu.

15.6. The Main Menu

The Main Menu provides the following choices. To choose an option, enter the letter(s) in upper case on the left side of the menu.



```
GP1 Diagnostic    Rev x.x   Date      Main Menu

Gp    Run all GP tests.
S     Run the 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.
PPP   Test the PP PROMs.
SL    Run Scope Loops.

Command ==>
```

The following selections operate on the Graphics Processor Board:

- Run All GP Tests
- Run the Slave Tests
- Run the Viewing Processor Tests
- Run Floating Point Tests
- Run the Painting Processor Tests
- Move Data from Shared Mem to FIFO to VME

The following selections operate on the Graphics Buffer Board:

- Run All GB Tests
- Test the GB DRAM
- Test the GB Integer Multiplier

If no Graphics Buffer Board is present, the test will query you about it then return to the Main Menu.

Selecting *Run All GP Tests* is equivalent to selecting *Run the Slave Tests*, *Run the Viewing Processor Tests*, *Run Floating Point Tests*, *Run the Painting Processor Tests*, and *Move Data from Shared Mem to FIFO to VME*.

Selecting *Run All GB Tests* is equivalent to selecting *Test the GB DRAM*and *Test the GB Integer Multiplier*.

Here is a brief description of each Main Menu choice:

- *Run All GP Tests* — Includes the following: Slave, VP, Floating Point, and PP.
- *Run the Slave Tests* — Slave tests are VME addressable parts of the GP: Board ID, Status and Control Registers, Shared Memory, and Microstore.
- *Run the Viewing Processor Tests* — Tests the VP Status Register, microword General and N Fields, N and Branch Registers, microcode Bank Select

Hardware, Shared Memory interface, Two Address operations, VP PROM, VP Control and the VP Interprocessor Flags.

- □ *Run Floating Point Tests* — Tests Floating Point Registers and Buffers, Flow and Pipeline modes.
- □ *Run the Painting Processor Tests* — Tests the PP Status Register, microword General and N Fields, N and Branch Registers, microcode Bank Select Hardware, Two Address operations, Scratchpad memory, PP Control and PP Interprocessor flags.
- □ *Move Data from Shared Mem to FIFO to VME* — VP moves data patterns from Shared Memory to the FIFO, then from FIFO to VME back to Shared Memory. It tests VME slave and master, both GP processors, and the FIFO and IP flags between them.
- □ *Run All GB Tests* — Tests the GB Interface hardware on the GP, GP DRAM, and the Integer Multiplier.
- □ *Test the GB DRAM* — Uses several methods to test the GB's Dynamic RAM.
- □ *Test the GB Integer Multiplier* — Tests the Integer Multiplier hardware on the GB.
- □ *Test the PP PROMs* — Tests the data and address path for the GB's PP PROM.
- □ *Run Scope Loops* — Provides a variety of scope loops.

Several of these options are discussed in more detail in the following sections.

Run the Slave Tests

Selecting s from the Main Menu tests access to the GP1 board identification register, the status register, the control register, shared memory, and the microcode storage RAM. These tests are described below.

- □ The *Board ID* test reads the ID register and displays an error message if the register contains a nonzero value.
- □ The *Status Register* test resets the control register then checks the status register to make sure that a board reset has occurred. The status register is expected to be all zeros. If this has not happened, an error message is displayed.
- □ The *Control Register* test toggles the writable bits of the control register and then checks the status register to see if certain functions are being properly enabled and disabled. In this case the interrupts on the GP are being enabled and disabled by setting the appropriate bits in the control register.
- □ The *Shared Memory* test exercises the Graphics Processor Board shared memory RAM. The types of tests used are write/read, word patterns, byte patterns, address uniqueness, and random.
- □ The *Write/Read* test writes a word-sized pattern to a shared memory RAM location and then reads it back. This process is repeated for each address location in RAM.

- □ The *Word Patterns* test writes a word pattern to all the memory locations and then reads it back for verification.
- □ The *Byte Patterns* test writes a byte pattern to each memory location and then reads it back for verification.
- □ The *Address Uniqueness* test writes a modified version of its address to each location and then reads it back for verification. The test writes to each location the ones complement of the modified address location.
- □ The *Random* test generates random numbers, writes them into the shared memory RAM, and then reads back the memory for verification.
- □ Access to the *Microstore* on the GP board is tested by using such methods as pattern sensitivity, address uniqueness, and random data.

Run the Viewing Processor Tests

Selecting **VP** from the Main Menu runs tests on the GP1 viewing processor. The individual tests are described below:

- □ The *Viewing Processor* status register test writes a pattern to the viewing processor and reading back for verification. If there is a mismatch, an error message is displayed
- □ The *IP Flag Register* test reads bit 8 and checks that it is always set to zero. If not, the appropriate error message is displayed.
- □ The *General Field 2* of the viewing processor is loaded with microcode and then verified for each location.
- □ The *General Field 1* of the viewing processor carries the 16-bit general field part of the microword from the microstore to the VP bus.
- □ The *N Register* test performs a series of write and read-back tests on the 4-bit N register on the viewing processor.
- □ The *N Register Field Assembly* test uses a pattern in the microcode instruction to make a binary calculation in the AM29116 and then reads it back for verification.
- □ To test the *Branch Register* for the 2910, a pattern is placed in the viewing processor branch register and then an attempt is made to branch to the register as an address. If the microcode went to a different address, an error is indicated.
- □ The *Microstore Bank Select* option tests the microstore bank logic. It attempts to load microcode into a high microstore location. If it cannot be verified during the read-back process, an error is displayed.
- □ The *Shared Memory* test performs a read/write test on the shared memory on the viewing processor.
- □ The two address operation for the AM29116 is tested through a series of write and read-back tests.
- □ The *Viewing Processor PROM* test checks the viewing processor reciprocal PROM look-up table for floating point operations. To do this, the test moves

the look-up table contents to shared memory and calculates a checksum for the data. The checksum is compared to the actual checksum for verification.

- □ The *Viewing Processor Continue* option uses a bit from the control register to start the viewing processor from a halted state without setting the program sequencer to microcode location zero.
- □ The *Viewing Processor FIFO* option tests the FIFO status bits in the IP flag register, the data path to and from the FIFO, the functionality of the FIFO as RAM and the FIFO full/empty condition code lines to the condition code selector for the 2910 sequencer.
- □ The *Viewing Processor CC Select* option tests the condition code select logic on the viewing processor. The test forces each condition code and then causes the microcode to branch to an error semaphore generator if the error condition is not true. It then generates one or more other condition codes (but not the one under test) and causes the microcode to branch to an error semaphore generator if the condition code under test is true.

Run Floating Point Tests

This option runs tests on the floating point circuitry. The tests are described in the following sections.

- □ To test the *Floating Point Register Sets A and B*, the diagnostic writes constant and address unique patterns to them and then reads them back for data comparison. The register sets are also tested with walking ones.
- □ To test *Weitek chip set* function, the Weitek ALU converts the fixed point data to floating point data and back again.
- □ To test the *Pipeline Mode and the Result Register*, an operation is set up using A and B operands. The operation is (A+B)*B.
- □ To test *Matrix Multiplication*, 1x4 and a 4x1 matrices are multiplied. This test is intended to provide a more thorough exercise of the pipeline mode.
- □ To test the *Floating Point Status Register*, various write operations are performed and then read back for verification.

Run the Painting Processor Tests

This option tests the GP1 Painting Processor. The tests are described in the following sections. The parts and functions that are tested are the painting processor status register; the IP flag bit; FIFO, General Field 1 and 2; IP flags; N register; branch register; microstore bank select; scratchpad memory; continue mode; VME interrupts; VME bus master write and read access; the VME status register; assembly contents; VME interrupt completion; two address operation, and the three-way branch operation.

- □ Carefully selected patterns are written to the painting processor status register and read back for verification.
- □ Bit 8 of the painting processor IP flag register is read to make sure that it is always set to 1.
- □ To test the painting processor side of the FIFO, the interprocessor flags register is read, the FIFO status bit is checked for "not empty" and then for

"empty" after a control register reset, and patterns are moved from the FIFO to the painting processor to assure that the data path is working.

- □ The *General Field 1 and 2* tests are the same as those for the viewing processor.
- □ The *IP Flags* test moves patterns from the painting processor to IP flag register 2 and then to shared memory for comparison. IP flag register 1 is tested in the same manner.
- □ The *4-bit N Register* test under the painting processor option is the same as that in the viewing processor option.
- □ The *Branch Register* test for the painting processor is identical to the one for the viewing processor.
- □ The *Microstore Bank Select* test for the painting processor is identical to that for the viewing processor.
- □ To test the *Scratchpad Memory* on the painting processor, both constant and address unique patterns are loaded and then read to shared memory through the FIFO for comparison. Sixteen walking ones patterns are used to check the data paths.
- □ The *Painting Processor Continue* test uses a bit from the control register to start the viewing processor from a halted state without setting the program sequencer to location zero of the microcode.
- □ The *GP1 Interrupts* are disabled and then an attempt is made to generate one using microcode. The GP1 status register and the VME status register are then checked for an interrupt pending signal.
- □ The capability of the painting processor as a VME bus master is tested by writing patterns over the bus into shared memory. The data is then compared for any errors.
- □ The painting processor also reads data from shared memory over the VME bus to test its reading capability as a VME bus master.
- □ The *Illegal Access* (bit 4) and *timeout* (bit 2) bits of the VME status register are tested by passing the contents from the VME status register to the viewing processor FIFO and then read back.
- □ The *Painting Processor Assembly* test uses a pattern in the microcode instruction to make a binary calculation in the AM29116 and then reads it back for verification.
- □ A *VME Interrupt Completion* test enables GP1 interrupts and then causes the microcode to try and generate one. Because an interrupt is expected, an interrupt handler is set up.
- □ The *Two Address Operation* test for the painting processor is identical to that of the viewing processor.
- □ Three possible addresses are set up to test the painting processor's ability to perform a 3-way branch.

Test the GB DRAM

This option runs a series of tests on the Graphics Buffer DRAM. The tests are described in the following sections.

- The *Address Uniqueness* test uses address unique patterns to test the GB DRAM. The DRAM is first filled with the value of the pattern that is being incremented each time with each different address. The data is then read back and compared in shared memory.
- The *Constant Pattern* test writes twelve different patterns to the DRAM and reads back each pattern for comparison.
- The test writes the value 0x0000 to to the even addresses and 0xffff to the odd addresses and then delays testing for a period of time, in order to test the DRAM refresh logic. Each address is then read back for comparison and verification. Finally, the test is repeated, with 0x0000 written to the odd addresses and 0xffff is written to the even addresses.
- The *Galloping Pattern* test writes a constant pattern to every third location and then compares in shared memory.
- The *Fill Mode* test fills the 4,000-word block of the GB DRAM with constant patterns and then reads them in shared memory for comparison.
- The *RMW* test writes constant patterns to the first 4,000-word block and then reads them back in shared memory for comparison. Both tasks are done in RMW (Read/Modify/Write) mode.

Test the GB Integer Multiplier

This option tests the Integer Multiplier on the Graphics Buffer board. Multiplication operations use shifting ones for the X and Y operands to test the functionality of the AM29517.

Test the PP PROMs

This option tests the data and address path for the Graphic Buffer PP PROM. It tests the address and data lines to the PROMs on the Graphics Buffer Board. The test moves the microcode from the PROMs through the VMEbus to the shared memory. There it is "checksummed," and all failures reported.

Run Scope Loops

Selecting this option 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 CTRL-C.

15.7. The Scope Loop Menus

The *Run Scope Loops* selection from the Main Menu displays 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 Loops
P      PP and GB All Source and Destination Loops
D      Graphics Buffer DRAM Scope Loops

Command ==>
```

Each of the choices shown above displays a submenu, described on the pages that follow, along with the submenu options.

S — Shared Memory

The *Shared memory from VME Scope Loops* selection from the Scope Loop Menu displays the following submenu of test choices:

```
GP1 Diagnostic  Rev x.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 *Fixed Pattern to 1 Location*, *Fixed Pattern to All Locations*, and *Read 1 Location* selections from the Shmem Scope Loop Menu 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 *Incrementing Pattern to Incr Loc* command writes incrementing patterns to incrementing locations.

The *Read All Locations* command reads every shared memory location.

M — Micstor Scope Loop

The *Microstore from VME Scope Loops* selection from the Scope Loop Menu displays 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 *Fixed Pattern to Microword Location*, *Fixed Pattern to All Microword Locations*, and *Read 1 Microword Location* 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 press [RETURN] without entering values, the default used is 0.

The *Incrementing Pattern to Incr Locs* choice writes incrementing patterns to incrementing microstore locations.

The *Read All Microword Locations* selection reads all microstore locations.

A — Microstore Address Register

Selecting *A* from the Scope Loop Menu displays the following submenu:

```
P     Write a pattern
R     Write a pattern then read forever
WTI   Write a pattern then increment
```

P If you select *Write a Pattern* from the Microstore Address Register test submenu, the program will prompt you for a pattern:

```
Please give a pattern to write (4 hex digits):
```

If you enter [RETURN], a default 0x0000 pattern is used, and then the test displays this message:

```
NOW WRITING 0000 TO MICROSTORE ADDRESS REGISTER
```

R If you select *Write a Pattern Then Read Forever* from the Microstore Address Register test submenu, the program prompts you for a pattern as shown above, and then displays this message:

```
NOW READING MICROSTORE ADDRESS REGISTER, EXPECT pattern you entered
```

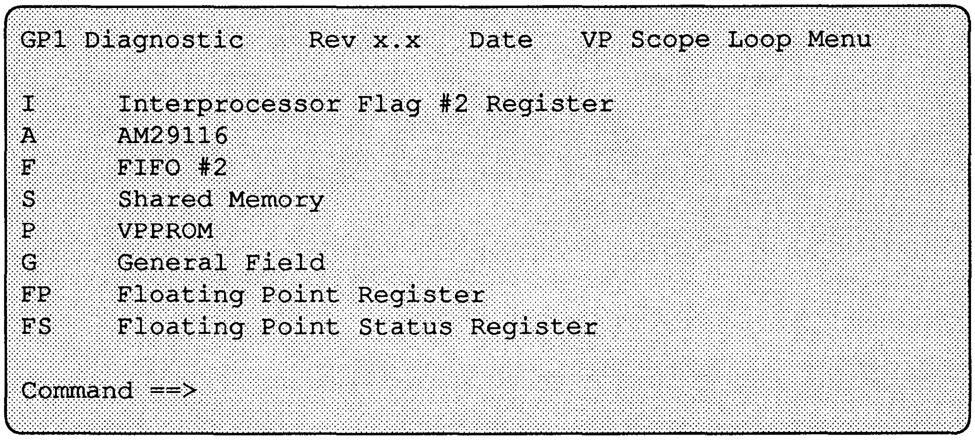
WTI

If you select *Write a Pattern Then Implement* from the Microstore Address Register test submenu, you are prompted for a pattern and the test displays this message:

```
USING pattern
INITIALIZED MICROSTORE ADDRESS REG WITH pattern
NOW INCREMENTING IT BY READING DATA REG
```

V — VP Scope Loop Menu

If you select *VP All Source and Destination Loops* from the Scope Loop Menu, this submenu 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
FP     Floating Point Register
FS     Floating Point Status Register

Command ==>
```

These menu choices, except for the *Interprocessor Flag #2 Register* 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 *AM29116*, the program asks you to select a pattern after you choose one of the following 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

   Command ==>
```

If, for example, you chose *Status LED Register* from the VP29116 Loop Menu, the program would prompt you to enter a pattern and then display this message:

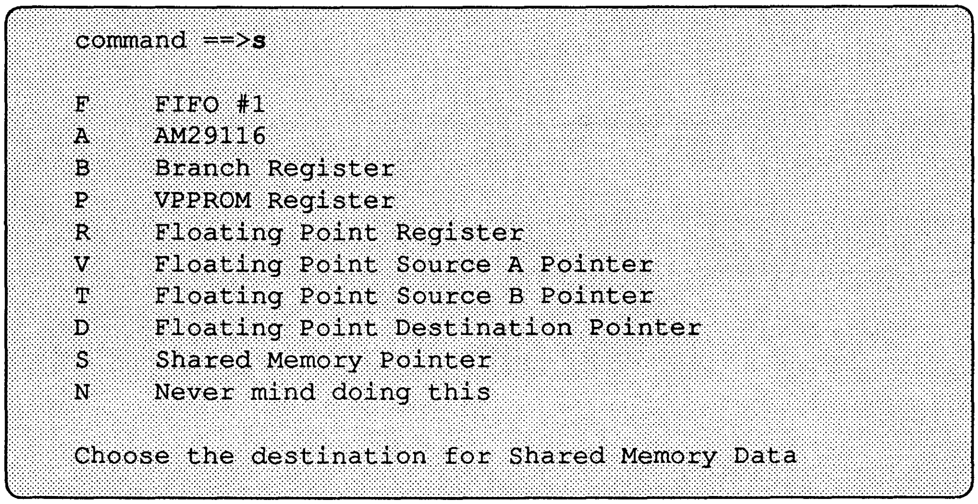
MOVING AM29116 (*pattern*) TO STATUS LED REGISTER

F If you select *FIFO #2* from the VP Scope 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 any of the first four options, the program prompts you to enter a pattern and then moves that pattern from the FIFO#2 to your destination choice. If you select N, you are returned to the VP Scope Loop Menu.

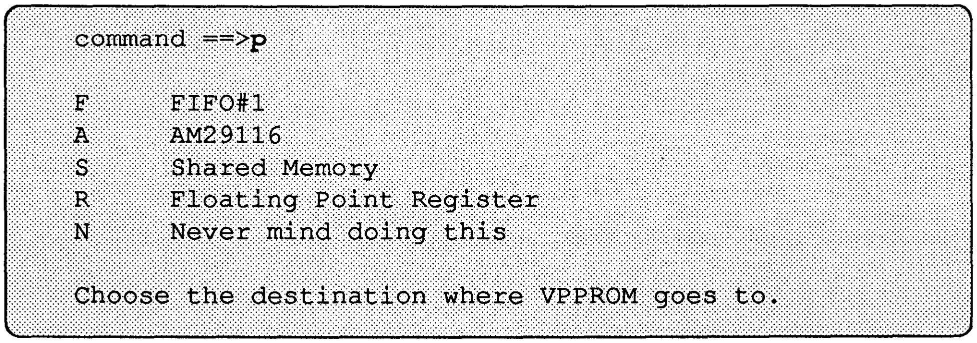
s If you select *Shared Memory* from the VP Scope Loop Menu, this selection of destinations for the Shared Memory data is displayed:



```
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 Pointer
S    Shared Memory Pointer
N    Never mind doing this

Choose the destination for Shared Memory Data
```

Any selection, except `N`, moves Shared Memory Data to the destination represented by the selection. `N` returns you to the VP Scope Loop Menu.

P Selecting *VPPROM* from the VP Scope Loop Menu displays 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 VP Scope Loop Menu; all the other selections move the VP PROM to the destination represented by that choice.

G If you select *General Field* from the VP Scope Loop Menu, this selection of destinations for the General Field data is displayed:

```
command ==>g

R    Interprocessor Flag #1 Register
F    FIFO #1
A    AM29116
B    Branch Register
P    VPPROM Pointer
S    Shared Memory
T    Floating Point Register
G    Floating Point Source A Pointer
U    Floating Point Source B Pointer
D    Floating Point Destination Pointer
M    Shared Memory Pointer

Choose the destination for General Field data:
```

Any selection moves General Field data to the destination represented by the selection.

FP If you select *Floating Point Register* from the VP Scope Loop Menu, this selection of destinations for the Floating Point Register data is displayed:

```
command ==>fp

1    FIFO #1
2    AM29116
3    Branch Register
4    VPPROM Pointer
5    Shared Memory
6    Floating Point Register
7    Floating Point Source A Pointer
8    Floating Point Source B Pointer
9    Floating Point Destination Pointer
A    Shared Memory Pointer
N    Never mind doing this

Choose the destination for FP Register data:
```

Any selection moves Floating Point Register data to the destination represented by the selection. `N` returns you to the VP Scope Loop Menu.

FS A *Floating Point Status Register* selection from the VP Scope Loop Menu displays 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 *PP and GB All Source and Destination Loops* 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
SP    Scratchpad Memory
F     FIFO #1
M     Multiplier Result
```

All the selections shown above bring up submenus, with the exception of the first choice, *Interprocessor Flag#1 Register*, which requires no user input. The submenus 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 submenu:

```
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      Multiplier Mode Register
ID      Interrupt ID Register
VH      VME High Address Register
VL      VME Low Address Register
VC      VME Control Register
GH      GB High Address Pointer
GL      GB Low Address Pointer

Command ==>
```

P Selecting *PPPROM* from the PP Scope Loop Menu displays this submenu:

```
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 Selecting *GB Read Data Register* from the PP Scope Loop Menu displays this submenu:

```
A       AM29116
H       VME High Address Register
L       VME Low Address Register
B       Branch Register
S       Scratchpad Pointer
P       PPPROM Pointer
N       Never mind doing this

Choose the destination for GB Read Data Register:
```

V Selecting *VME Read Data Register* from the PP Scope Loop Menu displays this submenu:

```
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 *VME Status Register* from the PP Scope Loop Menu displays this submenu:

```
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 *General Field* from the PP Scope Loop Menu displays this submenu:

```
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.

SP Selecting *Scratchpad Memory* from the PP Scope Loop Menu displays this submenu:

```
SP

1       FIFO #2
2       AM29116
3       GB Write Data Register
4       VME Write Data Register
5       Multiplier x Operand
6       Multiplier y Operand
7       Multiplier Mode Operand
8       Interrupt ID Register
9       VME High Address Register
10      VME Low Address Register
11      Branch Register
12      Scratchpad Pointer
13      GB High Address Pointer
14      GB Low Address Pointer

Choose destination for Scratchpad Memory data:
```

F Selecting *FIFO #1* from the PP Scope Loop Menu displays this submenu:

```
F

A       AM29116
H       VME High Address Register
L       VME Low Address Register
B       Branch Register
S       Scratchpad Pointer
P       PPPROM Pointer
N       Never mind doing this

Choose destination for FIFO #1 data:
```

If you select `N`, you will be returned to the PP Scope Loop Menu.

M When you select *Multiplier Result* from the PP Scope Loop menu, this submenu 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 *Graphics Buffer DRAM Scope Loops* from the Scope Loop Menu, this submenu 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 *Write to One Word Address* 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 If you select *Write to One Word Address, then Read It Back Forever* from the DRAM Scope Loop Menu, the program prompts you to select a location for the write scope loop. The valid hexadecimal range is 0–FFFFF (do not use a 0x prefix).

WA If you select *Write Address-unique Patterns to 4K-word Block* from the DRAM Scope Loop Menu, you are prompted for a location (as shown above), and, after you enter one, the test displays this message:

```
NOW WRITING ADDRESS UNIQUE COUNTING PATTERNS,
STARTING FROM 0 TO 4-K WORD BLOCK
STARTING ADDR HIGH=hex value LOW=hex value IN DRAM
```

R If you select *Read One Word Address* from the DRAM Scope Loop Menu, the program prompts you to select a location for the read scope loop. The valid hexadecimal range is 0–FFFFF (do not use a 0x prefix).

15.8. Error Messages

Messages from the GP1 Diagnostic begin with a number, which helps you to locate them in this text. Where the message examples show a value with a percent sign, it indicates the way 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 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.

The error messages are listed first in this section, together with the probable cause of the error.

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 nonzero 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 used a write-only Control Register to try to change the value of this bit to zero. Because 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 used the write-only Control Register to toggle this bit. Because 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 used the write-only Control Register to change this bit. Because 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 used the write-only Control Register to change this bit. Because 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 used the write-only Control Register to change this bit. Because 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 used the write-only Control Register to change this bit. Because 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.

During set-up, the microstore banks select test tried to load some microcode into a high microstore location, but couldn't verify it. That could mean that your GP1 has a very small microstore. This message is unlikely to show up.

57 VP Bank Select and address buffer to 2nd bank error found.
Couldn't jump to second bank (µstore 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 (µstore 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 as well as the path between it and the microstore hardware.

67 GP microstore address unique test error found.
µstore 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.
µstore 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 VMEbus.

69 GP microstore pattern test error found.
µstore 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 VMEbus.

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 match. 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 µ-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, the P2 bus is not connected across all adjacent slot pairs on the backplane. Consult the *Cardcage Slot Assignment and Backplane Configuration* document appropriate for this system. 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 that they did not match. 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 that they did not match. 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 that they did not match. 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 that they did not match. 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 that they did not match. 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%x EXP = 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 that 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 µ_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 (Mstore 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 (µstore 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 VMEbus, 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 backplane. They communicate using the P2 connectors. However, the P2 bus does not connect across all backplane slots. Refer to the *Cardcage Slot Assignment and Backplane Configuration* document. 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.

320 PP PROM Checksum Test error found
OBS = *0x(hex)* **EXP =** *0x(hex)*

The GB PP PROM test has failed. The checksum computed is indicated by OBS, the value expected is indicated by EXP.

15.9. 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.

16

Graphics Processor2 Diagnostic

16

Graphics Processor2 Diagnostic

16.1. Introduction to the GP2

The Graphics Processor 2 (GP2) offers increased graphics accelerator performance, and larger code and data memories than its predecessor. The GP2 supports firmware that can be downloaded. The GP2 is a three-stage "pipelined" graphic machine.

The first stage involves a transforming processor (Accel 8032) that performs 3-D matrix transforms, 3-D clipping, lighting model calculations, and 3-D to 2-D projection.
It supports 32-bit floating and integer data.

The second stage involves a Rendering Processor (Accel 8000) that performs the 2-D window clipping, Bresenham setup for vectors, and edge sorting for polygons. It supports 32-bit integer data.

The third stage involves a Pixel Processor (bit-slice machine - 39C10/49C402) that performs shading and tiling for triangles and polygon scan-line spans. It is responsible for sending data to the CG5 Frame Buffer.

16.2. What This Chapter Contains

This chapter first gives you an overview of the diagnostic. The Main Menu and each sub-menu are discussed, and menu selections and optional arguments are described. More detailed information on the tests, as well as error message and protocol information, is presented at the end of the chapter.

16.3. Diagnostic Objectives

The GP2 Diagnostic ensures that the GP2 board works correctly. The GP2 Diagnostic covers these functional areas of the GP2 Board:

□ Shared Memory

Shared memory is 512K x 32 bits wide and can be accessed by the CPU through the VME bus, the XP (Transform Processor), or the RP (Rendering Processor).

□ Writable Control Store (WCS) Memory

There is a 64K x 64-bits-wide memory for XP, a 32K x 32-bits-wide memory for RP, and 4K x 64-bits-wide memory for PP (Pixel Processor). All of these memories can be accessed by the CPU through the VME interface.

- Transform Processor (XP)

  The XP contains a 7136 Sequencer, a 7137 Integer Unit, and a 3132 Floating-Point Unit. It also contains 16K x 32 bits wide local SRAM (Shared Random Access Memory) that keeps the current context data and some housekeeping data. The SRAM can be accessed only by the XP. The data is sent to the RP through a 512 x 32 bits wide FIFO (first-in, first-out). The XP reads the data from the RP through a read-back register. The XP can access the shared memory.

- Rendering Processor (RP)

  The RP contains a 7136 Sequencer and 7137 Integer Unit. It also contains 16K x 32 bits wide local SRAM that keeps the data. It sends the data to the PP through a 512 x 32 bits wide FIFO. The RP reads the data from the PP through a read-back register. The RP can access the shared memory.

- Pixel Processor (PP)

  The PP is a micro-coded (64 bits wide) bit-slice machine. It contains a 39C10C micro-sequencer and two 49C402 Arithmetic Logic Units. There are several registers that send the data to the frame buffer by way of the P2 bus. The PP has a 1M x 16 bits- wide memory for the z- (depth) buffer that compares the *z* values. The PP can send the data back to the RP, or to the frame buffer over the P2 bus.

16.4. Hardware Requirements

In order to use the GP2 Diagnostic, the system being tested must have:

1. A Sun CPU Board.
2. A GP2 Board.
3. A CG5 or CG9 Board.
4. A Color Monitor.
5. A Working Keyboard.
6. A boot device (local disk, local tape, or remote disk over Ethernet).

16.5. Diagnostic Overview

The GP2 Diagnostic provides individual tests for each functional section of the GP2 board. Commands are also provided that enable you to perform the tests in a continuous sequence.

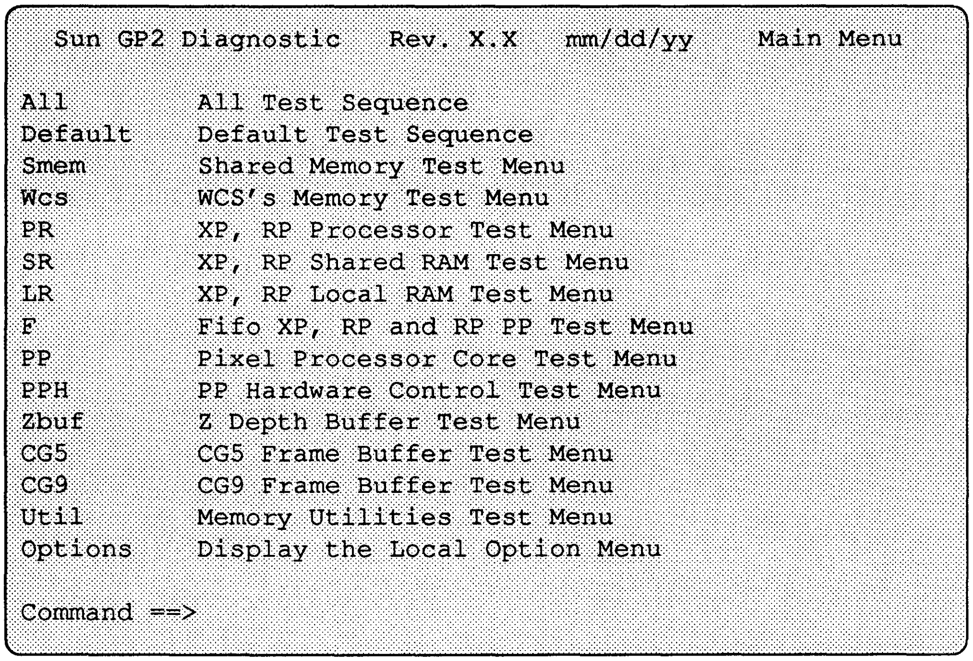
The user interface is menu driven: you select tests from a menu. Each test has several parameters or options that can be selected from the command line. For example, the options for the memory tests are as follows:

- pass count;
- memory to test (shared memory, XP, RP, PP WCS's); starting address;
- size of memory to test;

□ pattern to use. The user interface of the GP2 consists of a Main Menu and sub-menus. A help option describes command syntax. You can return to the current submenu or the Main Menu from the sub-menus.

16.6. The Main Menu

The Main Menu has fifteen 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 below.



```
  Sun GP2 Diagnostic   Rev. X.X   mm/dd/yy    Main Menu

All        All Test Sequence
Default    Default Test Sequence
Smem       Shared Memory Test Menu
Wcs        WCS's Memory Test Menu
PR         XP, RP Processor Test Menu
SR         XP, RP Shared RAM Test Menu
LR         XP, RP Local RAM Test Menu
F          Fifo XP, RP and RP PP Test Menu
PP         Pixel Processor Core Test Menu
PPH        PP Hardware Control Test Menu
Zbuf       Z Depth Buffer Test Menu
CG5        CG5 Frame Buffer Test Menu
CG9        CG9 Frame Buffer Test Menu
Util       Memory Utilities Test Menu
Options    Display the Local Option Menu

Command ==>
```

NOTE *The All (***A***) and Default (***D***) commands automatically determine which type of frame buffer is installed (CG5 or CG9) and run the appropriate tests.*

A *pass=*

Option **A** on the Main Menu is for the *All Test Sequence.* This option executes all of the tests available for this diagnostic. These tests are described below.

The `pass=` argument specifies the number of times the test should execute before it returns to the Main Menu.

D *pass=*

Option **D** from the Main Menu executes a default test sequence. These tests provide a quick confidence level check of the GP2 board to verify that it functions properly. These tests are discussed in the sections that describe each menu. The Default Test Sequence section specifies all of the tests executed for the global Default test sequence.

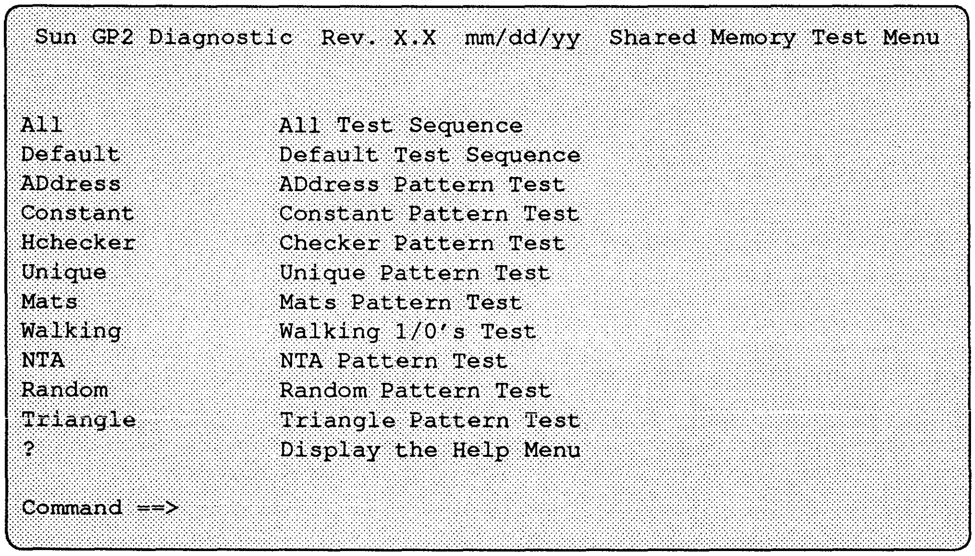
The `pass=` argument specifies the number of times the test should execute before it returns to the Main Menu.

When you select one of the other options from the Main Menu, the sub-menu containing the various tests for that sub-menu is displayed (see the chart below). The tests for these sub-menus are discussed in each sub-menu section.

Option	*Sub-menu*	*Argument*
S	Memory Tests	None
W	WCS Tests	None
PR	XP, RP Processor	None
SR	Shared RAM Tests	None
LR	Local RAM Tests	None
F	Fifo Tests	None
PP	Pixel Processor	None
PPH	Pixel Hardware	None
Z	Z Buffer Tests	None
CG5	CG5 Frame Buffer Test	None
CG9	CG9 Frame Buffer Test	None
U	Utility Menu	None
O	Option Menu	None

Shared Memory Tests Menu

Selecting `s` from the Main Menu displays the Shared Memory Test Menu options:



```
 Sun GP2 Diagnostic  Rev. X.X  mm/dd/yy  Shared Memory Test Menu


All                  All Test Sequence
Default              Default Test Sequence
ADdress              ADdress Pattern Test
Constant             Constant Pattern Test
Hchecker             Checker Pattern Test
Unique               Unique Pattern Test
Mats                 Mats Pattern Test
Walking              Walking 1/0's Test
NTA                  NTA Pattern Test
Random               Random Pattern Test
Triangle             Triangle Pattern Test
?                    Display the Help Menu

Command ==>
```

A *pass=*

The **All** option executes all of the tests in the Shared Memory Test Menu using their default parameter setting as described in *Shared Memory Tests*.

The *pass=* argument specifies the number of times the test should execute before it returns to the Main Menu.

D *pass=*

The **Default** option executes the default test sequence for the Shared Memory Test Menu. The Address, Mats, Walking 1's, and Random tests are executed using their default parameters.

The *pass=* argument specifies the number of times the test should execute before it returns to the Main Menu.

Shared Memory Tests

Selecting **S** from the Main Menu brings up a menu that lists these tests:

Addressing Test
AD *pass= offset= size= compl=[0,1] dmode=[0,1,2]*

Constant Pattern Test
C *pass= offset= size= pattern= dmode=[0,1,2]*

Checker Pattern Test
H *pass= offset= size= pattern= dmode=[0,1,2]*

Uniqueness Pattern Test
U *pass= offset= size= incr= dmode=[0,1,2]*

Mats (modulo 3's) Pattern Test
M *pass= offset= size= pattern= dmode=[0,1,2]*

Walking 1/0's Pattern Test
W *pass= offset= size= compl=[0,1] dmode=[0,1,2]*

NTA Pattern Test
NTA *pass= offset= size=*

Random Pattern Test
R *pass= offset= size= seed= dmode=[0,1,2]*

Triangle Pattern Test
T *pass= offset= size= dmode=[0,1,2]*

These memory tests perform a block write, then read each location, using VME accesses to the shared RAM. These tests are described in detail under the Memory Test Descriptions section.

The table below describes the default values and acceptable ranges for each of the command arguments (parameters).

Argument	*Default*	*Range*	*Description*
pass	1	0-2147483657	pass count.
offset	0x0000	0-0x3fc00	address offset from starting virtual address.
size	0x3fc00	0-0x3fc00	size of SRAM to test.
compl	2	0-2	0=no invert, 1=invert, 2=do both.
pattern	0x55555555	0-0xffffffff	pattern used in test.
incr	3	0-0xffffffff	incr added to previous pattern.
seed	5	0-0x7fffffff	seed for random pattern.
dmode	2	0-2	0=byte, 1=word, 2=long word addressing.

where:

The `pass, offset and size` arguments are self-explanatory.

The `compl` argument determines whether or not to invert the data pattern.

The `incr` argument specifies the increment count for the Uniqueness test.

The value of `incr` is added to the previous pattern each time.

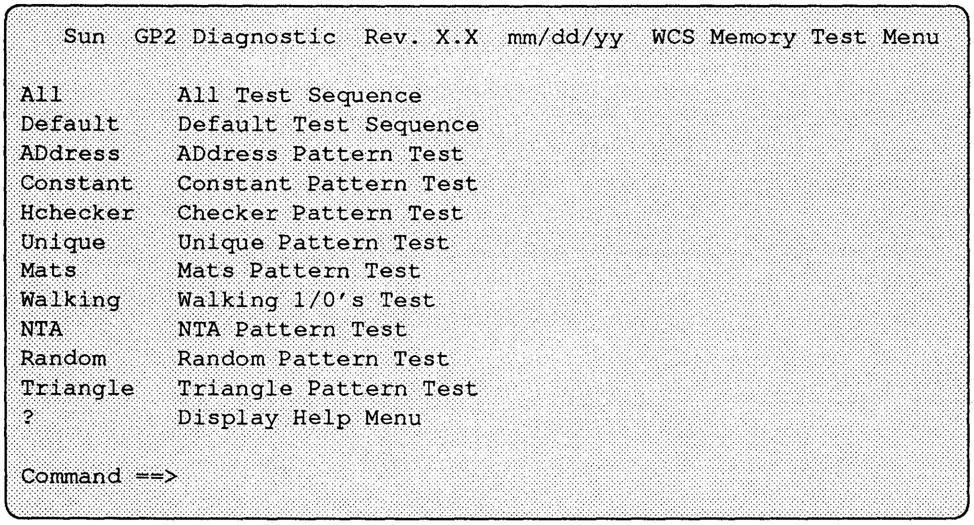
The `seed` argument specifies the seed number sent to the random number generator to change the pattern written to the Shared RAM. The default is 5.

The `dmode` argument specifies byte, word, or long word addressing, byte = 0, word = 1, and long word = 2.

? Entering **?** displays the help menu.

WCS Memory Tests Menu

The WCS Memory Test Menu has these options:



```
    Sun  GP2 Diagnostic  Rev. X.X  mm/dd/yy  WCS Memory Test Menu

All          All Test Sequence
Default      Default Test Sequence
ADdress      ADdress Pattern Test
Constant     Constant Pattern Test
Hchecker     Checker Pattern Test
Unique       Unique Pattern Test
Mats         Mats Pattern Test
Walking      Walking 1/0's Test
NTA          NTA Pattern Test
Random       Random Pattern Test
Triangle     Triangle Pattern Test
?            Display Help Menu

Command ==>
```

The test descriptions are found later in this chapter.

A *pass=*

The *All Test Sequence* executes all of the tests on the WCS Memory Test Menu using their default settings.

The `pass` argument specifies the number of times the test should execute before it returns to the Menu.

D *pass=*

The *Default Test Sequence* executes the default test sequence for the WCS Memory Test Menu. These tests are executed using their default arguments: Address test, Mats test, Walking 1's test, and Random test. The `pass=` argument specifies the number of times the test should execute before it returns to the Menu.

AD

Entering **AD** selects and executes the *Address Pattern Test.*

C

Entering **C** selects and executes the *Constant Pattern Test.*

H

Entering **H** selects and executes the *Checker Pattern Test.*

U

Entering **U** selects and executes the *Unique Pattern Test.*

M

Entering **M** selects and executes the *Mats Pattern Test.*

W

Entering **W** selects and executes the *Walking Ones Pattern Test.*

NTA

Entering **NTA** selects and executes the *NTA Pattern Test.*

R

Entering **NTR** selects and executes the *Random Pattern Test.*

T

Entering **T** selects and executes the *Triangle Pattern Test.*

? Option **?** displays the help menu.

The memory tests used for the XP WCS, RP WCS, and the PP WCS memory differ from the Shared RAM tests in that they are written completely in ''C''. The WCS can be accessed only through an address and a data register over the VME bus interface. The location is accessed by writing the location address into the register and reading the data from data register (the address register acts as a pointer to the location). The address test is important for ensuring that the data is placed in the right WCS memory location.

These tests operate from one menu. To select the desired WCS, an option for each test has been added. These options are as follows:

WCS=1 stands for XP WCS
WCS=2 stands for RP WCS
WCS=3 stands for PP WCS.

For example, to test the RP WCS, type **wcs=2** as part of the command line. The tests listed below are described in detail under the Memory Test Descriptions section.

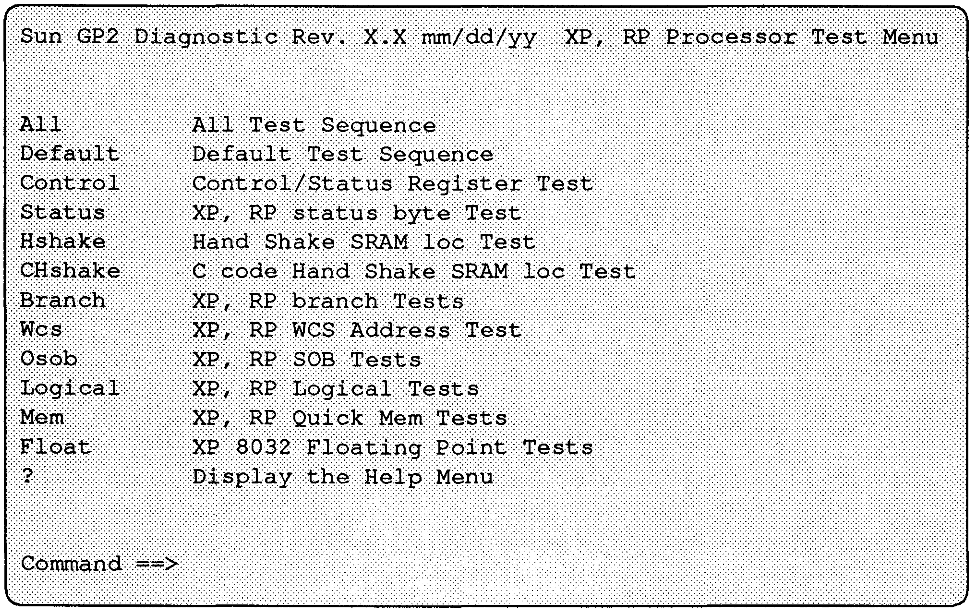
Test	Syntax
Addressing test	ad pass= wcs=[1,2,3] compl=[0,1]
Constant pattern test	c pass= wcs=[1,2,3] pattern=
Checker pattern test	h pass= wcs=[1,2,3] pattern=
Uniqueness pattern test	u pass= wcs=[1,2,3] incr=
Mats (modulo 3's) pattern test	m pass= wcs=[1,2,3] pattern=
Walking 1/0's Pattern test	w pass= wcs=[1,2,3] compl=[0,1]
NTA pattern test	nta pass= wcs=[1,2,3]
Random pattern test	r pass= wcs=[1,2,3] seed=
Triangle Pattern test	t pass= wcs=[1,2,3]

The following table describes the default values and acceptable range for each of these test parameters.

Option	*Default*	*Range*	*Description*
`pass`	1	0-2147483657	pass count.
`wcs`	1	1-3	1=XP, 2=RP, 3=PP WCS to test
`compl`	2	0-2	0=no invert, 1=invert, 2=do both.
`pattern`	0x55555555	0-0xffffffff	pattern used to test with.
`incr`	3	0-0xffffffff	incr added to previous pattern.
`seed`	5	0-0x7fffffff	seed for random pattern.

XP, RP Processor Tests Menu

An example of the XP,RP Processor Test Menu:



```
Sun GP2 Diagnostic Rev. X.X mm/dd/yy  XP, RP Processor Test Menu


All         All Test Sequence
Default     Default Test Sequence
Control     Control/Status Register Test
Status      XP, RP status byte Test
Hshake      Hand Shake SRAM loc Test
CHshake     C code Hand Shake SRAM loc Test
Branch      XP, RP branch Tests
Wcs         XP, RP WCS Address Test
Osob        XP, RP SOB Tests
Logical     XP, RP Logical Tests
Mem         XP, RP Quick Mem Tests
Float       XP 8032 Floating Point Tests
?           Display the Help Menu


Command ==>
```

The following table describes the optional arguments used with the XP, RP, Processor menu selections.

Option	*Default*	*Range*	*Description*
`pass`	1	0-2147483657	pass count.
`wcs`	0	0-1	XP wcs = 0, RP wcs = 1.
`proc`	1	1-2	XP = 1, RP = 2.
`loop`	0	0-1	loop off = 0, loop on = 1.
`test`	1	1-10	XP FPU (Transform Processor Floating Point Unit) tests 1-10.

Refer to the Help Menu for a list of valid options for each test.

The following paragraphs describe XP, RP Processor Test Menu selections.

A *pass=*

The **All** selection executes all of the tests on the XP, RP Processor Test Menu using their default settings.

D *pass=*

The *Default* selection executes the default test sequence for the XP, RP Processor Test Menu using the following default options:

XP branch
XP Sob branch
XP logical
XP memory
RP branch
RP sob branch
RP Logical
RP Memory
XP Fload
XP Fmpy
XP Flut
XP F_cc

C *pass=*

Option **C** executes the *Control/Status Register Test.* This test uses the VME bus to write a pattern into the GP2 Control register from the host processor. It then reads the GP2 Status register, comparing it and reporting any errors. The pattern is then incremented by 1 for 256 times. The `pass=` argument specifies the number of times the test should execute before it returns to the Menu.

S *pass= wcs number*

Option **S** executes the *XP, RP Status Byte Test.* This test is the first one that the Accel processors use. It tests the ability of the XP or RP to write to the processor's designated status byte. The data written is 0xff, 0x55, 0xAA, and 0x00.

H *pass=*

The **H** option executes the *Hand Shake SRAM Loc Test*, which tests the ability of the XP or RP to write to the shared RAM handshake status bits that the diagnostic uses to gather information from the GP2. The handshake status locations and definitions are in the Shared RAM Comm section. For more information refer to the XP, RP Processor Tests section.

CH *pass=*

The **CH** option executes the *Hand Shake SRAM Loc Test*, which is similar to the previously documented test, except that it uses ''C'' code and writes different data to the same memory locations. For more information refer to the Memory Test Descriptions section.

B *pass= proc= loop=*

The **B** option tests the branch instructions on the 7136 (Weitek sequencer) and the 7137 (Weitek Integer Unit) chip. These tests are down-loaded from the host processor into the XP or RP WCS. The tests communicate to the host processor through handshake locations in Shared RAM. The handshake protocol and definitions are found in the Shared RAM Comm section. A description of the branch test is given in the XP, RP Processor Tests section.

W *pass= proc=*

The **W** option executes the *WCS Addressing Lines Test,* which tests all of the address lines from the Weitek sequencer to the WCS memory. For a brief description of the test, refer to the XP, RP Processor Tests section.

O *pass= proc= loop=*

Option **O** executes the *SOB Branch Tests,* which are down-loaded from the host processor into the XP or RP WCS. See the XP, RP Processor Tests section for a descriptions of these tests:

SOB Test
SOB Neutralize Test
SHOB Test
Push Pop Tests
Call Tests

L *pass= proc= loop=*

The **L** option executes the Weitek 7137 chip's logical instructions. These tests are down-loaded from the host processor into the XP or RP WCS. See the XP, RP Processor Tests section for a description of the Logical Tests mentioned below.

All Logical Tests	Test 0
Logical Tests	Test 1
Register Tests	Test 2
Add, Sub Tests	Test 3
Condition Code Tests	Test 4
Merge, Deposit Tests	Test 5
Extract Tests	Test 6
Multiply Test	Test 7
Divide Test	Test 8

Use the test numbers to select individual tests (other than Test 0, the default):

M *pass= proc= loop=*

The **M** option executes the XP, RP Quick Mem Tests. These test are down-loaded from the host processor into the XP or RP WCS. See the XP, RP Processor Tests section for a description of the Mem Tests listed below.

Byte Alignment Test
Byte Store Test
Address Test
Memory Test

F *pass= test= loop=*

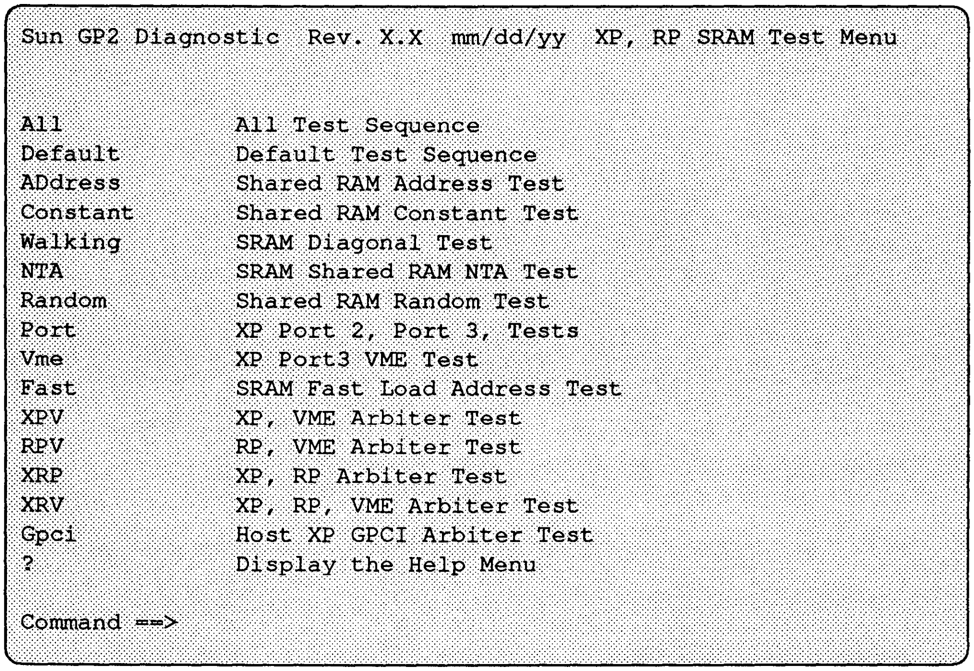
The **F** option tests the 3132 Weitek floating-point processor chip. These tests are down-loaded from the host processor into the XP WCS. There are 11 tests that communicate to the host processor through handshake locations in Shared RAM. See the Shared RAM Comm section for the handshake protocol and definitions. You may replace *test=* with numbers 1 - 11 to select individual tests; `test=0` is the default.

All Tests	Test 0
XP Fload Tests	Test 1
XP Fabs Tests	Test 2
XP Float Tests	Test 3
XP Fadd Tests	Test 4
XP Faddt Tests	Test 5
XP Fadd2 Tests	Test 6
XP Fmac Tests	Test 7
XP Fmul Tests	Test 8
XP Flut Tests	Test 9
XP Floating Point Conditions	Test 10
XP Fndloop Tests	Test 11

? The **?** option displays the Processor's Help Test Menu:

XP, RP Shared Memory Tests Menu

The XP, RP Shared Memory Test Menu displays the XP, RP Shared Memory arbiter test options.



```
Sun GP2 Diagnostic  Rev. X.X  mm/dd/yy  XP, RP SRAM Test Menu


All            All Test Sequence
Default        Default Test Sequence
ADdress        Shared RAM Address Test
Constant       Shared RAM Constant Test
Walking        SRAM Diagonal Test
NTA            SRAM Shared RAM NTA Test
Random         Shared RAM Random Test
Port           XP Port 2, Port 3, Tests
Vme            XP Port3 VME Test
Fast           SRAM Fast Load Address Test
XPV            XP, VME Arbiter Test
RPV            RP, VME Arbiter Test
XRP            XP, RP Arbiter Test
XRV            XP, RP, VME Arbiter Test
Gpci           Host XP GPCI Arbiter Test
?              Display the Help Menu

Command ==>
```

The following table describes the optional parameters used for the SRAM and Arbiter test menu selections.

Option	*Default*	*Range*	*Description*
`pass`	1	0-2147483657	pass count.
`loop`	0	0-1	loop off = 0, loop on = 1.
`sram`	1	1-2	XP = 1, RP = 2.
`test`	1	1-8	XRP tests 1-8.

Refer to the Help Menu for a list of valid options for each test.

A *pass=*

The **`All`** option executes all of the tests in the XP,RP Shared RAM Test Menu using their default settings.

D *pass=*

The **`Default`** option executes the default test sequence from the XP, RP Shared RAM Test Menu. The tests are executed using their default parameters.

AD, C, W, NTA and **R**

These XP, RP Shared RAM Tests are a subset of the Local RAM tests described in the next section (the integrity of the Shared RAM has already been verified). You need only to check the ability of the XP and RP to address and access the shared RAM, and to verify the arbiter. The shared memory is written from the either the XP or RP and read from it and the CPU processor. The following tests are executed, (see the Memory Test Descriptions section for test descriptions.)

NOTE *The Addressing test, the Constant test, and the Random test use only long word mode addressing while the NTA Test uses byte-mode addressing.*

Test Syntax

AD The addressing test syntax is:

`ad` *pass= loop= sram=*

C The Constant Test syntax is:

`c` *pass= loop= sram=*

Walking

The walking 1's SRAM diagonal test syntax is:

`w` *pass= loop= sram=*

NTA

The NTA Test syntax is:

`nta` *pass= loop= sram=*

R The random test syntax is:

`r` *pass= loop= sram=*

Test Options

P *pass= loop=*

The **P** option executes the *XP Port 2 and Port 3* tests. These tests perform various combinations of reads from the special hardware alignment ports. These ports were added because of the difference in the byte organization of the Weitek processor as compared to the CPU board processor. See the Memory Test Descriptions section for a more detailed description of the tests.

V *pass= loop=*

The **V** option executes the *XP Port3 VME Test*. This test writes a known pattern into the shared RAM then reads the data using fast loads from the 32 bit port at the same time that the host (through the VME interface) reads the shared RAM. If an error is detected it is reported. See the Memory Test Descriptions section for a more detailed description.

F *pass= loop= sram=*

The **F** option executes the *XP, RP Fast Address* test. This test is similar to the address test except that it uses Weitek's XP and RP processor fast load and fast store modes.

XPV *pass= loop=*

The **XPV** option executes the *XP and VME Arbiter Test,* which tests the arbitration between the XP and the VME or host processor. Both the XP and the VME interface write and read data to the Shared RAM at the same time. The XP writes and reads every other location starting at address 0x40400. The VME writes and reads every other location starting at address 0x40404. These tests write, read, and check a constant pattern. The pattern for the XP is 0x33333333. The constant pattern for the VME is 0xCCCCCCCC. The test reports any errors.

RPV *pass= loop=*

The **RPV** option executes the *RP and VME Arbiter Test,* which tests the arbitration between the XP and VME or host processor. The RP and the VME bus both write and read data to the Shared RAM at the same time. The RP writes and reads every other location starting at address 0x40400 while the VME writes and reads every other location starting at address 0x40404. These tests write, read, and check a constant pattern. The pattern for the RP is 0x99999999. The constant pattern for the VME is 0xCCCCCCCC. The test reports any errors.

XRP *pass= loop= test=*

The **XRP** option executes the *XP and RP Arbiter* tests, which test the arbitration between the XP and RP. The ten tests in this group are described in the Memory Test Descriptions section.

XRV *pass= loop= t=*

The **XRV** option executes the *XP, RP and VME Arbiter Test,* which tests the arbitration between the XP, RP and the host. The XP, RP and VME bus all write and read data to the Shared RAM at the same time. The XP writes and reads every third location starting at address 0x40400. The RP writes and reads every third location starting at address 0x40404. The VME writes and reads every third location starting at address 0x40408. These tests write, read and check a constant pattern. The pattern for the XP is 0x33333333. The constant pattern for the RP is 0x99999999. The pattern for the VME is 0xCCCCCCCC. The test reports any errors. The second test with this option is similar except that XP and RP use the Weitek Chip Set fast load and fast store modes.

Gpci *pass=*

This option executes the host *XP GPCI Arbiter Test.* This test simulates as closely as possible the normal interaction between the host and the GP2 processor. The host processor writes a command to the Shared RAM, and the XP reads it and tests some memory. While the XP is testing a portion of the memory, the host writes more commands into the Shared RAM. This test tests the arbiter in a way that differs from the previously described tests.

FIFO Tests Menu

The FIFO Test Menu is divided into three screens. When you select this menu, the following options are displayed:

```
Sun GP2 Diagnostic  Rev:.x.x MM/DD/YY  Fifo Tests
Menu

  All          All Test Sequence
  Default      Default Test Sequence
  Func         Fifo Functional Test Menu
  Ram          Fifo Ram Test Menu

Command ==>
```

A *pass=*

Option **A** executes all of the tests on the FIFO Test Menu using their default settings as described later in this test.

D *pass=*

Option **D** executes all of the tests on the FIFO Test Menu using their default settings as described later in this test.

F

Option **F** displays the *FIFO Function Test Menu.*

R

Option **R** displays the *FIFO Ram Tests Menu.*

FIFO Function Tests Menu

Entering **F** from the FIFO Tests Menu brings up the FIFO Function Test Menu:

```
Sun GP2  Diagnostic  Rev. X.X  mm/dd/yy  Fifo Function Test Menu


All                     All Test Sequence
Default                 Default Test Sequence
Creg                    Fifo Reg Reset Check Test
One                     Fifo One Word Write Test
Half                    Fifo Half Full Write Test
Full                    Fifo Full Write Test Test
Vful                    Fifo Over Full Write Test
Slow                    Fifo SLow Write Test
Quick                   Fifo Fast Store Test
Neut                    Fifo Neutralize Test
B                       Fifo 512 Word Read Back Test
?                       Display the Help Menu

Command ==>
```

The following table describes the optional arguments to the FIFO Menu selections.

Option	*Default*	*Range*	*Description*
`pass`	1	0-2147483657	pass count.
`loop`	0	0-1	loop off = 0, loop on = 1.
`fifo`	0	0-1	XP RP fifo=0, RP PP fifo = 1.
`size`	0xc00	0=03c00	number of words to send to FIFOs.
`pat`	0xa5a5a5a5	0-0xffffffff	data pattern used in test.

Refer to the Help Menu for a list of valid options for each test.

FIFO Test Menu Options

A *pass=*

Selecting **All** executes all of the tests in the FIFO Test Menu using their default settings.

D *pass=*

Selecting **Default** executes the *Default Test Sequence,* all of the tests on the FIFO Test Menu using their default settings.

C *pass=*

Selecting **C** executes the *FIFO Reg Reset Check Test.* This test resets the GP2 board, then down-loads micro code into the XP WCS. It then reads the FIFO Flag register and expects to read 0xe. The test posts any errors.

O *pass= fifo=[0,1] loop=[0,1]*

Selecting **O** executes the *FIFO One Word Write Test.* This test writes the word 0x55555555 to the RP through the FIFO interface. The RP is in a wait loop, expecting data to be put into the FIFO. When the RP sees that there is data in the FIFO, it reads the data from the FIFO and writes it to a known memory location in the Shared RAM (0x40400). The XP, after writing data into the FIFO, waits for the FIFO status register to tell the XP that the FIFO is empty.

The XP then reads the known memory location in the Shared RAM and checks it against the data sent. The test posts any errors. When you select the RP, PP FIFO test, the data sequence is identical, except that the data is sent back to the RP through the read-back register.

H *pass= fifo=[0,1] loop=[0,1]*

Selecting **H** executes the *FIFO Half-Full Write Test.* This test writes 256 words of "0x55555555" data to the RP through the FIFO interface. The XP then checks the FIFO half-full bit in the FIFO Status register. If it is not set, the test posts an error. The RP sees data in the FIFO. It then reads the data and writes it to a known location in Shared RAM (0x40400). The XP waits until the FIFO is empty, then reads and checks the data that is put in the known locations in Shared RAM. The test posts any errors.

When you select the RP, PP FIFO test, the RP writes 256 words of 0x55555555 data to the PP through the FIFO interface. The PP is in a loop, checking data to be put into the FIFO. If there is data in the PP FIFO , the test writes this data to the read-back register. Then the PP waits until the read-back register is empty before reading the FIFO again. The RP, after half-filling up the FIFO, checks the half-full status bit. If set, it then polls the read-back status bit to see if the PP has put data into it. When the read-back register has data, the RP reads and checks the data. The test posts any errors.

F *pass= fifo=[0,1] loop=[0,1]*

Selecting **F** executes the *FIFO Full-Write Test.* This test writes 512 words of the walking-ones data pattern to the FIFO, which should fill it. The FIFO full bit should be on in the FIFO status register. The test posts any errors. The RP then reads the FIFO and writes the data to a known location in Shared RAM (starting at 0x40400). The XP then checks the FIFO empty status bit and posts an error if it is never empty. The XP then reads and checks the data from the known locations in the Shared RAM. The test posts any errors.

When the RP, PP FIFO test is selected, the RP writes 512 words of 0xAAAAAAAA data to the PP through the FIFO interface. The PP is in a loop checking data to be put into the FIFO. If there is data in the PP FIFO, it writes it to the read-back register. Then the PP waits until the read-back register is empty before reading the FIFO again. The RP, after filling up the FIFO, checks the full status bit. If set, it then polls the read back status bit to see if the PP has put data into it. When the read back register has data, the RP reads and checks the data. The test posts any errors.

V *pass= fifo=[0,1] loop=[0,1]*

Selecting **V** executes the *FIFO Over-Full Write Test.* This test is similar to the previous test, except that it writes more than 512 data words to the FIFO. On the 513th data word, the FIFO should hang the XP because it is full. The RP then begins to empty the FIFO, writing the data to Shared RAM. This action frees the XP so that it can put more data into the FIFO. The XP then reads and checks the data in Shared RAM. The test posts any errors. The data written to the FIFO is a walking 0's pattern in a field of 1's.

When the RP, PP FIFO test is selected, the RP writes more than 512 words of 0xAAAAAAAA data to the PP through the FIFO interface. On the 513th data word, the FIFO should hang the RP because the FIFO is full. The PP is in a loop, checking data to be put into the FIFO. If there is data in the PP FIFO it writes it to the read-back register. Then the PP waits until the read-back register is empty before reading the FIFO again. An error is detected if the RP does not hang. The host processor should stay in a wait loop, waiting for the RP to post status that it has completed the test (which should not happen). This wait loop lasts about 30 seconds. The test posts any errors.

S *pass= loop=[0,1]*

Selecting **S** executes the *XP, RP FIFO Slow Test*, which checks the RP hang condition when the RP tries to read an empty FIFO. The XP writes a word into the FIFO. Then the RP waits until the FIFO is not empty, reads it, and writes the data to a known location in shared RAM (starting at 0x40400) before trying to read the FIFO again.

The XP, after writing data to the FIFO, waits 120 clock cycles, then writes the FIFO again. This causes the RP to lock up until the FIFO is written. The process is repeated 576 times. The data transferred is 0x55555555. After all of the data has been transferred, the XP checks that the RP wrote into shared RAM. The test posts any errors.

Q *pass= fifo=[0,1] loop=[0,1]*

Selecting **Q** executes the fast-store instructions to the FIFO. This test writes one word of data 16 times, using the floating-point registers every clock cycle. This loop is repeated 24 times, for a total of 576 words transferred to the FIFO. The data pattern is an address pattern. The RP reads the data and puts it into known location of shared RAM (starting at loc 0x40400). After all of the data has been transferred to the FIFO, the XP waits for the FIFO to be empty. It then checks the data in shared RAM, checking 16 successive writes to the XP FIFO, which also fill the FIFO. The test posts any errors.

This test is the same for the RP, PP FIFO except that it performs 16 output instructions consecutively.

N *pass= fifo=[0,1] loop=[0,1]*

Selecting **N** executes the *FIFO Neutralization Test*. The XP writes one word to the XP FIFO. The RP executes a branch instruction followed by an input instruction (RP reads the XP FIFO), then executes another branch instruction followed by an output instruction (a write PP FIFO). The RP should read one word from the XP FIFO and write it to the PP FIFO. Then the RP checks the Flag registers for the conditions of both FIFO' s. The XP FIFO should be empty, and the PP FIFO should not be empty. The test reports any errors.

B *pass= fifo=[0,1] loop=[0,1]*

Selecting **B** executes the *FIFO 512-Word Read-Back Test*. This test is executed only between the XP and RP. This test is similar to the FIFO Full Test with exception that instead of the RP's writing data to shared RAM, it writes it to the read-back register. The XP then reads the data from the read-back register, and checks and compares it. The test reports any errors. The data used for this test is 0xAAAAAAAA, 0x55555555, repeated 256 times.

FIFO RAM Tests Menu

Entering **R** from the FIFO Tests Menu brings up the FIFO RAM Tests Menu:

```
All         All Test Sequence
Default     Default Test Sequence
WA          Fifo Address Test
W1          Fifo Walking 1's Test
W0          Fifo Walking 0's Test
Rand        Fifo XP, RP Random Test
Konst       Fifo XP, RP, PP Constant Test
Prand       Fifo XP, RP, PP Random Test
Marching    XP, RP, PP Marching Ones Scopeloop
Exer        XP, RP System Exerciser Test
?           Display the Help Menu
```

The *FIFO RAM Test Menu Options* are listed below.

A *pass=*

Option **A** executes all of the tests on the FIFO RAM Test Menu using their default settings.

D *pass=*

Option **D** executes all of the tests in the FIFO RAM Test Menu, with their default settings.

WA *pass= fifo=[0,1] loop=[0,1]*

Option **WA** from the FIFO RAM tests menu executes the FIFO Address Test. This test writes an address pattern into each FIFO. This process tests the address pointers and counters inside each of the FIFO s. Once the data has been written into the FIFO, the XP or RP reads the readback register and checks the data. The test reports any errors.

W1 *pass= fifo=[0,1] loop=[0,1]*

Option **W1** executes the *FIFO Walking 1's Test.* This test writes a walking ones pattern into each FIFO. This process tests the data sensitivity of each FIFO. Once the data has been written into the FIFO, the XP or RP reads the readback register and checks the data. The test reports any errors.

This test starts out writing a pattern (0x01010101) to the FIFO. It then reads the pattern written and saves it into Shared RAM. The data is then shifted left one bit and written to the FIFO. This action continues 512 times. Then the data is checked, and any errors posted. Next, the same sequence is executed with two words written to the FIFO before any are read. This action continues until 512 words have been written into the FIFO before any are read.

WO *pass= fifo=[0,1] loop=[0,1]*

Option **WO** executes the *FIFO Walking 0's Test.* This test writes a walking zeroes pattern into each FIFO. This process tests the data sensitivity of each FIFO. Once the data has been written into the FIFO, the XP or RP reads the readback register and checks the data. Errors detected are reported.

This test first writes a pattern (0xfefefefe) to the FIFO. It then reads the pattern written and saves it in Shared RAM. Next the data is shifted left one bit and written to the FIFO, which is repeated 512 times. Afterwards, the data is checked, and any errors reported. Next, the test sequence is executed with two words written to the FIFO before any are read. This action continues until 512 words have been written into the FIFO before any are read.

R *pass= loop=[0,1]*

Selecting **R** executes the *FIFO Random Test.* This test writes a random pattern of 3072 (0xc00 hex) words to the FIFO. The RP reads the data, writes it to shared RAM, and then waits for 120 cycles, allowing the XP FIFO to fill and hang for most of the test. After the XP has written all of the data, it waits for the FIFO to be empty, then checks the data in shared RAM. The test reports any errors.

K *pass= size= pat= loop=[0,1]*

Selecting **K** executes the *FIFO Constant Test.* This test checks a constant pattern through the XP FIFO and PP FIFO, then back through the PP read-back register to RP, then to Shared RAM.

The XP then writes the data to the FIFO, RP reads the data and writes it to the PP. The PP reads the FIFO and writes it back to the read-back register. The RP, after filling the PP FIFO, begins to read the read-back register and write the data to Shared RAM at location 0x40400. After the XP has written the data, it waits for the FIFO to empty, then check the data in Shared RAM. The test reports any errors.

P *pass= size= pat= loop=[0,1]*

Selecting **P** executes the *FIFO Random Test* through all stages. This test is similar to the Constant Test except that the data sent through the FIFO is random.

M

Option **M** executes the *FIFO XP RP PP Marching Ones Scopeloop* test, which fills the FIFO with all zeroes except for the last location of the FIFO, which it fills with all ones. The RP reads the FIFO and writes the data to Shared RAM where the XP reads and checks the data. If the test detects an error, it halts. On the next pass, the XP writes 510 words of zeros: one word of ones and one word of zeros. The XP continues writing one less word of zeros until the first location of the FIFO is written with all one's. Then the test starts over.

This test is useful for checking FIFO signals with the oscilloscope, and for testing FIFO addressing.

A message that looks like this should appear, requesting a carriage return to quit:

```
Enter <CR> to quit.
```

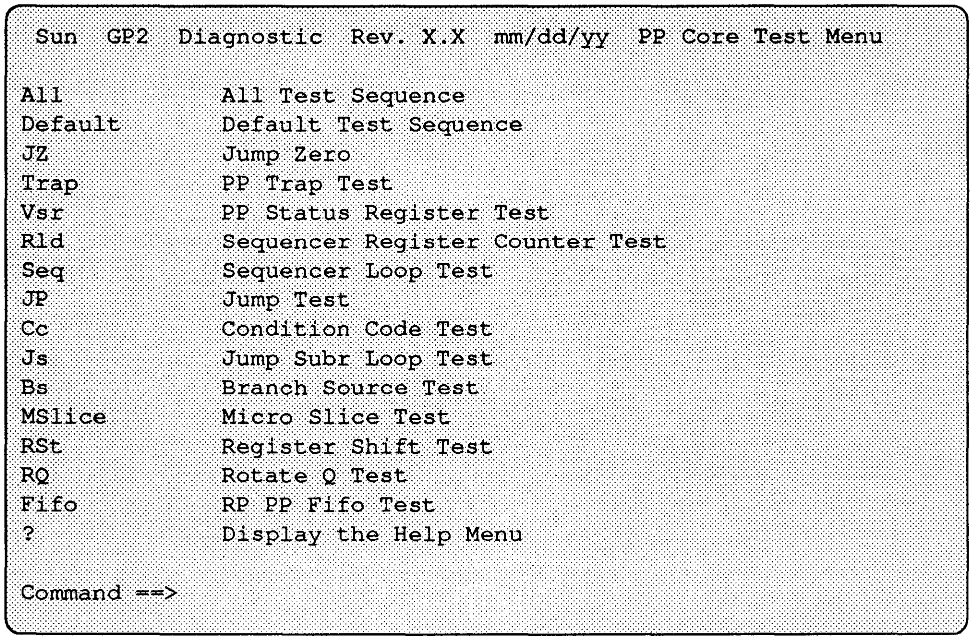
E *pass=*

This option executes the FIFO exerciser test, which is similar to the GPCI test described above, except that the data is sent through the FIFO s (XP, RP, PP) and then stored into Shared RAM and checked.

? Entering **?** displays the FIFO Test Help Menu.

PP Core Test Menu

The PP Core Test Menu has the following options:



```
 Sun  GP2  Diagnostic  Rev. X.X  mm/dd/yy  PP Core Test Menu

All           All Test Sequence
Default       Default Test Sequence
JZ            Jump Zero
Trap          PP Trap Test
Vsr           PP Status Register Test
Rld           Sequencer Register Counter Test
Seq           Sequencer Loop Test
JP            Jump Test
Cc            Condition Code Test
Js            Jump Subr Loop Test
Bs            Branch Source Test
MSlice        Micro Slice Test
RSt           Register Shift Test
RQ            Rotate Q Test
Fifo          RP PP Fifo Test
?             Display the Help Menu

Command ==>
```

A *pass=*

Selecting **`All`** from the PP Core Test Menu executes all of the non-scope tests from the PP Core Test Menu using their default settings. The *pass* argument specifies the number of times the test should execute before it returns to the Menu.

D *pass=*

Selecting **Default** executes the default test sequence for the PP Core Test Menu, using default test parameters.

These following tests are performed:

Option	*Description*
Trap	PP Trap Test
Vsr	PP Status Register Test
Rld	Sequencer Register/Counter Test
JP	Jump Test
Cc	Condition Code Test
Js	Jump Subr Loop Test
Bs	Branch Source Test
Mslice	Micro Slice Test
RSt	Register Shift Test
RQ	Rotate Q Test

The *pass* argument specifies the number of times the test should execute before it returns to the Menu.

JZ Selecting **JZ** executes a single instruction jump to itself. It is the simplest program to perform and it loops forever. This is a scope loop test. No arguments are required.

Trap *pass=*

The **Trap** selection executes five consecutive trap instructions. This test ensures the functionality of starting and halting the PP.

Vsr *pass= scope=[0,1]*

Selecting **Vsr** executes the *PP Status Register Test.* This test loads the 8-bit PP VME status register with data and halts after each load. The order in which the data is loaded is 55, AA, 33, and CC.

Setting the `scope` parameter to `scope=1` causes the PP to loop forever. The last instruction of the microcode is a jump to the first instruction, which is true for all PP options that have scope loop capability.

RLD *pass=*

Selecting **Rld** executes the *Sequencer/Register Counter Test.* This test loads the ALU register/counter with a value X and loops for X + 1 times. Internal to the loop, a register is incremented. Upon completion of the loop, the incrementing register is checked to see if the correct number of increments was executed. The value X is first 555, then AAA.

Seq *pass=*

Selecting **Seq** executes the *Sequencer Loop Test.* This test is similar to the Trap test, except now every microword of the writable control store contains a trap instruction. A count is kept to ensure that every location has been stepped through.

JP *pass= scope= [0,1]*

Selecting **JP** executes the jump test. This test performs a sequence of address jumps to test the PP's jump to address functionality. The jumps are located at addresses 0x333, 0x555, 0xAAA, 0xCCC.

Cc Selecting **Cc** executes the *Condition Code Test.* This test checks all of the different kinds of branch operations available. The condition jump pipeline micro-instruction is frequently used. All condition codes, for example, true, false, eq, ne, lt, gt, le, and ge are tested.

Js *pass= scope=[0,1]*

Selecting **Js** executes the *Jump to Subroutine Test.* This test checks that when a jump subroutine instruction is performed, a valid return occurs. Subroutines are located at 0x333, 0x555, 0xAAA, and 0xCCC.

Bs *pass=*

Bs executes the *Branch Source Test.* This test performs branch source functions vec16 and vec128. The jump locations that are performed are a combination of the contents of the branch mux and which vector (16 or 128) is being tested.

Mslice *pass=*

Selecting **Mslice** executes the *Micro Slice Test.*

Fifo *pass= scope=[0,1]*

Selecting **Fifo** executes the *RP/PP FIFO Read-back Test.* This test loads the RP/PP FIFO until it is full and then writes one more, which causes the RP to hang, then reads back and compares all of the data.

RSt *pass= scope=*

The *Register Shift Test* starts with 0x80003000, a negative number. It then performs ten LEFT shifts and compares the data with the expected data 0x00c0000. Since left shifts are not sign-extended, the answer should be positive. It then traps and prints an error message if an error was found. Next, ten RIGHT shifts are performed on the value 0x80c00000 and the data is compared with 0xffe03000. Right shifts are sign extended, meaning that the answer remains negative.

If the test encounters an error, one of these messages may appear:

```
*****PP rst error: register shift left.
*****PP rst error: register shift right.
```

RQ *pass= scope=[0,1]*

The *Rotate Q Test* checks the rotation of the q register upon shifting either left or right. The test starts by initializing the q register of the ALU to the value 0x1. The q register is rotated once to the RIGHT and then compared with the expected value 0x80000000. An error message is printed if the compare failed. The q register is next initialized to 0x80000000 and then rotated once to the LEFT. The q register then is compared with the expected value 0x1. An error message is printed if the compare failed. The q register is next initialized to 0x00000032 and then rotated 34 times to the LEFT. The q register then is compared with the expected value 0x000000c8. An error message is printed if the compare failed. The final test has the q register initialized to 0x00003200 and then rotated 34 times to the RIGHT. The q register then is compared with the expected value 0x0000c800. An error message is printed if the compare failed.

If the test encounters an error, one of these messages may appear:

```
*****PP rq error: single rotate right.
*****PP rq error: single rotate left.
*****PP rq error: 34 left rotates.
*****PP rq error: 34 right rotates.
```

Fifo

Entering **F** selects and executes the RP PP Fifo Test.

?

Entering **?** displays the PP Core Test Help Menu.

PP Hardware Control Test Menu

The PP Hardware Control Test Menu has these options:

```
All             All Test Sequence
Default         Default Test Sequence
W1              PP Data Bus walking 1's Test
W0              PP Data Bus walking 0's Test
DBX             PP Data Bus EXOR Test
Imm             PP Immediate Scope Test
DBrw            PP Data Bus Read/Write Test
Bar             Addr Gen Barrel Shifter Test
Off             Addr Gen Offset Generator Test
Noise           ALU noise Test
Idt             IDT ALU noise Test
DIther          Dithering Test
DBS             PP Data Bus Scope Test
DBC             PP Data Bus Constant Scope Test
AGS             Address Generator Scope Test
Graph           PP Graphics Scope Test
?               Display the Help Menu

Command ==>
```

The graphics control field of the pipeline register provides control of the address generator, depth buffer, and frame buffer interface.

PP Hardware Control Test Menu Selections

A *pass=*

Selecting **All** executes all of the tests in the PP Hardware Control Test Menu. The *pass* option specifies how many passes to execute.

D *pass=*

Selecting **Default** executes the default test sequence, which performs the following tests:

W1	PP Data Bus walking 1's Test
Bar	Addr Gen Barrel Shifter Test
Off	Addr Gen Offset Generator Test
DIther	Dithering Test

W1 *pass= scope=[0,1]*

Selecting **W1** executes the *PP Data Bus walking 1's Test.* This test checks the integrity of the PP data bus by shifting a 1 from the immediate field of the microword into the ALU r0 register by way of the PP data bus. Each bit is compared upon each shift and an error message is displayed at the first failing compare. For scope loops, the test performs a jump to zero on a failure. On the cycle with the spare bit, the data received over the PP data bus into the r0 register is placed back on the data bus. The data sent to the r0 register is compared with a similarly shifting q register.

W0 *pass= scope=[0,1]*

Selecting **W0** executes the *PP Data Bus walking 0's Test.* This test is similar to the W1 test except that a zero is walked through the PP data bus. Putting the data into the ALU r0 register now takes two cycles because the data from the immediate field holds only 16 bits and the data bus is a 32-bit bus.

DBX *pass= scope=[0,1]*

Selecting **DBX** executes the *PP Data Bus exor test.* This test exclusive-ors 32 1's from the immediate field of the PP microword into the ALU r0 register. After all 32 bits have been xor-ed, the r0 register is compared with the q register.

DBrw

Selecting **DBrw** executes the *PP Data Bus Read/Write Test.* This test writes, reads and compares fives and A's to the gmd, amd, and bsel registers.

Imm *pass= scope=[0,1]*

Selecting **Imm** executes the *PP Immediate Scope Test.* This test continuously places a selected pattern onto the PP data bus. The lower 16 bits of data are placed into the least significant slice of the PP ALU, and in the next cycle the upper 16 bits are placed into the most significant slice. The data sent to the ALU is then placed onto the PP data bus on the next cycle.

Bar

Selecting **Bar** executes the *Address Generator Barrel Shifter Test.* The d x, ad x, and ad y circuitry are tested. Both the ld x and ad x perform "walking 1's" tests. The ad y consists of performing a "walking 1's" test and a scan test. The scan test does a ld x with a 0 and a ld y, incrementing y for each scan line. The ad y tests are performed with the screen width of 1024 and 1152.

Off

Selecting **Off** executes the *Addr Gen Offset Generator Test.* This test performs combinations of x, y, x and y increment, x, y, xy decrement. All y and xy tests are performed at widths 1024 and 1152.

NOise *pass= scope=[0,1]*

Entering **no** executes the ALU Noise Test. Also, this test and the **IDT** test check for internal noise in the IDT component part on the GP2 board.

IDT *pass= scope=[0,1]*

Selecting **idt** executes another ALU Noise Test, similar to the *noise* test.

DIther

Selecting **DIther** executes the *Dithering Test.* This test checks the dither pattern memory. A unique pattern is generated for each x or y scan line increment. For each x/y combination as y = 0 to 15 and x = 0 to 15, an 8-bit value is determined in the dither circuitry and in the ALU. The two values are then compared.

DBS

Selecting `DBS` executes the *PP Data Bus Scope Test,* which is specifically a scope loop test. It loads A's and 5's into the `gmd, ival, bra, zval, amd, bsel`, and `rp`. Reads are done to the `gmd, ival, zval, amd`, and `bsel`. Reads are not done to the `rp`.

DBC

Selecting `DBC` executes the *PP Data Bus Constant Scope Test.* This test puts a constant onto the data bus through the immediate field of the writable control store and loops forever. The default is AAAA into the lower 16 bits of the data bus.

AGS

Selecting **AGS** executes the *Address Generator Scope Test.* This test is specifically used for oscilloscope testing. The test does a combination of `ld x`'s and `ad y's, xce`'s and `yce`'s.

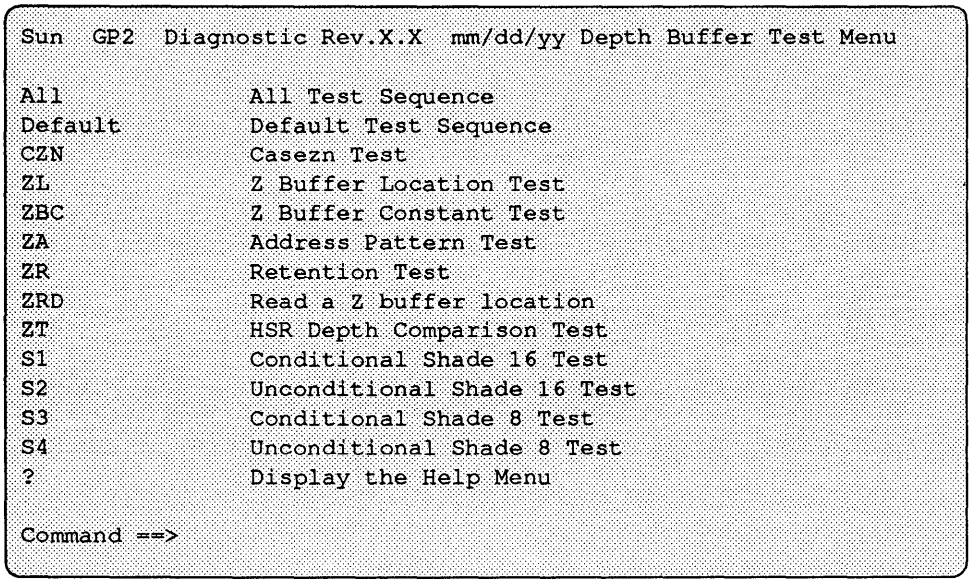
Graph

Selecting **Graph** executes the *PP Graphics Scope Test.* This test is specifically used for oscilloscope testing. The test does an `mrd, mwr, xce, yce, zrq`, and an `frq`. This test loops forever.

? Entering **?** displays the Help Menu.

Z (depth) Buffer Test Menu

The Z (depth) Buffer Test Menu has these options:



```
Sun  GP2  Diagnostic Rev.X.X  mm/dd/yy Depth Buffer Test Menu

All             All Test Sequence
Default         Default Test Sequence
CZN             Casezn Test
ZL              Z Buffer Location Test
ZBC             Z Buffer Constant Test
ZA              Address Pattern Test
ZR              Retention Test
ZRD             Read a Z buffer location
ZT              HSR Depth Comparison Test
S1              Conditional Shade 16 Test
S2              Unconditional Shade 16 Test
S3              Conditional Shade 8 Test
S4              Unconditional Shade 8 Test
?               Display the Help Menu

Command ==>
```

The `scope=` parameter causes the test to loop forever when set to 1.

A *pass=*

Selecting **All** executes all of the tests on the Depth Buffer Test Menu. The *pass* option specifies how many passes to execute.

D Selecting **D** executes the default test sequence. The tests executed are:

```
CZN          Casezn Test
ZL           Z Buffer Location Test
ZBC          Z Buffer Constant Test
ZA           Address Pattern Test
ZT           HSR Depth Comparison Test
S1           Conditional Shade 16 Test
```

CZN *pass= scope=*

Selecting **CZN** executes the *casezn Test.* This test performs the n-way branches for z-buffer availability and ALU status. It tests the different combinations of z-buffer available or busy and ALU negative or positive status.

ZL *pass= scope= addr= pat=*

Selecting `ZL` executes the *Z Buffer Location Test.* You can specify an address and a pattern. The test writes, reads, and compares the 16-bit pattern before performing the same test using the pattern's exclusive-or pattern. The results are passed through the RP to be placed into shared RAM, and then displayed. The default address is 0 and the default pattern is 5555. See the beginning of the Z Buffer test descriptions for information on the `scope=` setting.

ZBC *pass= scope= pat=*
Selecting **ZBC** executes the *Z Buffer Constant Pattern Test.* This test detects stuck bits at faults. A write, read compare sequence is performed using an alternating byte pattern (0xAAAA, 0x5555) for all depth buffer memory. The pattern is then inverted and the test performed again. See the beginning of the Z Buffer Test descriptions for information on the `scope=` setting.

ZA Selecting **ZA** performs the *Z Buffer Address Pattern Test.* This test detects coupling faults. Write, read, compare, sequences are done for all Z buffer memory.

ZR *pass= scope=*
Selecting **ZR** performs the *Z Buffer Retention Test.* This test checks the static column DRAM refresh of the Z buffer. The memory is written to, delayed and read back for comparison. See the beginning of the Z Buffer Test descriptions for information on the `scope=` setting.

ZRD *pass= scope=*
This test reads the data of a Z buffer location. See the beginning of the Z Buffer Test descriptions for information on the `scope=` setting.

NOTE *The following commands cannot be used for CG9 testing:*

ZT *pass= scope=*
Selecting **ZT** executes the *HSR Depth Comparison Test.* This test performs hidden surface removal tests of <, <= >, and >=. The data is checked from both the Z buffer and the data sent to the CG. See the beginning of the Z Buffer Test descriptions for information on the `scope=` setting.

S1 *pass= scope=*
Selecting **S1** executes the *Conditional Shade 16 Test.* See the beginning of the Z Buffer Test descriptions for information on the `scope=` setting.

S2 *pass= scope=*
Selecting **S2** executes the *Unconditional Shade 16 Test.*

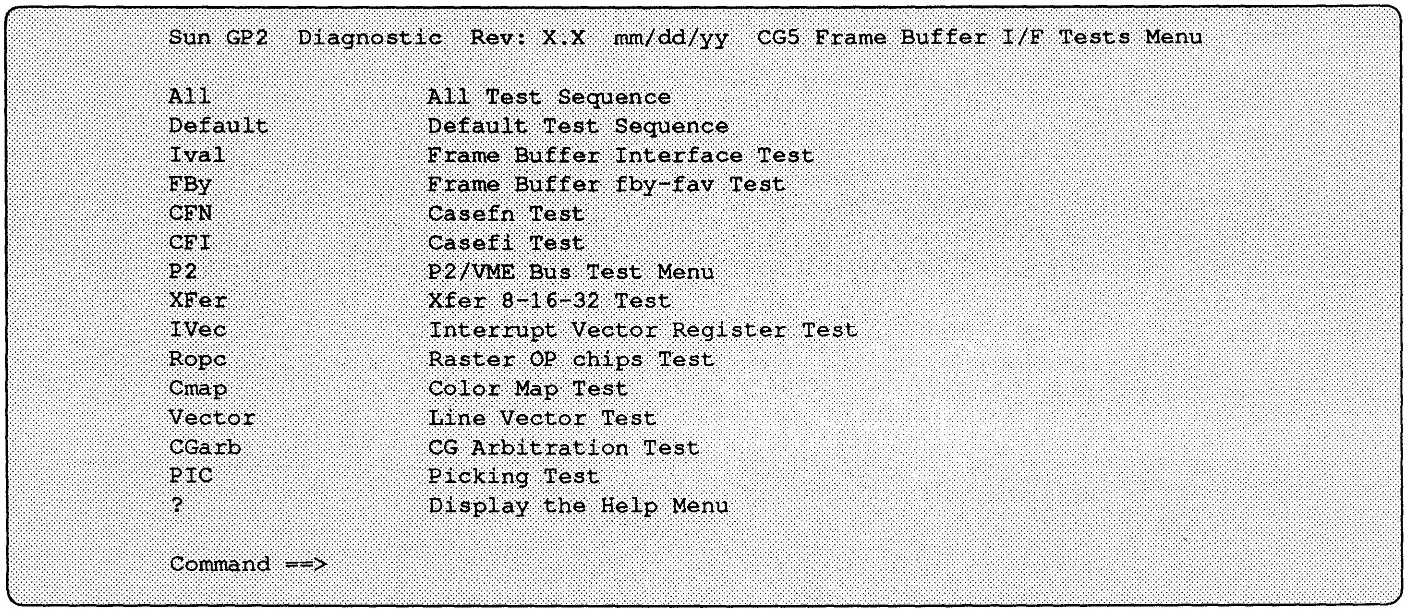
S3 *pass= scope=*
Selecting **S3** executes the *Conditional Shade 8 Test.* See the beginning of the Z Buffer Test descriptions for information on the `scope=` setting.

S4 *pass= scope=*
Selecting **S4** executes the *Unconditional Shade 8 Test.* See the beginning of the Z Buffer Test descriptions for information on the `scope=` setting.

? Selecting "?" displays the help menu.

CG5 Frame Buffer Interface Tests

The CG5 Frame Buffer Interface Test Menu appears as shown below:



```
Sun GP2  Diagnostic  Rev: X.X  mm/dd/yy  CG5 Frame Buffer I/F Tests Menu

All                 All Test Sequence
Default             Default Test Sequence
Ival                Frame Buffer Interface Test
FBy                 Frame Buffer fby-fav Test
CFN                 Casefn Test
CFI                 Casefi Test
P2                  P2/VME Bus Test Menu
XFer                Xfer 8-16-32 Test
IVec                Interrupt Vector Register Test
Ropc                Raster OP chips Test
Cmap                Color Map Test
Vector              Line Vector Test
CGarb               CG Arbitration Test
PIC                 Picking Test
?                   Display the Help Menu

Command ==>
```

The Frame Buffer I/F Test options are described in the following paragraphs.

A *pass=*

Selecting **All** executes the All Test Sequence of the Frame Buffer Interface Test Menu using their default settings.

D *pass=*

Selecting **D** executes the default test sequence. The following tests are executed using their default parameters:

```
Ival          Ival Register Test
FBy           Frame Buffer fby/fav Test
CFN           Casefn Test
CFI           Casefi Test
XFer          Xfer 8/16/32 Test
Sreg          FB Status Register Test
Vector        Line Vector Test
CGarb         CG Arbitration Test vme=5 pp=0-5
```

Ival

Selecting **Ival** executes the *Ival Register Test.* The 32-bit Ival register is the data latch for the P2 interface. The sequence of tests performed are:

- 32 bit transfers of walking 1's,
- 32 bit test of A's, 5's, C's and 3's,

- 16 bit tests of A's, 5's, C's and 3's,
- 8 bit tests of A's, 5's, C's and 3's.

CFN *pass= scope=[0,1]*

Selecting **CFN** executes the *casezn Test.* This test performs the n-way branches for frame buffer availability and ALU status. The different combinations of frame buffer available or busy, positive or negative ALU status, are tested.

P2 If you enter P2 from the Frame Buffer Interface Test Menu, The P2 Bus Interface Test Menu is displayed. This menu contains eight tests that exercise the VME bus or the P2 bus separately. (See CG arbitration test discussion in the *CG5 Frame Buffer Interface Tests* section.

CFI *pass= scope=[0,1]*

Selecting **CFI** executes the *casefi* test. This test performs the n-way branches for frame buffer availability and the RP-PP FIFO status. The combinations tested are:

- frame buffer busy and FIFO empty,
- frame buffer busy and FIFO available,
- frame buffer available and FIFO empty,
- frame buffer available and FIFO available.

XFer *pass= scope=[0,1]*

Selecting **XFer** executes the *Xfer 8-16-32 Test.* This test sends 8-, 16-, and 32-bit data to the CG5 over the P2 interface, testing the integrity of the interface. The 8-bit test writes 0x2233 to one address on the frame buffer, then writes the complement to the next address on the frame buffer. The data is read back and compared. For the 16- and 32-bit test, the data is sent as a 16-bit word to one address, then as a 32-bit word to the next address. Both locations are read back and the data is compared.

IVec *pass= scope=[0,1] pat=*

Selecting **IVec** executes the *Interrupt Vector Register Test,* which sends a pattern to the GP2 interrupt vector register. The interrupt vector register can then be read from the VME bus to see if the pattern is correct.

Ropc *pass= scope=[0,1]*

Selecting **Ropc** executes the *Raster OP chips Test,* which writes, reads, and compares all of the registers of one ROP chip.

Cmap *pass= scope=[0,1]*

Selecting **Cmap** executes the *Color Map Test.* The first part of this test is visual and the screen should flash red, green, and blue. The second part of this test writes 0 to 255 into the red look up table (LUT), reads it back and compares. The data is reversed starting with 255 down to 0, and writes, reads and compares are again performed. The same procedure is done to the green and blue LUTs.

Vector *pass= scope=[0,1]*

Selecting **Vector** executes the *Line Vector Test,* which is a visual test. A box, diagonals, and a square appear on the screen. The box vertices are at (0,0), (1151,0), (1151,899), and (0,899). One diagonal ranges from (0,0) to (899,899), and the other from (899,0) to (0,899). The square is made of incremental vectors radiating from a single point.

CGarb *pass= scope= vme=[0-5] pp=0-5]*

Selecting `CG` executes the CG Arbitration Tests. These tests work on the CG5 from both the VME host side and the P2 bus to test the arbitration circuitry. Testing different address spaces is available. The choices are:

	VME	P2
0	--	--
1	control	control
2	pixel	pixel
3	word	word
4	rop	rop
5	ctrl/pix/word	ctrl/pix/word

For example, selecting `vme=1 pp=3` executes the arbitration test in which the VME side writes to the CG in control space, and the P2 side writes to the CG in word space. Selecting a 5 performs control, pixel, and word read/write accesses.

? Selecting **?** Displays the Help Menu.

P2 Bus Interface Test Menu

Selecting **P2** from the Frame Buffer Interface Test Menu brings up the following menu of tests that exercise the VME bus or the P2 bus.

```
             Sun  GP2  Diagnostic  Rev:X.X MM/DD/YY  P2 Bus Interface Test Menu

            All          All Test Sequence
            Default      Default Test Sequence
            VW           VME Write
            VR           VME Read
            VWRC         VME Write/Read/Compare
            PW           P2 Write
            PR           P2 Read
            PWRC         P2 Write/Read/Compare
            Sreg         FB Status Register Test
            Perplane     Per_Plane Register Test
            ?            Display the Help Menu
```

The following pages describe the P2 Bus Interface Test Menu options.

All

Selecting `all` executes the following sequence:

set perplane mask through VME:

```
vwrc xfer=8 addr=30a001 pat=ff
```

test xfer types in pixel space:

```
vwrc xfer=32 pat=aa55aa55
vwrc xfer=16 pat=33cc
vwrc xfer=8 pat=c3
pwrc xfer=32 pat=55aa55aa
pwrc xfer=16 pat=cc33
pwrc xfer=8 pat=3c
```

test the status register and perplane registers in control space:

```
sreg
perplane
```

test word space

```
vwrc xfer=32 addr=0 pat=aa55aa55
pwrc xfer=32 addr=0 pat=55aa55aa

vwrc xfer=16 addr=0 pat=33cc
pwrc xfer=16 addr=0 pat=cc33

vwrc xfer=8 addr=0 pat=c3
pwrc xfer=8 addr=0 pat=3c
```

All test sequence, continued.

test rop space

```
vw addr=200000 pat=3c3c
vwrc addr=200000 pat=3c3c

vw addr=200000 pat=c3c3
vwrc addr=200000 pat=c3c3

pw addr=200000 pat=5a5a
pwrc addr=200000 pat=5a5a

pw addr=200000 pat=a5a5
pwrc addr=200000 pat=a5a5

pw addr=200000 xfer=8 pat=55
pwrc addr=200000 xfer=8 pat=55

pw addr=200000 xfer=8 pat=aa
pwrc addr=200000 xfer=8 pat=aa
```

Default

Selecting `default` from the P2 Bus Interface menu executes the following sequence:

```
vwrc xfer=8 addr=30a001 pat=ff
vwrc
pwrc
pw addr=200000 pat=5a5a
pwrc addr=200000 pat=5a5a
sreg
perplane
```

VW *Xfer= Addr= PAT= pass= scope=[0,1]*

Selecting `VW` executes the *VME Write test.* This test writes to a location on the CG through the VME. You may specify the bus transfer size, the address, and the pattern to write.

Transfer types (*Xfer=*) may be 8-, 16-, or 32-bit transfers.

Setting the pass count at a very large number causes continuous writes.

Setting `scope=1` causes the test to write forever.

The default setting is

```
VW xfer=16 addr=100000 pat=55aa.
```

VR *Xfer= Addr= pass= scope=[0,1]*

Selecting VR executes the VME Read test. This test reads a specified location on the CG through the VME. You may specify the bus transfer size and the address. Transfer types (*Xfer=*) may be 8-, 16-, or 32-bit transfers.

For continuous reads, set the pass count to a very large number.

Setting `scope=1` causes the test to read forever.

The default setting is

```
VR xfer=16 addr=100000 .
```

VWRC *Xfer= Addr= PAT= Dec= pass= scope=[0,1]*

Selecting `VWRC` executes the *VME Write/Read/Compare Test.* You may specify the bus transfer size, address and pattern. The test writes, reads, and compares a pattern.

Transfer types (*Xfer=*) may be 8-, 16-, or 32-bit transfers. All transfers are done over the VME bus.

Setting *Dec=* takes the pattern and performs the test repeatedly while decrementing the pattern *Dec=* times. The test is completed when the unsigned pattern is greater than or equal to 0.

If an error has been detected, the results are passed through the RP to be placed into shared RAM. The results are then displayed.

The default setting is

```
VWRC xfer=16 addr=100000 pat=55aa.
```

If an errors occurs, a message similar to this is displayed:

```
***** vwrc error: xfer=16, &0x100000 exp(0x55aa), obs(0x0), xor(0x55aa).
***** number of errors = 1
```

PW *Xfer= Addr= PAT= pass= scope=[0,1]*

Selecting `PW` executes the *P2 Write test.* This test writes to a location on the CG through the P2 bus. You may specify the bus transfer size, address, and pattern to write.

Transfer types (*Xfer=*) may be 8-, 16-, or 32-bit transfers. All transfers are done by the PP over the P2 bus.

Setting `scope=1` causes the PP to write continuously.

The default setting is

```
PW xfer=16 addr=100000 pat=55aa.
```

PR *Xfer= Addr= pass= scope=[0,1]*

Selecting `PR` from the P2 menu executes the *P2 Read Test.* This test reads a specified location on the CG through the P2 bus. You may specify the bus transfer size and the address.

Transfer types (*Xfer=*) may be 8-, 16-, or 32-bit transfers.

Setting `scope=1` causes the PP to read continuously. The data read from the CG through the P2 is placed in the RP readback register. The RP then puts the data into shared RAM for the host to display.

The default setting is

```
PR xfer=16 addr=100000.
```

PWRC *Xfer= Addr= PAt= pass= scope=[0,1]*

Selecting `PWRC` executes the *P2 Write/Read/Compare Test.* You may specify the bus transfer size, address, and pattern. The test writes, reads, and compares a pattern. If an error has been detected, the results are passed through the RP and placed into shared RAM for host display.

Transfer types (*Xfer=*) may be 8-, 16-, or 32-bit transfers.

Setting `scope=1` causes the test to run continuously.

The default setting is

```
PWRC xfer=16 addr=100000 pat=aa55.
```

If the test encounters an error, a message similar to this may be displayed:

```
***** pwrc error:  xfer 16, addr(0x100000), exp(0xaa55) obs(0x0) xor(0xaa55)
```

Perplane *pass= scope=[0,1]*

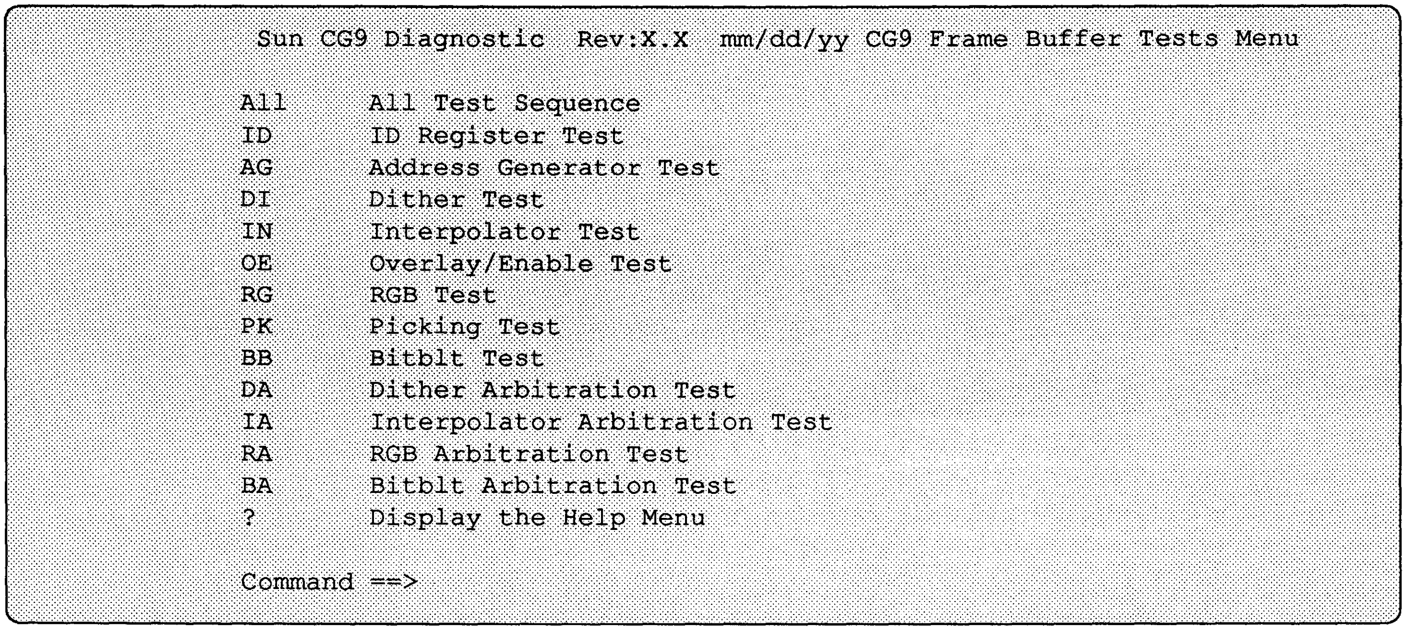
Selecting `Perplane` executes the *Per_Plane Register Test*, which writes, reads, and compares 5's, A's, 3's, and C's to the CG5 status register.

Sreg *pass= scope=*

Selecting `Sreg` executes the *FB Status Register Test.* This test writes, reads, and compares 5's, A's, 3's, and C's to the CG5 status register.

CG9 Frame Buffer Test Menu

An example of the CG9 Frame Buffer Test Menu follows.



```
                Sun CG9 Diagnostic  Rev:X.X  mm/dd/yy CG9 Frame Buffer Tests Menu

               All      All Test Sequence
               ID       ID Register Test
               AG       Address Generator Test
               DI       Dither Test
               IN       Interpolator Test
               OE       Overlay/Enable Test
               RG       RGB Test
               PK       Picking Test
               BB       Bitblt Test
               DA       Dither Arbitration Test
               IA       Interpolator Arbitration Test
               RA       RGB Arbitration Test
               BA       Bitblt Arbitration Test
               ?        Display the Help Menu

               Command ==>
```

The following text describes the CG9 Frame Buffer menu options.

A *pass=*

Selecting the *All Test Sequence* executes all tests displayed in the CG9 Test Menu. The *pass=* option specifies the number of times the test is executed.

ID *pass= scope=[0,1]*

The *ID Register Test* checks for the presence of the CG9 board.

AG *pass= scope=[0,1]*

The *Address Generator Test* generates a linear address for the entire screen. The linear address is read back as an address value. The address generator circuit is tested using the following:

- linear address
- (x, y) coordinate
- xce (step in x)
- yce (step in y)
- combination of xce and yce

The output of the AG Test is a linear address. The returned linear addresses are compared to expected values.

DI *pass= scope=[0,1]*

Selecting the *Dither Test* executes the following sequence:

- render the entire screen in a shaded 12-bit mode and enable dither

- □ render first polygon in a shaded 12-bit mode and enable dither
- □ render second polygon penetrating the first polygon in a shaded 12-bit mode and enable dither.
- □ render depth-cued vector penetrating the first polygon in a shaded 12-bit mode and enable dither.
- □ compute 32-bit check sum value and compare to expected value.

IN *pass= scope=[0,1]*
The *Interpolator Test* executes its test in the following order:

- □ render the entire screen and first polygon to a shaded 24-bit mode
- □ render second polygon penetrating the first polygon in a shaded 24-bit mode
- □ render depth-cued vector penetrating the first polygon in a shaded 24-bit mode
- □ compute 32 bit check sum value and compare to expected value.

OE *pass= scope=[0,1]*
The *Overlay/Enable Test* checks the enable and overlay planes with values of 0000, 5555, AAAA, and FFFF. These values are alternately written to the enable and overlay planes. Returning values are read back for comparison with expected values.

RG *pass= scope=[0,1]*
RGB Test executes these steps repetitively:

- □ paints the entire screen with a constant color
- □ renders smaller polygon pixels for comparison with expected value
- □ changes the color values for the background and small polygons
- □ writes to the frame buffer in burst mode then read back expected value

PK *pass= scope=[0,1]*
Selecting the *PK Picking* test executes the following sequence:

- □ initialize the z-buffer
- □ render first rectangle, all pixels visible. Tests picking flag, which should be set to 1.
- □ render second rectangle, all pixels invisible, behind first rectangle. Tests picking flag, which should be 0.
- □ render third rectangle, all pixels visible, partially in front of the first rectangle. Tests picking flag should be set to 1.
- □ render fourth rectangle, some pixels invisible, partially behind third rectangle. Tests picking flag should be set to 1.

BB *pass=*

Selecting the *BITBLT* test executes the following sequence:

- render the first span line width to 1152 pixels and use it as a seed
- paints the entire screen using BITBLT circuit.

Note that the maximum width that a BITBLT circuit can process is 1024 pixels, therefore, it is separated into two parts.

- entire screen pixels are read back and compared with expected values a textured pattern on the screen.

NOTES Each of the following four arbitration tests reserves the upper half of the screen for PP. PP renders the upper half of screen in a high-speed infinite loop, either reading back expected values of data, or computing a checksum. If an error is detected, testing is halted and notification is sent to the host through the vsr register.

The lower half of screen is reservered for the host. The host simply cycles through five different colors writing and reading each pixel in sequence.

If the parameter "overlay" is enabled, the host accesses overlay memory instead of RGB memory.

DA *pass= overlay=[0,1]*

The *Dither Arbitration* test executes the following sequence:

- paints the upper half of screen twice using different colors
- render the same pattern as cg9_dither_diag.bin, but only smaller
- compute the checksum and compare with expected value; if it passes, it increments the counter and writes it to the vsr register else trap.

IA *pass= overlay=[0,1]*

The *Interpolator Arbitration* test is identical to **DA** except that it has 24-bit mode dither disabled.

RA *pass= overlay=[0,1]*

The *RGB Arbitration* test performs the RGB test in the upper half of screen.

BA *pass= overlay=[0,1]*

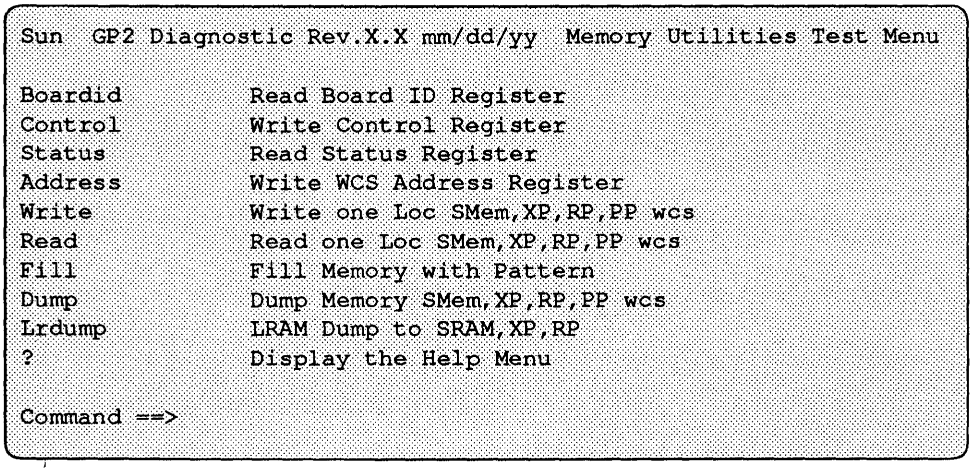
The *Bitblt Arbitration Test* executes the following steps:

- set the upper half of the screen using the same procedure as the **BB** test. Pixel values are read back and compared with expected values. If the test passes, it increments the counter and writes to the vsr register. Otherwise, it causes a trap.
- change color and repeat test

? Entering a question mark displays the help menu.

Memory Utilities Menu

The Memory Utility Menu appears as shown below.



```
Sun  GP2 Diagnostic Rev.X.X mm/dd/yy  Memory Utilities Test Menu

Boardid          Read Board ID Register
Control          Write Control Register
Status           Read Status Register
Address          Write WCS Address Register
Write            Write one Loc SMem,XP,RP,PP wcs
Read             Read one Loc SMem,XP,RP,PP wcs
Fill             Fill Memory with Pattern
Dump             Dump Memory SMem,XP,RP,PP wcs
Lrdump           LRAM Dump to SRAM,XP,RP
?                Display the Help Menu

Command ==>
```

The following table describes the optional arguments used for Memory Utility Menu selections.

Option	*Default*	*Range*	*Description*
pass	1	0-2147483647	pass count.
pattern	0x5a5a5a5a	0-0xffffffff	pattern to test with.
wcs	4	1-4	XP=1, RP=2, PP=3,SRAM=4.
loc	0	0-0x3ffff	starting location of testing RAM.
size	4	0-max size	size to fill or display selected RAM.
		0-0x10000	XP WCS size
		0-0x8000	RP WCS size
		0-1000	PP WCS size
		0-0x40000	SRAM size
lram	0	0-1	XP=0, RP=1.

The Help Menu lists the valid arguments for each test.

These utilities are used mostly during the initial stages of board development. They provide the ability to read one location, write one location, and read and write one location from any of the memories (XP WCS, RP WCS, PP WCS, and Shared RAM). The utilities are as follows.

B *pass=*

The ID*Read*Board command reads the GP2 ID register. The value read should be equal to 0xe.

C *pass= pattern=*

Selecting **C** writes to the control register. The *pattern* argument may be any two-digit hex value.

S *pass=*

Selecting **S** reads the status register.

A *pass= pattern=*

Selecting **A** writes data to one of the WCS (XP, RP, or PP) address registers.

W *loc= wcs=[1,2,3,4] size= pattern= pass= dmode=[0,1,2]*

Selecting **W** writes to one location of the specified RAM (Shared RAM, XP WCS, RP WCS, or PP WCS). For Shared RAM, only bytes, words, or long word writes can be used. Long words only are used for the writable control stores. When writing to the XP or PP WCS' s both 32-bit data words of the same location receive the same data. The default is to write to Shared RAM.

R *loc= wcs=[1,2,3,4] size= pass= dmode=[0,1,2]*

Selecting **R** reads one location from the specified RAM (Shared RAM, XP WCS, RP WCS, or PP WCS). When reading the Shared RAM, byte, word, or long words reads can be selected (default is long word). When reading the WCS, only long word reads occur. For the XP and PP, both 32-bit data registers are read. The address and data is printed out for the first 100 passes. After that, nothing is printed to allow for scope loops when a very large pass count is set. The default is to read Shared RAM.

F *loc= wcs=[1,2,3,4] size= pattern=*

Selecting **F** fills any number of locations with the same user-specified data pattern (the default data pattern = 0). The data is in long word format. The Fill command can fill the Shared RAM, XP, RP, and PP writable control stores. The default is to fill Shared RAM.

D *loc= wcs=[1,2,3,4] size=*

Selecting **D** executes the dump command. This command displays the data in either the Shared RAM or any of the Writable control stores. For Shared RAM, data is displayed in bytes on the screen. For the XP and PP WCS, data is displayed in long words in two 32-bit halves. For the RP, the display is in long words. The default is to dump Shared RAM.

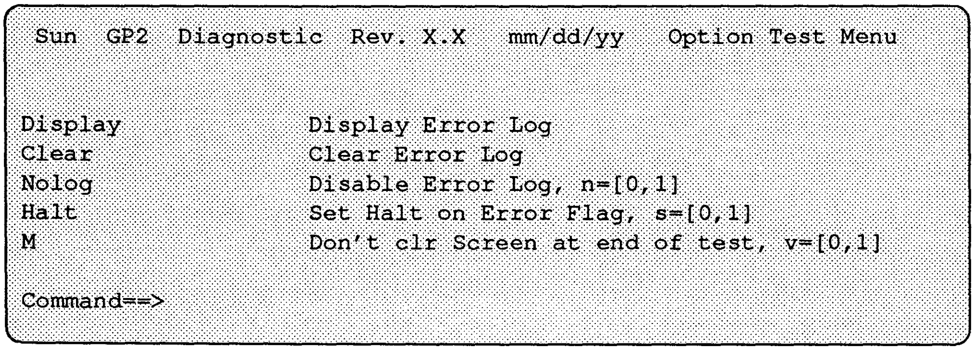
L *l=[0,1]*

Selecting **L** executes the Local RAM dump into a shared RAM routine, which dumps the either the XP's or RP's Local RAM, depending on which is selected. The data is put into shared RAM at location 0x40400 (location 0 if you look at the host using the dump command).

? Option **?** displays the Help Menu.

Option Menu

Entering **O** from the GP Diagnostic Main Menu brings up this option menu:



```
 Sun  GP2  Diagnostic  Rev. X.X   mm/dd/yy   Option Test Menu


Display               Display Error Log
Clear                 Clear Error Log
Nolog                 Disable Error Log, n=[0,1]
Halt                  Set Halt on Error Flag, s=[0,1]
M                     Don't clr Screen at end of test, v=[0,1]

Command==>
```

D

Enter **D** to *display the error log.*

C

Enter **C** to *clear the error log.*

N

Enter **N** to *disable the error log.*

Halt *s=*

Select **Halt** to stop error-message scrolling so that you can read a message. To continue, press [Return] and, if it is one of the host Shared RAM or WCS RAM tests, it continues. If the error message is from one of the micro-coded tests loaded into WCS then the next test is executed. The *s* argument specifies whether the halt-on-error message is to be on or off. The default is off. `s=0` means halt is off; `s=1` means halt is on. Currently, the first 20 error messages are saved in the error log. With the halt option turned off, you may look at the error log to find out what errors were detected.

M *v=*

Select **M** to view the messages that were printed during or after the test executed. The Exec usually clears the screen when the test finishes, which makes it difficult to see if the test passed or failed. This option, when turned on, prevents the screen from clearing. The *v* argument specifies either `view on` or `view off`. To turn view on, set `v = 1`; for view off, set `v = 0`. The default is for view to be off, which allows the diagnostic to run without operator intervention. If view is turned on, the diagnostic halts after each test. To continue, press [Return].

16.7. Memory Test Descriptions

The memory pattern tests in detail and their command line options are discussed below.

Addressing Test

AD *pass= offset= size= compl=[0,1,2] dmode=[0,1,2]*

The **AD** selection tests the specified block of memory using as data the low-order bits of the address of each location, or its complement. This test runs in byte, word, or long word modes, except for the WCS test, which is long word mode only.
Examples for each data mode follow. Note that the starting address "0x78440" is a virtual address.

Byte Mode

```
0x78440  40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
0x78450  50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
```

Word Mode

```
0x78440  84 40 84 42 84 44 84 46 84 48 84 4a 84 4c 84 4e
0x78450  84 50 84 52 84 54 84 56 84 58 84 5a 84 5c 84 5e
```

Long Word Mode

```
0x78440  00 07 84 40 00 07 84 44 00 07 84 48 00 07 84 4c
0x78450  00 07 84 50 00 07 84 54 00 07 84 58 00 07 84 5c
```

Use the *Address Pattern Test* to detect coupling faults.

The *pass=* argument sets the number of passes this test should execute before it returns to the `Test Menu.`

The *offset=* argument is the offset added to the base address of the memory being tested.

The *size* argument is the amount of RAM to be tested.

CAUTION **If the offset is not zero, the size has to be smaller so that the test does not try to test too much RAM.**

The *compl=* argument is used with this test to invert the physical address. This inverted address is the data that is written into the RAM chips. `comp=2` is the default and tests both non-inverted and inverted modes.

The *dmode=* argument selects byte, word, or long word mode write/reads, in this order.

To abort the test and return to the test Menu, press the `q` key.

Constant Pattern Test

`c` *pass= offset= size= pattern= dmode=[0,1,2]*

RAM, using the user-specified 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 runs in byte, word or long word modes, except when testing the WCS and SRAM from XP, RP, which is long word mode only.

An example screen dump follows, using the data pattern 0x12345678.

```
Byte Mode

0x78440  78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78
0x78450  78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78

Word Mode

0x78440  56 78 56 78 56 78 56 78 56 78 56 78 56 78 56 78
0x78450  56 78 56 78 56 78 56 78 56 78 56 78 56 78 56 78

Long Word Mode

0x78440  12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0x78450  12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
```

Use the constant pattern test to detect stuck bits at faults. The default pattern for testing is 0xa5a5a5a5, with the mode set to long word mode.

The *pass=* argument sets the number of passes this test should execute before it returns to the test menu.

The *offset=* argument is the offset address added to the base address of the memory that is tested.

The *size=* argument is the amount of RAM to be tested.

CAUTION **If the offset is not zero, the size has to be smaller so that the test does not try to test too much RAM.**

The *pattern=* argument is the pattern selected to test the RAM, and should be a 32-bit pattern.

The *dmode=* argument selects byte, word, and long word mode write/reads in this order.

To abort the test and return to the test menu, press the q key.

Checker Pattern Test

H *pass= offset= size= pattern=*

Entering **H** brings up the checker pattern test, which test the given RAM, using the data pattern you specify.

First, the data pattern is written into the RAM location, and then the data is inverted and written into the next RAM location. The RAM is completely filled in this manner. The RAM is then read back and compared with the patterns written. If the data read back does not match the original data pattern, an error is flagged. If there were no errors, two locations are written with the same data, followed by the inverted data pattern. The RAM is checked again, continuously putting a checker pattern into memory, until 20 locations with the same data pattern, followed by 20 locations of inverted data fill up the frame buffer. This test is executed only in long word mode. An example of a dump follows.

Pass 1

```
0x78440  00 00 00 00 ff ff ff ff 00 00 00 00 ff ff ff ff
```

Pass 2

```
0x78440  00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff
```

Pass 3

```
0x78440  00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff
0x78450  ff ff ff ff ff ff ff ff 00 00 00 00 00 00 00 00
0x78460  00 00 00 00 . . . . . . . . . . . . . . . . . .
```

Pass 20

```
0x78440  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x78450  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x78460  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x78470  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x78480  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x78490  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
0x784a0  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
0x784b0  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
0x784c0  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
0x784d0  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
```

Use the checker pattern test to detect stuck-at faults and to check the RAM sense lines and addressing problems. The testing default pattern is 0xa5a5a5a5 with the mode set to long word mode.

The *pass=* argument sets the number of passes that this test is to execute before it returns to the test menu.

The *offset=* argument is the offset added to the base address of the memory being tested.

The *size=* argument is the amount of RAM to be tested.

CAUTION **If the offset is not zero, the size has to be smaller so that the test does not try to test too much RAM.**

The *pattern=* argument is the pattern selected to test the RAM, which should be a 32-bit pattern.

To abort the test and return to the test menu, press the `q` key.

Uniqueness Addressing Test

U *pass= offset= size= incr= dmode=[0,1,2]*

U invokes the Address Uniqueness Test, which tests the specified block of memory using the sequence {incr, 2 * incr, 3 * incr, 4 * incr, ... } for the test data. This test can run in byte, word, and long word mode, except for the WCS, which is in long word mode. An example of a screen dump follows.

```
Byte Mode

0x78440  00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f

Word Mode

 0x78440  00 00 00 01 00 02 00 03 00 04 00 05 00 06 00 07

Long Word Mode

 0x78440  00 00 00 00 00 00 00 01 00 00 00 02 00 00 00 03
 0x78450  00 00 00 04 00 00 00 05 00 00 00 06 00 00 00 07
```

Use the unique pattern test to detect coupling faults.

The *pass=* argument specifies the number of times the test is to execute before it returns to the test menu.

The *offset=* argument is the offset added to the base address of the memory being tested.

The *size* argument is the amount of RAM to be tested.

CAUTION **If the offset is not zero, the size has to be smaller so that the test does not try to test too much RAM.**

The *incr* argument determines the increment value that is added to the data pattern written into the memory. The default increment value is 3. If no increment value is given, the memory is cleared.

The *dmode* argument selects byte, word, or long word mode write/reads in this order.

Mats (Modulo 3's) Test

M *pass= offset= size= pattern= dmode=[0,1,2]*

Selecting **M** performs the Mats test, which writes a sequence of pattern and ~pattern in a series of write/read scans as follows:

Loop 0 {pattern, ~pattern, ~pattern}

```
0x78440  33 33 33 33 CC CC CC CC CC CC CC CC 33 33 33 33
0x78450  CC CC CC CC CC CC CC CC 33 33 33 33 CC CC CC CC
```

Loop 1 {~pattern, pattern, ~pattern}

```
0x78440  CC CC CC CC 33 33 33 33 CC CC CC CC CC CC CC CC
0x78450  33 33 33 33 CC CC CC CC CC CC CC CC 33 33 33 33
```

Loop 2 {~pattern, ~pattern, pattern}

```
0x78440  CC CC CC CC CC CC CC CC 33 33 33 33 CC CC CC CC
0x78450  CC CC CC CC 33 33 33 33 CC CC CC CC CC CC CC CC
```

This test can be executed in byte, word, or long word mode. Use the Mats pattern test to detect pattern-sensitive faults.

The *pass=* argument sets the number of passes this test is to execute before it returns to the test menu.

The *offset=* argument is the offset added to the base address of the memory being tested.

The *size* argument is the amount of RAM to be tested.

CAUTION **If the offset is not zero, the size has to be smaller so that the test does not try to test too much RAM.**

The *pattern* argument is the pattern selected to test the RAM, which should be a 24-bit pattern.

The *dmode* argument selects byte, word, and long word mode write/reads, in this order.

To abort the test and return to the test Menu, press the `q` key.

Walking 1's/0's Test

`W` *pass= offset= size= compl=[0,1] dmode=[0,1,2]*

The Walking 1/0's test writes a 1 in a field of 0's in every bit location. This test starts with the data word equal to 1 and writes the data into the first memory location. It then shifts the data word one bit to the left, making the data equal to 0x2. Next it writes the new data into the next memory location. This action continues until all of the bits of the word (byte, word, long word) have had a "1" bit turned on.

The memory pattern then starts over with 0x1, and memory is filled in this manner. The memory is checked and any errors are displayed. The next loop (loop 2) starts the data 0x2 in the first location. This action continues for eight loops for byte mode, 16 loops for word mode, and 32 loops for long word. A 1 bit is now "on" in each cell of the RAM, with the other bits set to zero. The data can be inverted using the `compl` mode (the default is to invert the data). For the Local RAM, the data is always inverted.

An example screen dump (in byte mode) follows.

```
Loop 1

 0x78440  01 02 04 08 10 20 40 80 01 02 04 08 10 20 40 80
 0x78450  01 02 04 08 10 20 40 80 01 02 04 08 10 20 40 80

Loop 2

 0x78440  02 04 08 10 20 40 80 01 02 04 08 10 20 40 80 01
 0x78450  02 04 08 10 20 40 80 01 02 04 08 10 20 40 80 01

Loop 8

 0x78440  80 01 02 04 08 10 20 40 80 01 02 04 08 10 20 40
 0x78450  80 01 02 04 08 10 20 40 80 01 02 04 08 10 20 40
```

The *pass* argument sets the number of passes this test is to execute before it returns to the test menu.

The *offset=* argument is the offset added to the base address of the memory being tested.

The *size* argument is the amount of RAM to be tested.

CAUTION **If the offset is not zero, the size has to be smaller so that the test does not try to test too much RAM.**

The *dmode* argument selects byte, word, and long word mode write/reads, in this order.

To abort the test and return to the test menu, press the `q` key.

NTA Pattern Test

NTA *pass= offset= size=*

Selecting **NTA** 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 eight 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.

The *pass=* argument sets the number of passes this test is to execute before returning to the test menu.

The *offset=* argument is the offset added to the base address of the memory being tested.

The *size* argument is the amount of RAM to be tested.

CAUTION **If the offset is not zero, the size has to be smaller so that the test does not try to test too much RAM.**

To abort the test and return to the test menu, press the q key.

Random Pattern Test

R *pass= offset= size= seed= dmode=[0,1,2]*

The *Random Pattern Test* tests the specified block of memory using a sequence of random numbers generated from the specified seed. The random number generator is the same as that in the 'C' run-time libraries. First the block of memory is filled with data, the random sequence is reseeded, and the data is then read back and compared with data that was originally written. If the data read from an address location does not match the original data pattern, an error is flagged. This test can be executed in byte, word, or long word mode.

The *pass=* argument sets the number of passes this test should execute before it returns to the test menu.

The *offset=* argument is the offset added to the base address of the memory to be tested.

The *size=* argument is the amount of RAM to be tested.

CAUTION **If the offset is not zero, the size has to be smaller so that the test does not try to test too much RAM.**

The *dmode* argument selects byte, word, and long word mode write/reads, in this order.

To abort the test and return to the test menu, press the q key.

Triangle Pattern Test

`T` *pass= offset= size= dmode=[0,1,2]*

The *Triangle Pattern Test* does a write/read and compare with the following pattern:

```
0,1,3,7,f,1f,3f,7f,ff,.... .
```

An error message is displayed if an error occurs. This test can be executed in byte, word, or long word mode, except for the WCS, which is in long word mode. An example of a screen dump in long word mode follows:

```
0x78440  00 00 00 00 00 00 00 01 00 00 00 03 00 00 00 07
0x78450  00 00 00 0F 00 00 00 1F 00 00 00 3F 00 00 00 7F
0x78460  00 00 00 FF 00 00 01 FF 00 00 03 FF 00 00 07 FF
0x78470  00 00 0F FF 00 00 1F FF 00 00 3F FF 00 00 7F FF
0x78480  00 00 FF FF 00 01 FF FF 00 03 FF FF 00 07 FF FF
0x78490  00 0F FF FF 00 1F FF FF 00 3F FF FF 00 7F FF FF
0x784a0  00 FF FF FF 01 FF FF FF 03 FF FF FF 07 FF FF FF
0x784b0  0F FF FF FF 1F FF FF FF 3F FF FF FF 7F FF FF FF
0x784c0  FF FF FF FF 00 00 00 00 00 00 00 01 00 00 00 03
```

The *pass=* argument sets the number of passes this test should execute before it returns to the test menu.

The *offset=* argument is the offset added to the base address of the memory being tested.

The *size* argument is the amount of RAM to be tested.

CAUTION **If the offset is not zero, the size has to be smaller so that the test does not try to test too much RAM.**

The *dmode* argument selects byte, word, and long word mode write/reads in this order.

To abort the test and return to the test menu, press the `q` key.

LRAM, SRAM Bus test

`L` *pass= loop=[0,1]*

This test tests the ability of the address and data buses to switch from the Local RAM to the Shared RAM and back again.

This test writes pattern=0xAAAAAAAA to the Local RAM and then the next instruction writes pattern=0x55555555 to the Shared RAM. After all of the RAM 16K words have been written, the test reads and checks the Local RAM for correct data.

It then reads and checks the Shared RAM for correct data. The test posts any errors.

The *pass=* argument sets the number of passes this test executes before it returns to the test menu.

The *loop* argument loops on the error. For no *loop on error*, enter **`loop=0`**. For *loop on error*, enter **`loop=1`**.

XP Port 2, Port 3 Tests

These tests test the special hardware that allows the Accel or 7138 Weitek chip to get data in SF9010IU byte ordering, for 16 bits and 32-bit modes. The Weitek chip can get the bytes in the right order for 32 bits if on a 32-bit boundary but cannot do so when on a 16-bit boundary, which can be the case during GPCI commands. The special hardware fixes this problem. The sub-tests are described next.

Test 1
writes data patterns 0x12345678 0x9abcdef0 into shared RAM.
checks half word = 0x1234 from port 3, rd16, rd16 even boundary.
checks half word = 0x5678 from port 3, rd16, rd16 odd boundary.
checks half word = 0x9abc from port 3, rd16, rd16 even boundary.
checks half word = 0xdef0 from port 3, rd16, rd16 odd boundary.

Test 2
Uses same data in shared RAM.
checks half word even boundary = 0x1234, rd16 even boundary.
checks long word odd boundary = 0x56789abc, rd32 odd boundary.

Test 3
Using same data in Shared RAM,
checks half word odd boundary = 0x5678, rd16 odd boundary.
checks long word even boundary = 0x9abcdef0. rd32 even boundary.

Test 4
Fills RAM with with pattern 0x00000001, then adds 0x00010001 to the pattern for each successive location. Then checks half word on an even boundary, followed by the remaining long words on an odd boundary. This action continues for all of shared RAM.

Test 5
Using the pattern that was written in Test 4, performs fast loads (16 fast loads into 16 different registers) and checks the data in the registers. This action continues until all of the Shared RAM has been checked in this manner.

Any errors are reported. Error messages are found in the Error Messages section.

XP Port3 VME Test

This test checks the arbiter that processes reading from port2 (32 bit port) and VME writing and reading Shared RAM at the same time. First the XP writes a pattern of 17 words into Shared RAM. Then it reads this pattern into 17 registers and checks the data. While this is going on, the host is writing and reading to 0x1000 locations of Shared RAM over the VME bus. If an error is detected by either the XP or the host it is reported. The data written to Shared RAM is:

loc	XP Data
0x40400	01020304
0x40404	05060708
0x40408	090a0b0c
...	....
...	...

loc	Host Data
0x40450	0xcccccccc
...	....
0x50450	0xcccccccc

If an error is detected, first read the error message to obtain the failed address. Then go to the utility menu and dump the first 100 locations. Check the data visually. Usually the data was written correctly but the XP could have read the wrong data and mixed up the bytes. If this is the case, verify that the arbiter PAL s are correct. If they are not, change them.

XP, RP Arbiter Test

This test has ten sub-tests that check the arbitration between the XP and RP. It checks to make sure that the XP has first priority over the bus. When most of these tests execute, the XP should get first access to the bus, followed by the RP, allowing the RP to lag behind the XP. The following describes each sub-test.

Test 1: XP Store, RP Store Arbiter Test
The XP writes 16 words of data (0x33333333) to every other location, starting at loc 0x40400. The RP writes 16 words of data (0x9999999) to every other location, starting at loc 0x40404. This process interleaves the data between XP and RP. The data is then read into 16 registers and checked. This test can cause data to read incorrectly because the XP or RP obtained the other's data when the arbiter did not correctly lock out the processors.

Test 2: XP Slow Load, RP Slow Load Arbiter Test
The XP and RP read the data written by test 1. They read using a slow load instruction mix into 16 registers. The data is checked and errors are reported if detected. The slow load instruction sequences is a follows.

```
addr+ .rx, 2, .word
load .rc
addr+ .rx, 2, .word
load .ry
```

Test 3: XP Store, RP Slow load Arbiter Test
The XP stores 16 words of data at the same time as the RP reads data using the slow load instruction sequence. The data the RP reads was written during test 1.

Test 4: XP Store, RP Fast load Arbiter Test
The XP stores the same 16 words of data as Test 1. At the same time, the RP reads, using a fast load instruction sequence. The RP checks the 16 registers and reports any errors detected. The fast load sequence is as follows.

```
addr+ .rx, 2, .word
load rc; addr+ .rx, 2, .word
load rd; addr+ .rx, 2, .word
```

Test 5: XP Slow Load, RP Store Arbiter Test
The XP reads the data that was written in test 1 into 16 registers. The data is checked and if an error was detected it is reported. At the same time, the RP writes data into the Shared RAM.

Test 6: XP Fast Load, RP Store Arbiter Test
The XP reads the data, using fast load sequence as described in test 4. The data is checked and any errors detected are reported. The RP stores data into Shared RAM at the same time that the XP reads the data.

Test 7: XP Fast Load, RP Fast Load Arbiter Test
Both the XP and RP use the fast load sequence to read from the Shared RAM. Both display an error if it is detected.

Test 8: XP Slow Load, RP Fast Load Arbiter Test
The XP uses slow load reads while the RP simultaneously uses fast load reads from Shared RAM. The test reports any errors.

Test 9: XP, RP Address Arbiter Test
The XP stores a ''walking 1's'' pattern into Shared RAM. The address is then doubled each time until the end of the Shared RAM address space. At the same time, the RP is reading the data just written by the XP into Shared RAM. The RP is locked out of Shared RAM until the XP has written the data into it. The RP checks the data that was stored into registers and reports any errors that are detected.

Test 10: XP, RP Store NOP Load Arbiter Test
This test stores an address pattern into SRAM, starting at location 0x40400, then executes two `nops` to allow the RP access to SRAM. It then reads and checks the data and loops until all of SRAM is written and checked. The RP is doing the same thing at the same time.

Test 11: XP, RP 2 Store nop 2 Load Arbiter Test
This test checks the XP, RP arbiter to allow the RP to get control of the arbiter some of the time. It writes two words to SRAM with `nops` between them. It reads the two words from SRAM and checks them, looping 16 times. The RP is doing the same thing at the same time.

16.8. XP, RP Processor Tests

The following sections briefly describe the XP and RP 7136, 7137 processor tests. The syntax for these tests has already been described in the section *XP, RP Processor Test Menu.*

Hshake Test

This test tests the ability of the Accel or Weitek chip set to write to the last four long word locations of shared RAM. The first write is at location 0x7fff0 with the data pattern equal to 0x12345678. The next patterns are 0x87654321, 0xaaaa5555, and 0x5555aaaa.

The test briefly checks the addressing, the write signal and the data bus to the Shared RAM. If the arbiter between the XP, RP, and VME is really bad, the test fails. In order to test the Shared memory address bus more throughly, use the Addressing test in the Shared Memory Menu.

CHshake Test

The test tests the ability of a ''C'' program running in the XP to write to the last four locations of Shared RAM. The major difference between this and the Hshake Test is that the data for the first long word location is written in three separate bytes (lowest three bytes), and the upper byte just has one bit ''ored'' into it.

The data written is:

byte 1 = 0x34
byte 2 = 0x56
byte 3 = 0x78

The data for the last three long word locations is 0x55667788, 0x99aabbcc, and 0xddeeff00.

XP Branch Tests

This test performs a series of branch instructions to nearby locations. It also neutralizes some of these branches. If a branch goes to a wrong location, the test loops on itself (branch $). It also does a quick check of memory to see that the XP or RP can write and read one location of Shared RAM.

WCS Addressing Test

This test tests the address bus between the Weitek 7136 sequencer chip and the WCS (micro Code store), using the branch table:

address	branch address
0x0000	0xffff
0x0001	0x8000
0x0002	0x7000
0x0003	0x6000
0x0004	0x5000
0x0005	0x4000
0x0006	0x3000
0x0007	0x2000
0x0008	0x1000
0x0010	0x0800
0x0020	0x0700
0x0030	0x0600
0x0040	0x0500
0x0050	0x0400
0x0060	0x0300
0x0070	0x0200
0x0080	0x0100
0x0100	report good status
0x0200	0x0080
0x0300	0x0070
0x0400	0x0060
0x0500	0x0050
0x0600	0x0040
0x0700	0x0030
0x0800	0x0020
0x1000	0x0010
0x2000	0x0008
0x3000	0x0007
0x4000	0x0006
0x5000	0x0005
0x6000	0x0004
0x7000	0x0003
0x8000	0x0002
0xffff	0x0001

The RP code is similar to the XP code except that it has one less address line, meaning that the last address tested is 0x7fff.

XP Logical Test

This test performs the logical operations `xor, and, not register, clr, set.` It uses different patterns to test these operations in several combinations. The patterns used are ''0x33333333, 0x55555555'', ''0x66666666, 0xAAAAAAAA'', ''0xCCCCCCCC, 0x55555555'', and ''0x99999999, 0xaaaaaaaa''.

When an error is detected, the program branches to an error routine. This routine sets the error bit in the handshake status location, placing the failed address in status loc + 4; the expected value in status loc + 8; and the observed data in status loc + 12. The monitor reads these locations and displays the correct error message. The `error no` returned in the message is the sub-test number being executed. If an error is detected, the Weitek chips are probably at fault.

XP Register Tests

This test writes the register's number into each register, meaning that register R0 equals 0. Then the test, using subtracts, checks each register for its correct value, and posts an error if the value is incorrect.

XP Push Pop Tests

This test pushes 30 values on to the stack. The values pushed on the stack are as follows: 0, 1, 2, 4, 8, 0x10, 0x20, 0x40, 0x80, . . . 0x800000. The values are then popped back off the stack in descending order and checked. The test posts any errors.

XP SOB Test

This test tests the ''subtract one and branch'' instruction (`sob`). First a count of zero is tested, followed by a count of 1, count of 2, and then a count of 7. This is done by loading the count into a register, setting the count1 registers to 0, then executing the `sob` instruction. The count1 register is checked against the expected number of loops. The test posts any errors.

XP SOB Neutralize Test

This test is exactly the same as the SOB test described above, except that `revneut` and `ovneut` are also tested.

XP SHOB Test

This test is the same as the `sob neut` test (above) except that it uses the `shob` instruction.

XP Add, Sub Tests

This test tests the `add, addi, sub, subi, addil0, addshift, uadd, usub, neg, addr+, +addr` instructions. It adds or subtracts the contents of two registers and checks their results. The test posts any errors.

XP Call Tests

This test calls a subroutine and checks to see if it returned correctly. If it didn't, the test running in the GP2 hangs, and the host reports an error.

XP Condition Code Tests

This test tests overflow conditions on `signed/unsigned add/sub`. The test posts any errors.

XP Merge, Deposit Tests

This test tests the *deposit* and *merge* `(static, dynamic, immediate)` instructions. The deposit and merge instructions can be used as shift instructions, which is the condition that is tested here. A pattern is put into a register and a shift count is also put in the shift register while a length is put into the length register. Starting with a shift count of 0 and length of 32, it shifts the data and check it. Then it steps the shift amount and tests again, for shift amounts 0 to 32. Next, it changes the length to 1 and performs the test again. The test posts any errors.

XP Extract Tests

This test tests the `extract` instruction. This test is similar to the deposit and merge test. The difference is that the data is taken out of the pattern and put into a different register and checked. The test posts any errors.

XP Multiply Test

This test tests the `mpy` instruction. It first checks that every bit in the `.am` and `.al` registers can be written. It next performs these adds to these registers: 2+1, 3+1, 4+1+0, and 4+1+1. It next tries to multiply 1*1, 1 * -1, -1 * 1, and -1 * -1. The test posts any errors.

XP Divide Test

The divides that are performed are as follows:

most	least	divisor	quotient	remainder
0x00000000,	0x00020001,	0x00000004,	0x00008000,	0x00000001
0x00000000,	0x00000001,	0x00000002,	0x00000000,	0x00000001
0x00000001,	0x00000000,	0x00010000,	0x00010000,	0x00000000
0x00000000,	0xFFFFFFFF,	0xFFFFFFFF,	0x00000001,	0x00000000
0x00000000,	0x00000001,	0xFFFFFFFF,	0x00000000,	0x00000001
0x00000000,	0xFFFFFFFF,	0x00000001,	0xFFFFFFFF,	0x00000000
0x00000001,	0x00000002,	0x40000000,	0x00000004,	0x00000002
0x00000001,	0x00000007,	0x00000004,	0x40000001,	0x00000003

The test posts any errors.

XP Byte Alignment Test

This test tests the byte alignment for loads and stores. The test stores a pattern of 0x44332211 into a memory location, then reads and checks these locations in byte, word, tri-word, and long-word modes. For a byte, it gets the data 0x11, for word, 0x2211, for tri, 0x332211 and long word, 0x44332211. The test posts any errors.

This test and the two tests below check out a few locations of Shared RAM to allow proper function of the Floating point tests, which need Shared RAM to execute. They also test out the various instructions described in each test.

XP Byte Store Test

This test checks byte stores (at speed) and aligns. The test posts any errors.

XP Addrd Test

This test tests the `addrd` instruction with alignment. The data values written into memory are 0x44332211, 0x88776655, and 0xCCBBAA99. These data value are accessed in several ways with the `addrd` instruction in different alignment modes: byte, word, long word, quad word. The test posts any errors.

16.9. XP Floating Point Tests

The following sub-sections describe the 3132 floating-point tests that are downloaded into the XP WCS. The command line syntax for each of the following tests was described in the section "XP, RP Processor Test Menu".

XP Fload Tests

This tests the `store, fstore, load, fload` instructions to and from Shared RAM. It loads a value into a register in the integer unit then store it into shared RAM where the FPU loads and stores the same value. Then the 7137 loads the data and checks it. The test reports any errors. Possible failures could be 1) the FPU chip 2) the hardware logic the FPU uses to load and store data to Shared RAM.

XP Fabs Tests

This test checks the FPU `fabs` instruction. The three tests that it performs are listed below.

inst	*reg*	*data in*	*data out (exp data)*
`fabs`	.f0,.f1	0x00000000	0x00000000
`fabs`	.f1,.f0	0x4000A000	0x4000A000
`fabs`	.f31,.f31	0xC000B000	0x4000B000

The test reports any errors.

XP Float Tests

This test tests the `float` and `fix` instructions of the FPU chip. This test takes floating-point numbers and fixes or changes them to integer numbers. The following lists the registers and data to be converted and the data that it expects.

Table 16-1 *Integer Float Conversion*

reg	*data*	*exp data*
.f1, .f31	0x00000000,	0x00000000
.f31,.f1	0x00000002,	0x40000000
.f1, .f1	0xFFFFFFFE,	0xC0000000

Table 16-2 *Float Integer(fixr) Conversion*

inst	*reg*	*data*	*exp data*	*int number*
`fix`	.f1, .f31	0x00000000,	0x00000000	0.0
`fix`	.f31, .f1	0x40000000,	0x00000002	2.0
`fix`	.f1, .f1	0xC0000000,	0xFFFFFFFE	-2.0
`fix`	.f31, .f31	0xC0933333,	0xFFFFFFFB	-4.6
`fix`	.f1, .f1	0xC059999A,	0xFFFFFFFD	-3.4
`fix`	.f1, .f1	0x40933333,	0x00000005	4.6
`fix`	.f1, .f31	0x4059999A,	0x00000003	3.4

The test reports any errors.

XP Fadd Tests

`Fadd` tests the `fadd, fsub, fsubr` instructions of 3132 chip. This performs the instructions mentioned with various data mixtures. The test posts any errors.

XP Faddt Tests

`Fadd2` tests the `fadd to t register` instructions. It tests the 8-bit exponents, the 23-bit mantissa. The test reports any errors.

XP Fadd2 Tests

This test tests the `fadd with 2` instruction. The test reports any errors.

XP Fmac Tests

This test tests the `fmac, fmna, fmns` instructions of the 3132 chip. It uses various combinations of data with these instructions. The results are checked and an error is reported if found.

XP Fmul Tests

This test uses the `Fmul` instruction of 3132 chip. It performs simple multiplies with seven different sets of data. The data is listed below.

Table 16-3 *Fmul Test Data*

data in		*exp data*
0x00000000	0x00000000	0x00000000
0x400aaaaa	0x00000000	0x00000000
0x00000000	0x400aaaaa	0x00000000
0x400aaaaa	0xfloat 1	0x400aaaaa
0xc000000f	0x40000000	0xc080000f
0x40000000	0xc000000f	0xc080000f
float -2	float -4	float 8

The test reports any errors.

XP Flut Tests

This test tests the `flut` instruction of the 3132 chip. This is the floating-point divide instruction. This test uses data stored in shared RAM. It reads a value out of shared RAM then executes the `flut` instruction and checks the results against the data stored in the next location of shared RAM. This loop is performed a number of times. The test reports any errors.

XP F_cc Tests

This test tests all of the condition codes for 3132 chips using the following instructions: `fmac,fmna,fmns,fadd,fsub,fsubr,fabs,fix,float`. The test reports any errors. The expected data is stored in Shared RAM at loc 0x40414. Observed data is at loc 0x4041c.

XP Fndloop Tests

This test checks the `fndloop` instruction of the 3132 chip. The test reports any errors.

16.10. Shared RAM Comm

Handshake Communications

This section describes the handshake communications between the host CPU and the Shared RAM on the GP2 board.

The data block definitions of the memory locations for the data block and the bit assignments are shown below. The memory locations are the last ten 32-bit memory locations in Shared RAM (Hardware address 0x7ffd8 - 0x80000 utility dump offset = 0x3ffd8 to 0x3fffc).

Table 16-4 *The Data Block*

location	*Purpose*	*Size*
0x7ffd8	XP fifo wrote cnt no.	32 bits
0x7ffdc	RP fifo read count no.	32 bits
0x7ffe4	RP Address Register	32 bits
0x7ffe8	RP Expected Data	32 bits
0x7ffec	RP Observed Data	32 bits
0x7fff0	Status Register	32 bits
0x7fff4	XP Address Register	32 bits
0x7fff8	XP Expected Data	32 bits
0x7fffc	XP Observed Data	32 bits

The XPFIFO ''wrote cnt no.'' location is used by these tests. The FIFO random, FIFO constant, FIFO PP random tests all use this location for the XP to tell the host how many words were written to the XP fifo.

The RP FIFO ''read count no.'' locations are used by the same tests as the ''wrote cnt no.'' location (0x7ffd8). The difference is that the RP writes this count when it reads data out of the FIFO. These two locations should match; if they do not, the test posts any errors. In this way, the test checks that the XP wrote the correct amount of data into the FIFO.

Location 0x7ffe4 is used by the RP to store the failed address of the bad Local or Shared RAM location during one of the memory tests.

Location 0x7ffe8 is used by the RP to store the Expected data that was to be written into the memory location.

Location 0x7ffec is used by the RP to store the Observed data that was found in the RAM location during the failure.

The Status register is for talking between the host CPU and the XP,RP,PP processors. It can be written by any of the processors (host, XP, RP). Setting certain bits causes certain actions to be taken by either side. The bit definitions are as follows:

Table 16-5 *Status Register Bit Definitions*

Bits	*Purpose*
31	Pass Fail bit:
30	XP Completed:
29	RP fail bit:
28	RP complete:
27	Loop on error:
26-25	Data Mode:
24	Not Used:
23-17	Execute Test Number:
16-8	Number of Errors:
7-0	Test Number:

Bit 31 ''Pass/Fail'':

0 = XP test is still running or has passed.

1 = XP test has failed.

Get further information from rest of the status word and data block.

Bit 30 ''XP Completed'':

0 = XP test not complete.

1 = XP test completed.

If the test completed and the fail bit is set, the host reads the ''failed address loc'', ''expected data loc'', and the ''observed data loc'', then prints an error message.

Bit 29 ''RP fail bit'':

0 = RP test is still running or has passed.

1 = RP test has failed.

Get further information from rest of the status word and data block.

Bit 28 ''RP complete'':

0 = RP test not complete.

1 = RP test completed.

If the test completed and the fail bit is set, the host reads the ''failed address RP loc'', ''expected data RP loc'', and the ''observed data RP loc'', then prints an error message.

Bit 27 ''Loop on error'':

0 = no loop on error.

1 = loop if an error is detected.

If an error is detected, the XP, RP, PP must set the fail bit and test number of failure.

Bits 26 - 25 ''Data Mode'':

0 = Byte mode.

1 = word mode (16 bits).

2 = long word mode (32 bits).

Bit 24 ''Not Used'':

Bits 23 - 17 ''Execute Test Number'':

This byte sends to the XP, RP, WCS the test code number to execute.

Bits 16 - 8 ''Number of errors'':

This byte represents the number of errors that were detected during execution of the test that failed. Maximum number of errors is 256, and the number is in hex code. This could also be the number of the sub-test that failed.

Bits 7 - 0 ''Test number'':

This byte represents the test number presently being executed. This is used primarily as an index into the error message array to get the correct test name. It also signifies that the test has at least started to execute.

After the desired WCS has been down-loaded with the micro-code and the user selects a test from the menu, the host CPU's code puts the selected test number into the ''execute test number'' bits. The monitor running in WCS reads this number and executes the selected test. The micro-code's next step is to set the test number into ''Test Number'' position. If the test fails, the micro-code sets the fail bit and the complete bit, then returns to its monitor. The host CPU is continuously polling the handshake status location. When it sees the fail bit set it reads the test number and uses it to fetch the correct error message. The host CPU then gets the failed address, expected data, and received data information from other three handshake locations and displays the error message. If there are no errors, only the complete bit is set, and the host CPU returns to the test menu.

16.11. Error Messages

Error Message No. 1.

The following error message is displayed when an error is detected while testing the Shared RAM, WCS. The name of the test is displayed, followed by the address at which it failed. The expected data and observed data is then displayed. For memory testing an `xor` data value is displayed to help in determining the failing bit/s.

```
1 - Test name - Failed @(0x%x) exp(0x%x) obs(0x%x) xor(0x%x)
```

Error Message No. 2

The following error message is displayed when an error is detected while executing any of the eight down-loaded tests to the WCS's. These include XP, Floating Point, and RP processor tests. Also included is the XP, RP local RAM tests, the XP, RP Shared RAM test, and the XP, RP FIFO tests. The test name is displayed along with the address of the failing RAM (if a RAM test). If the test that failed is a processor test, the address is the address of the `bsr` instruction where the failure was detected.

```
2 - TestName - Failed @(0x%x) exp(0x%x) obs(0x%x) xor(0x%x) No.
of Errors %d0,noerrs
```

If the error was, for example, an error due to one of the processor tests reading the failed address, you could then use the 8032 simulator or disassembler to determine which test failed. The expected and observed data is displayed along with the ''xored'' value. The number of errors that were detected, up to 256, is displayed. Most of the tests that are down-loaded into the WCS's are set up to continue testing even if an error is detected. This is true for all RAM tests. The processor tests posts status then goes to the next test.

Shared RAM Error Messages

The following error messages are extra messages to help the technician understand what the shared RAM or Port test was doing when it hung up. The status byte number is the value returned in the XP status byte portion of the GP2 Status register.

`Hung Initializing Shared RAM,`	status byte = 0
`Checking data in SRAM,`	status byte = 1
`Write, Write, Read Test 1,`	status byte = 2
`Write, Write, Read Test 2,`	status byte = 3
`Checking Half Word Port 3, Even,`	status byte = 4
`Checking Half Word Port 3, ODD,`	status byte = 5
`Checking Long Word Port 2, ODD,`	status byte = 6
`Checking Long Word Port 2, EVEN,`	status byte = 7
`Checking All SRAM From Port 2,`	status byte = 8
`Checking Using Fast Loads,`	status byte = 9
`Unknown Status,`	status byte = 10

Arbiter Error Messages

The following error messages are extra messages to help the technician understand what the arbiter test was doing when it hung up. The status byte number is the value returned in the XP status byte portion of the GP2 Status register.

`Hung Initializing Shared RAM,`	status byte = 0
`Hung Using Fast loads from SRAM,`	status byte = 1
`Hung During compare,`	status byte = 2
`Hung Using Slow loads from SRAM,`	status byte = 3
`Hung Using Slow stores to SRAM,`	status byte = 4
`Hung Using Store, Loads to SRAM,`	status byte = 5
`Hung Using 2 Stores, 2 Loads to SRAM,`	status byte = 6
`Unknown Status,`	status byte = 7
	status byte = 40

The messages shown above are self explanatory after reading the arbiter test descriptions in the Memory Test Descriptions section.

FIFO Error Messages

The following error messages are extra messages to help the technician understand what the FIFO's were doing when one or both processors are hung up. Generally when one hangs up, both of them are hung. The status byte number is the value returned in the XP, and RP status byte portion of the GP2 Status register.

`In wait loop,`	RP status byte = 0
`Fifo not empty,`	XP status byte = 1
`Hung writing to Fifo,`	XP or RP status byte = 2
`Fifo Empty,`	XP status byte = 3
`Fifo Half Full flag wrong,`	XP status byte = 4
`Fifo Full flag wrong,`	XP status byte = 5
`RP can not Read PP read back reg,`	RP status byte = 6
`XP Waiting for RP to empty FIFO,`	XP status byte = 7
`RP Fifo not Hung,`	RP status byte = 8
`RP Hung reading XP Fifo,`	RP status byte = 9
`PP Fifo Not Empty,`	RP status byte = 10
`Fifo Bad status,`	status byte = any thing else

"In wait loop" messages are displayed when the RP is waiting for the XP to put the first word into the FIFO.

"Fifo not empty" messages are displayed when the XP is testing the FIFO empty flag right after reset. This flag bit should be zero.

"Hung writing to Fifo" messages are displayed when the XP has filled up the FIFO and can't write any more data to it until the RP reads one word from the FIFO. This message usually means something is wrong with the RP, or that the Arbiter between Shared RAM, XP FIFO, and RP Shared RAM is bad. Check the PAL s.

"Fifo Empty" messages are displayed by the XP after it has finished writing the FIFO and just before the RP has time to read all of the data out of the FIFO. This means that the data was never correctly written. The flag register is bad, or the RP read the fifo too soon.

The ''Fifo Half Full flag wrong'' message has the same meaning as the previous message except the Half Full flag is checked.

The ''Fifo Full flag wrong'' message has the same meaning as the previous message except that the Full flag is checked.

The ''RP can not Read PP read back reg'' message is displayed when the RP is hung trying to read the PP read-back register. This is caused because the RP did not write the PP FIFO, or the PP was unable to read the FIFO and write the read-back register.

''XP Waiting for RP to empty FIFO'' messages are displayed when the XP never sees that the FIFO Empty flag goes empty.

The ''RP Fifo not Hung'' message is displayed during the PP FIFO Over Full test. The RP writes 513 words to the PP FIFO, which should cause the RP to lock up. If it doesn't, an error has occurred.

''RP Hung reading XP Fifo'' messages are displayed when the XP FIFO is empty when it should not be.

''Fifo Bad status'' messages are displayed when ''garbage'' gets into the XP or RP status byte of the GP2 status register.

PP Error Messages

Here are the error messages for the PP core menu:

Example Trap Test Error Message

```
*****PP trap: exp(pc=0x1) rcv(pc=0x0).
test done.
```

If an error occurs, the PP trap test does four consecutive traps but only prints the first error.

Example VSR Test Error Messages

```
*****PP vsr: exp( 55), rcv( 00)
*****PP vsr: exp( aa), rcv( 00)
*****PP vsr: exp'( 33), rcv( 00)
*****PP vsr: exp( cc), rcv( 00)
test done.
```

If an error occurs, the `vsr` test prints out the expected and received data that was read from the VME PP status register.

Example RLD Test Error Message

```
*****PP rld: error
```

The rld test loads the branch register in the IDT bit slice machine. The test first loads 5's, then A's. It executes

```
rld ext; push; inc r1; rfct; sequence;
```

Example Sequencer Test Error Message

```
*****PP seq: exp(001) rcv(000)
test done.
```

If an error occurs, the sequencer test stops at the first errant expected location.

Example JP Test Error Message

```
***** jp error: pp halted at location 0x0.
```

If an error occurs, the `jp` test traps at a bad location if an illegal jump or no jump has occurred.

Example CC Test Error message

```
*****PP cc error: invalid condition code.
```

Example JS Test Error Message

```
***** js error: pp halted at location 0x0.
```

If an error occurs, the `js` test traps at a bad location if an illegal jump or no jump has occurred.

Example BS Test Error Message

```
*****PP bs error: location exp(0xaa4) rcv(0x0),
     branch taken =
```

If an error occurs, the `bs` test traps at a bad location if an illegal branch jump has occurred. If the location trapped on is not 0xaa4, an error has occurred.

Example Microslice Test Error Message:

```
*****PP mslice error: bad alu.
```

If an error occurs, the `mslice` test prints the message shown above.

Example RP/PP FIFO Test Error Message

```
WARNING RP Down load failed:
addr 0 exp 6400010 obs fffffff
*****PP fifo error: bad data.
```

The `rp/pp` test can fail if the microcode sent to the RP did not occur. If this happens, a WARNING message is displayed. The warning message prints the address of the RP WCS, the expected microword, and the observed microword. If the download to the RP was good, and an error occurs, the `rp/pp` test prints a ‘‘bad data’’ message.

```
WARNING RP Down load failed:
addr 0 exp 6400010 obs fffffff
*****PP fifo error: bad data.
total errors = 13
```

PP Hardware Menu Error Messages

Example dbwalk Error Message

```
*****PP dbwalk error.
test done.
```

If an error occurs, the dbwalk test prints the message shown above.

Example dbrw Test Error Message

```
dbrw error.
```

If an error occurs, the dbrw test prints the message shown above.

Example All Test Error Messages

```
***** PP bar error: fault detected.

***** PP bar error: ld x bit 0.
***** PP bar error: ld x bit 1.
***** PP bar error: ld x bit 2.
***** PP bar error: ripple test failed.
test done.
```

If an error occurs while running the all test, the barrel shifter test prints a fault detected message. The test must then be manually executed to get a more descriptive message. When manually executed, the test tells you whether the error occurred during a ld x, ad x, or ad y at width 1024 or 1152. The failing bit is reported.

An extra message on the ripple test is reported if the ripple test failed. This test loads the address generator in this fashion: Load x with 5's, ad x with A's, then increment the address generator. The address generator output should then be 0.

Example Offset Generator Test Error Message

```
*****PP off error.
```

If an error occurs, the offset generator test prints the message shown above.

Example ALU Noise Test Error Message

```
***** NOise error: bad cmp.
```

If an error occurs, the ALU noise test prints the message shown above.

Example IDT Noise Test Error Message

```
dt noise error: bad sub.
```

The difference between the ALU noise test and the IDT noise test is that the ALU noise test checks to see if the cmp instruction is destroyed. The IDT noise test then executes a sub instruction check to see if the status is 0.

The IDT noise test was implemented due to an outside IDT bit slice user noise problem. These tests are necessary for older revision level ALU s.

CG5 Frame Buffer Menu Error Messages

Example PP ival Test Error Message

```
*****PP ival error.
```

If an error occurs, the `ival` test prints the message shown above.

Example fby Test and fav Condition Code Test Error Message

```
*****PP fby error.
```

If an error occurs, the fby test prints the message shown above.

Example cfn Test Error Message

```
casefn test:
        *****PP cfn error.
```

If an error occurs, the `cfn` test prints the message shown above.

Example cfi Test Error Message

```
Casefi test:
        WARNING RP Down load failed:
        addr 0 exp 6400010 obs ffffffff
```

The `cfi` test may fail if the microcode sent to the RP did not occur. If this happens, a WARNING message is displayed. The warning message prints the address of failed RP WCS, the expected microword, and the observed microword. If the down-load to the RP was good, and an error occurs, the `cfi` test prints a bad data message.

Example Transfer Test Error Message

```
transfer 8/16/32 test:
        *****PP xfer error.
```

If an error occurs, the xfer test prints the message shown above.

Example Status Register Test Error Message

```
status register test:
        *****PP sreg error.
```

If an error occurs, the `sreg` test prints the message shown above.

Example per plane Test Error Message

```
per-plane register test:
        *****PP perplane error.
```

If an error occurs, the `per plane` test prints the message shown above.

Example Raster Op Chip Test Error Message

```
Raster OP chip test:
        *****PP ropc error.
```

If an error occurs, the `ropc` test prints the message shown above.

Example Color Map Test Error Message

```
color map test:
        display red, green, blue
        *****PP cmap error.
```

If an error occurs, the `color map` test prints the message shown above.

Example Vector Pattern Test Error Message

```
vector pattern test:
frame buffer add checksum = 0xbbf33b00, xor checksum = 0xa2200.
        ***** PP vector test: add checksum obs(0xbbf33b00),
exp(0xa93cfd90).

***** PP vector test: xor checksum obs(0xa2200), exp(0x1000000).
       test done.
```

If an error occurs, the vector pattern test prints the message shown above. The vector pattern test does a checksum on the display.

Example All Test Error Messages

```
CG Arbitration test:    pass no. 1
      pp = control, pixel, and word space.
      vme = control, pixel, and word space.
      *****cgarb error: VME=5 cpw, P2=5 cpw, fault detection
test done, errors = 10000

CG Arbitration test:    pass no. 1
      *****cgarb error: VME=5 cpw, P2=5 cpw, P2 statreg = 0x0
```

If an error occurs while running `all` test, `cg arbitration` tests print a "fault detected" message. The test could then be manually executed to get the address which failed. If the message

```
P2 statreg = 0x0
```

appears, an error occurred doing transfers over the P2 bus. Otherwise, the error occurred through the VME bus and the actual frame buffer address offset including the expected and observed data is displayed.

CG9 Frame Buffer Menu Error Messages

The format of the error message for the first eight commands is:

```
***ERROR: cg9 testname Failed
vsr status:xx
```

where *testname* is the name of the failing test, and *xx* is the hex value returned by the microcode. Possible values include:

```
FD - microcode is hung
01 - microcode found error
```

The format of the error message for the four arbitration commands is:

```
*** ERROR: cg9 testname arbitration failed (bus)
```

where *testname* is the name of the failing test, and *bus* is the name of the bus path that detected the error—either VME or P2.

Z Buffer Menu Error Messages

Example czn Test Error Message

```
casezn test:
        *****PP czn error.
```

If an error occurs, the czn test prints the message shown above.

Example Z Buffer Location Test Error Message

```
Z Buffer location Test:
     z addr=0x0, pattern=0x5555
     WARNING RP Down load failed:
     addr 0 exp 6400010 obs ffffffff
     ***** zl error: addr(0x0), exp(0x0), obs(0x0), xor(0x0)
```

The zl test can fail if the microcode sent to the RP did not occur. If this happens, a WARNING message is displayed. The warning message prints the address of the failed RP WCS, the expected microword, and the observed microword. If the down-load to the RP was good and an error occurs, the zl test prints the address, expected value, observed value and the xor of the two values.

Example Z Buffer Constant Test Error Message

```
z-buffer constant test:
        *****PP zbc error.
```

If an error occurs, the zbc test prints the message shown above.

Example Z Buffer All Test Error Message

```
Z Buffer Address Test:
    WARNING RP Down load failed:
    addr 0 exp 6400010 obs ffffffff
    addr(0x0), exp(0x0) obs(0x1) xor(0x1)
    ***** PP za error: addr(0x0), exp(0x0) obs(0x1) xor(0x1)
```

If an error occurs while running the all test, the za test prints a ''fault detected'' message. The test must then be manually executed to get a more descriptive message. The za test can fail if the microcode sent to the RP did not occur. If this happens, a WARNING message is displayed. The warning message prints the address of the failed RP WCS, microword, and the observed microword. If the down-load to the RP was good, and an error occurs, the za test prints the address, expected value, observed value and the xor of the two values.

Example Z Buffer Retention Test Error Message

```
Z Buffer Address Retention Test:
   WARNING RP Down load failed:
   addr 0 exp 6400010 obs ffffffff
   WARNING RP Down load failed:
   addr 0 exp 6400010 obs ffffffff
   addr(0x0), exp(0x0) obs(0x1) xor(0x1)
   ***** PP zr error: addr(0x0), exp(0x0) obs(0x1) xor(0x1)
```

The zr test can fail if the microcode sent to the RP did not occur. If this happens, a WARNING message is displayed. The warning message prints the address of the failed RP WCS, the expected microword, and the observed microword. If the download to the RP was good and an error occurs, the zr test prints the address, expected value, observed value and the xor of the two values.

Example zt Test Error Message

```
Z Buffer Hidden Surface Removal Depth Comparison Test:
        *****PP zt error.
```

If an error occurs, the zt test prints the message shown above.

Example s1 Test Error Message

```
Z Buffer Conditional Shade 16 Test:
        *****PP s1 error.
```

If an error occurs, the s1 test prints the message shown above.

Example s3 Test Error Message

```
Z Buffer Conditional Shade 8 Test:
        *****PP s3 error.
```

If an error occurs, the s3 test prints the message shown above.

16.12. Default Test Sequence

The tests to be executed when the default test is selected from the Main Menu are as follows:

Table 16-6 *Default Tests*

Test	*Menu*
Address Test	Shared Memory Test Menu
Mats Test	Shared Memory Test Menu
Walking 1's Test	Shared Memory Test Menu
Random Test	Shared Memory Test Menu
XP Address Test	WCS Test Menu
XP Mats Test	WCS Test Menu
XP Walking 1's Test	WCS Test Menu
XP Random Test	WCS Test Menu
RP Address Test	WCS Test Menu
RP Mats Test	WCS Test Menu
RP Walking 1's Test	WCS Test Menu
RP Random Test	WCS Test Menu
PP Address Test	WCS Test Menu
PP Mats Test	WCS Test Menu
PP Walking 1's Test	WCS Test Menu
PP Random Test	WCS Test Menu
XP Branch Test	XP, RP Processor Tests Menu
XP SOB Test	XP, RP Processor Tests Menu
XP Logical Test	XP, RP Processor Tests Menu
XP Memory Test	XP, RP Processor Tests Menu
RP Branch Test	XP, RP Processor Tests Menu
RP SOB Test	XP, RP Processor Tests Menu
RP Logical Test	XP, RP Processor Tests Menu

Table 16-6 *Default Tests— Continued*

Test	*Menu*
RP Memory Test	XP, RP Processor Tests Menu
XP Fload Test	XP, RP Processor Tests Menu
XP fmpy Test	XP, RP Processor Tests Menu
XP Flut Test	XP, RP Processor Tests Menu
XP F_cc Test	XP, RP Processor Tests Menu
XP SRAM Address Test	XP, RP Shared Memory Test Menu
XP SRAM Constant Test	XP, RP Shared Memory Test Menu
XP SRAM Random Test	XP, RP Shared Memory Test Menu
RP SRAM Address Test	XP, RP Shared Memory Test Menu
RP SRAM Constant Test	XP, RP Shared Memory Test Menu
RP SRAM Random Test	XP, RP Shared Memory Test Menu
XP Port 2, Port 3, Tests	XP, RP Shared Memory Test Menu
XP Port3 VME Test	XP, RP Shared Memory Test Menu
XP, RP, VME Arbiter Test	XP, RP Shared Memory Test Menu
XP Address Test	XP, RP LRAM Test Menu
XP Mats Test	XP, RP LRAM Test Menu
XP Walking 1's Test	XP, RP LRAM Test Menu
XP Random Test	XP, RP LRAM Test Menu
RP Address Test	XP, RP LRAM Test Menu
RP Mats Test	XP, RP LRAM Test Menu
RP Walking 1's Test	XP, RP LRAM Test Menu
RP Random Test	XP, RP LRAM Test Menu
XP, RP Fifo Over Full Write Test	Fifo Tests Menu
XP, RP Fifo 512 Word Read Back Test	Fifo Tests Menu
RP, PP Fifo Over Full Write Test	Fifo Tests Menu
Fifo XP, RP, PP Random Test	Fifo Tests Menu
All Pixel Processor related tests.	

16.13. Glossary

CPU	Central Processing Unit
FIFO	First In First Out
FPU	Floating Point Unit
IPU	Integer Processor Unit
LRAM	Local RAM
PAL	Programmed Array Logic chip
PP	Pixel Processor
PSU	Program Sequencer Unit
RP	Rendering Processor
SRAM	Shared RAM
WCS	Writable Control Store
XP	Transform Processor

17

TAAC-1 Applications Accelerator Diagnostic

17

TAAC-1 Applications Accelerator Diagnostic

17.1. Introduction to the TAAC-1

The TAAC-1 Applications Accelerator is designed to focus on the computationally intense and the display portions of applications involving spatial and geometric data. Dedication to the application task leaves the operating system, user interface, data base management, multiprocessing, and networking tasks to the Sun-3 or Sun-4 (SPARC) system, resulting in higher performance.

17.2. What This Chapter Contains

After an overview of the TAAC-1 Applications Accelerator and Diagnostic, this chapter describes the Main Menu and each Submenu. Menu selections and optional arguments are listed, along with brief descriptions. The end of the chapter contains more detailed test descriptions, along with error message and protocol information.

17.3. TAAC-1 Functional Overview

The objective of the TAAC-1 Diagnostic is to ensure that the TAAC-1 Accelerator Board works correctly. The TAAC-1 Diagnostic covers these functional areas of the TAAC-1 Board:

VME Interface

The VMEbus interface consists of two parts - a control register in VME address space and the interface to the local bus on the TAAC-1. The sections of the interface are as follows:

- Control Register: This is known as the slave mode register. It contains the TAAC processor control bits, image/data memory access mode bits and memory control bits. The memory control bits select which type of TAAC memory is being mapped into host virtual memory space.
- Local Bus: This is the major internal bus on the TAAC-1. It provides the connection between the processor, memory and the host interface.
- Data/Image Memory: This is 8 megabytes of memory organized as two million 32-bit words. In this chapter, data/image memory is referenced either as VRAM or DRAM.
- Program Memory: The processor on the TAAC-1 is controlled by the programs stored in the program memory. In this chapter instruction memory is referenced either as program memory or MCRAM.
- Scratchpad/Stack Memory: The TAAC-1 has 16 Kbytes of scratchpad/stack memory interfaced with the local bus. The data in the scratchpad/stack memory is available to the processor in one clock cycle. The scratchpad

memory stores frequently used variables and small data sets. In this chapter scratchpad/stack memory is referred to as SRAM.

- Control Register Memory: This area of memory contains all the TAAC-1 control registers that are addressable from the local bus. In this chapter, register space will be known as registers, or REG.
- Processor: The processor consists of multiple arithmetic/computational units connected by multiple buses. The processor has ports to the local bus and to the serial port of the data/image memory through the vector buses. There is a private bus from the program memory to the processor. The TAAC-1 processor contains two registered integer ALUs, a 32x32 multiplier/accumulator, a single/double precision floating point multiplier/ALU/accumulator, a 32 bit barrel shifter, an 8Kx32 bit PROM and 8Kx32 bit RAM look-up table, and a 16 bit microsequencer.
- RC and RD ALU s: The RC and RD are integer arithmetic logic units (ALUs), each containing a 64-register file. The two ALU s operate independently and simultaneously. The RC is generally used for data operations and the RD is generally used for address calculation.
- Multiplier/Accumulator: The Multiplier/Accumulator performs signed and unsigned multiplication on two 32-bit numbers and either stores it into or adds into a 64-bit accumulation register.
- Floating Point Processor: The Floating Point Processor performs floating point arithmetic operations: addition, subtraction, multiplication, comparison and conversion between floats and integers. The floating point processor contains an independent multiplier and ALU, which operate simultaneously.
- Barrel Shifter: The barrel shifter performs left or right shifts or rotations. The barrel shifter input is written to the BR register. The number of bits to shift can be a constant in an instruction word, or from a variable loaded into the count register.
- Look-up RAM: The Look-up RAM is an 8K by 32-bit look-up table that can be loaded by the processor.
- Look-up PROM: The Look-up PROM has two parts: an 8K by 8-bit exponent look-up PROM and an 8K by 23-bit mantissa Look-up PROM. These tables provide both floating-point and integer lookups.

Buses and Data Paths

The ability to independently select the sources and destinations for the six data buses in each program instruction gives the TAAC-1 much of its flexibility and power. The purpose of each bus is summarized in the following paragraphs.

The E-bus serves as the data path for memory I/O and access with the local bus. The data read and data write registers serve as an interface between the local bus and the processor. The E-bus provides a data path to all memory types for all buses and processing elements on them.

The A and B buses serve as operand source buses, providing inputs to the two registered integer ALUs, the integer multiplier, and the floating point multiplier/ALU.

The C-bus serves as the direct output bus for all the arithmetic units except the RD.

The D-bus is intended to serve as the memory address and control bus since it sources the two memory address registers (AC, AI) and the control registers (AM, MO).

The F-bus connects the sequencer to the D-bus for calculated input values and to the instruction data field for compiler or linker generated input values.

Registers

There are two memory address registers as destinations from the D-bus. These are known as the Address Immediate (AI) register and the Address Counter (AC) register.

The AI is used as a normal address register for addressing all the memory types and registers on the local bus.

The AC register provides 1-D, 2-D, and 3-D addressing mode support, as well as single pixel stepping in all modes.

The Access Processor Mode (AM) register controls the modes for the data/image memory access through the address registers and the vector ports.

The Miscellaneous Mode (MO) register controls modes for the vector ports, RC and RD ALU s, the floating point processor, the multiplier aad the look-up table PROM.

Vector Ports

The image/data memory is divided into 2 distinct banks: A and B. A high-speed port is dedicated to each bank so the display controller access and processor access can occur simultaneously with different banks. Bank A is accessed through vector port A, and bank B is accessed through vector port B.

Access Processor

When the processor becomes local bus master following arbitration, it controls the local bus through the Access Processor (AP). The AP consists of the START lower case acronyms address immediate register (AI), the address register/counter (AC), an address readback register, the address mapping logic circuit, an I/O pipeline register, a row/column address comparator, data registers DR and DW, a Z-data comparator, and the AP mode register (AM).

Video Output

The video output portion of the TAAC-1 receives digital pixel data from the data/image memory by the serial port and the display bus. The video data is separated into four 8-bit channels:

Channel 0 (also referred to as the RED channel) using bits 0-7.
Channel 1 (also referred to as the GREEN channel) using bits 8-15.
Channel 2 (also referred to as the BLUE channel) using bits 16-23.
Channel 3 (also referred to as the ALPHA channel) using bits 24-31.

17.4. Hardware Requirements

In order to use the TAAC-1 Diagnostic, the system being tested must have:

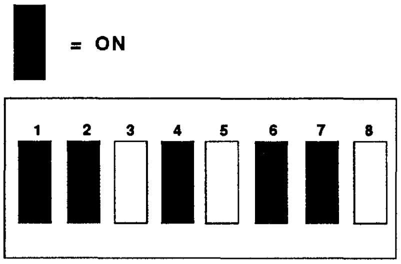
- □ A Sun-3/160, 3/260, 4/110, 4/260, 3/4xx or SPARCsystem330 CPU.
- □ Keyboard and Mouse.
- □ A color monitor.
- □ CG3 or CG5(Color Graphics boards).
- □ TAAC-1 board set (DFB and POP).
- □ An 8Mbyte memory board.
- □ A boot device (local disk, local tape, or remote disk over Ethernet).

17.5. Set-Up

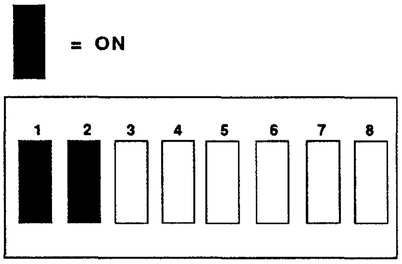
The current release of the software must first be installed on the system under test. This includes libraries for the TAAC, device driver, configuration files, etc. To properly install the software follow the instructions in *TAAC-1 Applications Accelerator: Software Installation Guide.*

The VMEbus address dip switches on the TAAC-1 processor board must also be set. (NOTE: Switch 1 is the nearest one with respect to the edge of the board).

For Sun-3/1xx, Sun-3/2xx, Sun-4/2xx, Sun-3/4xx, or SPARCsystem 330, the proper dip switch settings are as follows:



For Sun-4/1xx, the proper dip switch settings are as follows:



If the TAAC-1 Software has been installed properly and the DIP switches are set properly, this message should scroll by during the SunOS boot-up.

```
device     taac0 at vme32d32 ? csr 0x28000000
```

For the Sun-4/1xx, this message will scroll by during SunOS boot-up.

```
device     taac0 at vme32d32 ? csr 0xF8000000
```

The backplane jumpers must also be set properly. For the Sun-3/1xx, Sun-3/2xx, Sun-3/4xx, Sun-4/2xx and SPARCsystem 330, backplane locations PX03 and PX04 are both set to IN. The TAAC-1 board must be placed into slot 10.

17.6. Description Of The Diagnostic

The TAAC-1 Diagnostic tests each functional section of the TAAC-1 board. These tests can be run individually or in a continuous sequence.

The user interface is menu driven. A menu is displayed and either tests or submenus are selected from that menu. Each test has several parameters or options that may be selected from the command line.

Main Menu

The Main Menu has ten options and allows access to the submenus. It includes the default test commands and commands to adjust test parameters. These options are described following the display of the Main Menu, below.

```
TAAC Diagnostic  Rev. x.x  00/00/00  TAAC-1 Main Menu

All           All Test Sequence
Default       Default Test Sequence
Quick         Quick Test Sequence
POptests      Quick POP-Only Test Sequence
Nopoptests    Quick non-POP-Only Test Sequence
TAac          Select Test and Options From Command Line and Environment
Options       Display Option Flags
Tests         Display Test Selection Arguments
Globals       Display Global Options
PRecede       Display Preceding Command

Command ==>
```

The Main menu options are described in the paragraphs that follow.

A *pass=*

The *All* option executes an exhaustive test sequence that fully exercises TAAC-1 functional blocks. The *pass=* argument specifies the number of times to execute the test before returning to the Main Menu.

D *pass=*

The *Default* option executes a test sequence that provides a quick confidence level check of the TAAC-1 board to verify that it is functioning properly. The *pass=* argument specifies the number of times to execute the test before returning to the main menu.

Q *pass=*

The *Quick* option executes a sequence of tests that ensure minimum functionality of the TAAC-1. The *pass=* argument specifies the number of times to execute the test before returning to the main menu.

PO *pass=*

Selecting the *Quick POP-only Test Sequence* invokes quick confidence level checks that specifically test the frame buffer (POP) functions. These tests cover the Image/Data memory, the vector ports, the video registers and the Brooktree RAMDACs.

In the TAAC test sequence, this includes the Video RAM, Vector Port, Vector Port (Write Only), Video Registers, Brooktree RAMDAC and Frame Buffer Checksum tests.

N *pass=*

Selecting the *Quick Non-POP-Only Test Sequence* invokes all quick confidence level checks that are not dependent upon the POP board. This test sequence includes the Sequencer Functionality, ALU Registers, ALU Operations, Barrel Shifter, Floating Point Operations, Look-up Table, Multiplier Accumulator, RDRC, Sequencer Stack, Access Processor, and Interrupts tests.

TA *option*

The *Select Test and Options from Command Line and Environment* Main Menu option specifies all the run time parameters for the TAAC test sequence. Parameters are taken from the command line or from the Exec environment (refer to Chapter 2).

The Options menu on the following page describes possible parameters you may wish to set with the **TA** option. For example, to run a test pass with the "brief mode" on, enter

```
Command ==>TA brief=on
```

Command line parameters override all defaults as well as Exec environment parameters.

Another way to set global test parameters is described next, under the "Options Submenu" section.

The remaining TAAC Main Menu options are navigational commands. When you select one of these options from the Main Menu, a submenu of various tests is displayed. These menus are described briefly in the following text. Each submenu is then shown, and each submenu choice is described.

O

The *Display Option Flags* Main Menu choice displays a menu of the execution option parameters. You may set these parameters with the `set` command, as shown on the menu, or you may specify them as command line parameters without the `set` keyword, as described previously for the **TA** Main Menu option.

Exec environment parameters will override all defaults, but will not override a command line parameter. To view all the Exec environment parameters, enter **set** with no arguments. The option menu will be described later.

T

The *Display Test Section Arguments* Main Menu option brings up a menu of TAAC tests. As shown later in this test, you use the `set` command to determine which tests will be included in the diagnostic sequence.

G

The *Display Global Options* Main Menu choice displays the current setting of the Exec environment parameters. They are displayed on three lines just above the command prompt when the menu is redisplayed.

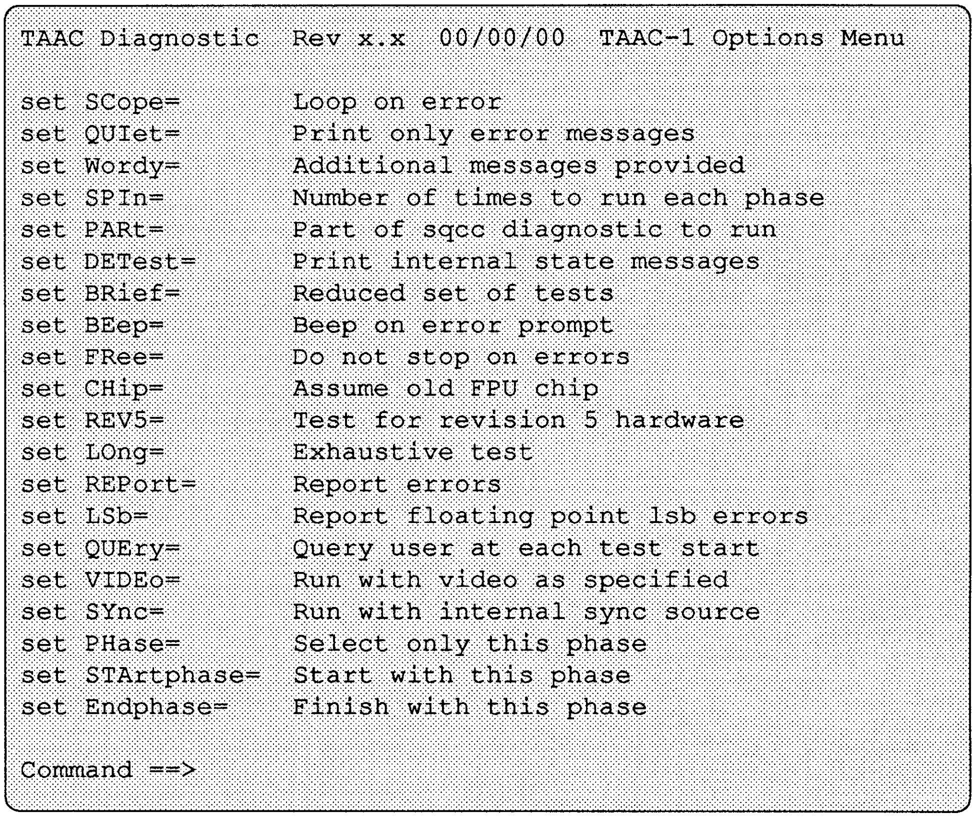
PR

The *Display Preceding Command* Main Menu option displays the previous `TAac` option command line with all its parameters. It does not restart the tests. Instead, the Exec exclamation point ("!") command (described in Chapter 2) is used for that purpose.

Options Menu

When you enter **O** (the *Display Option Flags* choice) from the Main Menu, the submenu shown below is offered. The parameters are specified using the `set` command in the Exec. These are the same parameters that are passed on the command line in the Main Menu `TAac` option. All parameters are set by entering

```
command==>set parameter=on

              or

command==>set parameter=off.
```



```
TAAC Diagnostic  Rev x.x  00/00/00  TAAC-1 Options Menu

set SCope=       Loop on error
set QUIet=       Print only error messages
set Wordy=       Additional messages provided
set SPIn=        Number of times to run each phase
set PARt=        Part of sqcc diagnostic to run
set DETest=      Print internal state messages
set BRief=       Reduced set of tests
set BEep=        Beep on error prompt
set FRee=        Do not stop on errors
set CHip=        Assume old FPU chip
set REV5=        Test for revision 5 hardware
set LOng=        Exhaustive test
set REPort=      Report errors
set LSb=         Report floating point lsb errors
set QUEry=       Query user at each test start
set VIDEo=       Run with video as specified
set SYnc=        Run with internal sync source
set PHase=       Select only this phase
set STArtphase=  Start with this phase
set Endphase=    Finish with this phase

Command ==>
```

The following text describes each of the Options Menu choices.

SET SC=

The *Loop on error* parameter specifies scope-looping. If you enter `scope=on`, the code that produced the error is executed repeatedly until there is acknowledgement from the host. Conversely, if `scope=off` the error is reported and the TAAC-1 waits for acknowledgement. The TAAC diagnostic default is `scope=on`. However, the Exec environment defaults to `SCope=off`. The `TAac` option will therefore use this unless explicitly overridden.

SET QUI=

The *Print only error messages* parameter controls the display of normal operating messages. When `quiet=on` only the error messages from the test are displayed. No extraneous messages are sent to the standard output. The default is `quiet=off`.

SET W=

The *Additional messages provided* option controls the display of operating messages that detail some of the internal test functions. For example, most tests will only display each pattern under test if `wordy=on`. The default is `wordy=off`.

SET SPI=*number*

The *Number of times to run each phase* option specifies the number of passes to run each phase of a single test. Enter the desired number after the equals sign:

```
set spi=1
```

The default is set to `spin=1`.

SET PAR=*number*

The *part* option is specific to the *Sequencer Functionality (SQCC)* test. It controls which of the sequencer instructions to test. After The command

```
set par=
```

is followed with a number between 1 and 46, corresponding to the numbered instruction you wish to test. The default is `part=0`, which executes all sequencer instructions that are supported by the TAAC compiler.

SET DET=

The *Print internal state messages* option controls a debug test flag. When `detest=on`, the various tests will print in great detail the internal operations of the test. The default is `detest=off`.

SET BR=

The *Reduced Set of Tests* option controls the size of the data sets used for testing, and the number of registers that are tested. The command `set brief=on` does more extensive testing while `set BRief=off` uses minimal data sets and and a smaller register set. The default setting is `brief=off` for the Main Menu TAAC option, and `brief=on` for the All, Quick, Default, Poptests, and Nopoptests options. Currently the `brief` parameter affects the following tests:

```
ALU operations
ALU registers
Barrel Shifter
Brooktree RAMDACs
Floating Point Operations
Look-up Table
Multiplier/Accumulator
Microcode RAM
RDRC
Scratchpad RAM
Video Registers
Video RAM
Poly
```

SET BE=

When an error is detected and reported, and **beep=on**, an audible keyboard bell will sound. The default is `beep=off`.

SET FR=

When an error is detected and reported, and `free=off`, the program will not stop. Instead it continues as specified by the `RER` parameter. If `free=on`, the diagnostic enters an interactive mode.

SET REV5=

There are two revisions of the TAAC-1 board. They are the -01 revision, commonly known as REV4, and the -02 revision, commonly known as

REV5. If `rev5=on`, the -02 revision tests are conducted, while if `rev5=off` the -01 revision tests are run.

If neither of these are specified, the diagnostic probes the ID PAL on the TAAC-1 board to determine the correct revision, then reports it. The default setting is to probe for the revision of the board.

SET LON=

Some tests have a long mode in which they generate 32 patterns of a rotating-one bit field (0x1, 0x2, 0x4, 0x8,...) instead using the basic 14 patterns normally used. Specifying `long=on` places those tests into this extended mode. The default is `long=off`.

SET REP=

The *report errors* parameter controls the first action taken when an error is detected. If `report=off` the error is not reported and testing continues. If `report=on` the error is reported and logged, and then execution continues, depending on the `FRee`, `RER`, and `STOP` parameter settings.. The default setting is `report=on`.

SET LS=

The TAAC-1 and the host floating point routines sometimes differ in their results by the value of one least significant bit. If `lsb=off` these discrepancies are not considered errors. If `lsb=on`, the floating point errors of one least significant bit value are treated as real errors and action is taken depending on the `report` parameter setting. The default is `lsb=off`.

SET QUE=

When `query=on` the program enters an interactive mode at the beginning of each test. You are prompted to modify the parameters as desired, and then testing is resumed. The default is `query=off`.

SET VID=

Some tests require that video display and fetch requests be turned on. If `video=off`, those tests that require video are ignored. The default is `video=on`.

SET SY=

The TAAC-1 can either genlock to an external sync source or generate its own sync. If `sync=on` the TAAC-1 expects and requires an external sync source. If `sync=off` the TAAC-1 runs on its internal sync. The default is `sync=off`.

SET PH=

Most of the TAAC-1 tests have more than one phase. In order to facilitate various loop methods, you may specify just a single phase. This applies to all the tests; that is, specifying `phase=3` allows only phase 3 to run in all the tests. Specifying `phase=0` returns to the default state of executing all phases. The default is `phase=0`.

SET STA=*number*

Each test normally starts at its own phase 1. By specifying some other value, all the tests will start at that phase. The specified number must be less than or equal to the number specified after `endphase=`. The default is `startphase=1`.

SET E=*number*

Each test normally finishes up with its own last phase. By specifying some other value, all the tests will end with at phase instead. The specified number must be greater than or equal to the number specified after `stArt-phase=`. The default setting is `endphase=32`.

SET SP=

When `sp=on` a single pass is executed regardless of the value of `pass=`. The default is `sp=off`.

SET RER=

When an error is detected and optionally reported, this parameter is examined for further action. If `rer=on` the program will return to the test which detected the error. If `rer=off` the program will return to the Main Menu. the default is `rer=off`.

SET PASS=

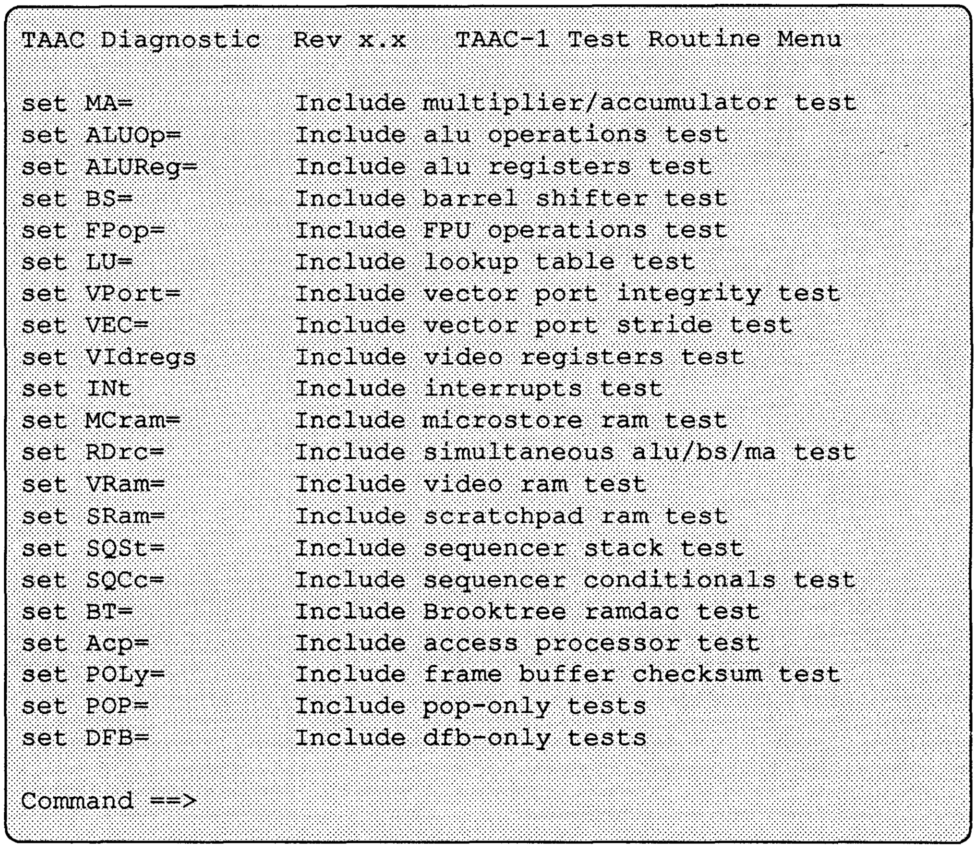
The `pass=` parameter specifies the number of times to execute the entire test suite before returning to the Main Menu. It can be overridden by the `sp=` parameter. The default setting is `pass=1`.

SET STOP=* *or number*

Entering `set stop=*` causes no limit on the number of errors reported during execution. Entering a number places a limit on the number of errors reported. The default setting is no limit for the number of errors to occur before termination.

Tests Routine Menu

The *Display Test Selection Arguments* Main Menu selection invokes a submenu of all possible TAAC test options. These options may be specified with the `set`command. Without the `set` keyword, these tests may be specified as command line parameters. The Test Routine submenu is described below.



```
TAAC Diagnostic  Rev x.x   TAAC-1 Test Routine Menu

set MA=          Include multiplier/accumulator test
set ALUOp=       Include alu operations test
set ALUReg=      Include alu registers test
set BS=          Include barrel shifter test
set FPop=        Include FPU operations test
set LU=          Include lookup table test
set VPort=       Include vector port integrity test
set VEC=         Include vector port stride test
set VIdregs      Include video registers test
set INt          Include interrupts test
set MCram=       Include microstore ram test
set RDrc=        Include simultaneous alu/bs/ma test
set VRam=        Include video ram test
set SRam=        Include scratchpad ram test
set SQSt=        Include sequencer stack test
set SQCc=        Include sequencer conditionals test
set BT=          Include Brooktree ramdac test
set Acp=         Include access processor test
set POLy=        Include frame buffer checksum test
set POP=         Include pop-only tests
set DFB=         Include dfb-only tests

Command ==>
```

To specify the test shown, use the command format

`set` *testname*=**`on`**

This allows only those tests specified to be executed during testing. Conversely, the format

`set` *testname*=**`off`**

allows the default tests to run and exclude those that have been specified.

Furthermore, the command

`set` *testname*=**`begin`**

starts the test suite at the specified test, and continues with the remaining tests in their normal sequence.

Interactive Operations

Under certain circumstances you may be prompted to take action. These circumstances are:

When beginning a test with **query=on.**

When an error is detected with **report=on** and **free=off.**

When you enter `t`.

In all cases, the interactive operation is the same. You may choose from a number of single key options that toggle on or off. The operational parameters affected are the same as those you may specify on the command line. The program returns from interactive mode when a carriage return is entered on a line by itself, or when the you enter a period (**.**).

The interactive options are:

b toggle brief mode
d toggle detest mode
e toggle run_on_error mode
f toggle free_running mode
g toggle single_pass mode
h|? displays help message
k toggle beep mode
l toggle lsb_report_mode
n toggle end_this_phase
o toggle host_scopeloop mode
p toggle end_this_test
q toggle quiet mode
r toggle report_errors mode
s toggle TAAC-1_scopeloop mode
t toggle query mode
u toggle rev5 mode (not normally used)
v toggle video mode
w toggle wordy mode
x exit the program ASAP
y toggle sync mode
<n> toggle inclusion of phase n (currently includes ....)

Host_scopeloop Mode

Most of the interactive options are self explanatory. The Host_scopeloop Mode applies specifically to the MCRAM, SRAM, VRAM, VIDREGS, and SQCC tests. It only applies when an error is reported. Entering **o** or a period to toggle host_scopeloop mode to **on** causes the diagnostic to enter an infinite loop of the failing test.

Entering `t` breaks the loop and displays the Interactive Menu. Entering **o** and a period toggles host_scopeloop to **off.**

17.7. TAAC-1 Test Descriptions

This section describes all of the tests in the diagnostic suite, and lists any error messages produced.

Microcode RAM Test

testmcram
The Microcode RAM Test verifies that the microcode storage RAM is functioning properly. It does not test whether microcode words are being correctly fetched by the sequencer, or correctly applied to the TAAC circuitry. (This is entirely a host based test).

Phase 1:
This phase writes a pattern for each address in the microcode memory, reads back that pattern, and then compares it with the original. Errors for this phase are in the following format:

```
testmcram: error phase 1 Adrs 0x?? Exp 0x?? Rec 0x?? XOR 0x??
 U0001   U0002...
```

Phase 2:
This phase is a unique address test. It writes a changing pattern to all of the microcode RAM, and then reads the data contained in each address cell, comparing it with its expected value. Errors for this phase are in the following format:

```
testmcram: error phase 2 Adrs 0x?? Exp 0x?? Rec 0x?? XOR 0x??
  U0001   U0002...
```

In both phases `Adrs` is the failing address, `Exp` is the pattern written, `Rec` is the pattern that is read back, and `XOR` is the bitwise *exclusive OR* of `Exp` and `Rec` (identifying error bits). A list of memory chip U-numbers involved in the error follows.

Since there is no TAAC-1 microcode running for the `MCram` test, the `Scopeloop` option does not apply. Instead, the `Host_scopeloop` option is used from the Interactive Menu.

Scratchpad RAM Test

testsram
The Scratchpad RAM Test verifies that the the Scratchpad RAM is functioning properly. This is entirely a host based test.

Phase 1:
This phase writes a pattern, for each address in the scratchpad memory, reads back that pattern, and compares it with the original. Errors for this phase are in the following format:

```
testsram: error phase 1 Adrs 0x?? Exp 0x?? Rec 0x?? XOR 0x??
  U0001   U0002...
```

Phase 2:
This phase is a unique address test. This phase writes a changing pattern to all of scratchpad RAM, then reads the data contained in each address, comparing it

with its expected value. Errors for this phase are in the following format:

```
testsram: error phase 2 Adrs 0x?? Exp 0x?? Rec 0x?? XOR 0x??
  U0001   U0002...
```

In both phases `Adrs` is the failing address, `Exp` is the pattern written, `Rec` is the pattern that is read back, and `XOR` is the bitwise *exclusive OR* of `Exp` and `Rec` (identifying error bits). A list of memory chip U-numbers involved in the error follows.

Since there is no TAAC-1 microcode running for the `MCram` test, the `Scopeloop` option does not apply. Instead, the `Host_scopeloop` option is used from the Interactive Menu.

Video RAM Test

`testvram`
The Video RAM Test verifies that the video RAM is functioning properly. This is entirely a host based test.

Phase 1:
This phase writes a changing pattern, for each address in the video memory, reads back that pattern, and compares it with the original. Errors for this phase are in the following format:

```
testvram: error phase 1 Adrs 0x?? Exp 0x?? Rec 0x?? XOR 0x??
  U0001   U0002...
```

Phase 2:
This phase is a unique address test. This phase writes a changing pattern to all of video RAM, then reads the data contained in each address, and compares it with its expected value. Errors for this phase are in the following format:

```
testvram: error phase 2 Adrs 0x?? Exp 0x?? Rec 0x?? XOR 0x??
  U0001   U0002...
```

Phase 3:
This phase places the VME interface into 2D mode before writing a sequence of diagonal line patterns into video memory. The patterns are then verified, in 2D mode. The 1D address of any errors found is reported in the following format:

```
testvram: error phase 3 Adrs 0x?? Exp 0x?? Rec 0x?? XOR 0x??
  U0001   U0002...
```

Phase 4:
This phase is identical to phase 3 except the VME interface is placed into 2D mode when the patterns are written, and then placed into 1D mode to verify (with a corresponding adjustment of the address). The address of any error found is the 1D address, reported in the following format:

```
testvram: error phase 4 Adrs 0x?? Exp 0x?? Rec 0x?? XOR 0x??
  U0001   U0002...
```

Phase 5:
This phase tests the wordmask functions of host access to video memory. It first writes patterns to the wordmask register, and reads them back. Then, the test writes into words 0 and 4 of video memory, and verifies the wordmasked write into selected words. The errors for this phase are in one of the following formats:

```
testvram: error phase 5 write/read expected 0x?? received 0x??
```

or

```
testvram: error phase 5 wordmask write
  mask 0x?? address 0x?? expected 0x??, received 0x??
```

Phase 6:
This phase tests the bitmask functions of host access to video memory. This phase first does a pattern write and subsequent read to all 16 bitmask registers. Then, for each of the bitmask registers, it writes each of 14 patterns into the first eight words of video memory, and verifies the bitmasked write in all eight words. The errors for this phase are in one of the following formats:

```
testvram: error phase 6 write/read
  bitmask ??, expected 0x??, received 0x??
```

or

```
testvram: error phase 6 bitmask write
  bitmaskid ?? mask 0x?? pattern 0x??
  address 0x?? expected 0x?? received 0x??
```

In all phases `Adrs` is the failing address, `Exp` is the pattern written, `Rec` is the pattern that is read back, and `XOR` is the bitwise *exclusive OR* of `Exp` and `Rec` (identifying error bits). For phases 1 through 4, a list of memory chip U-numbers involved in the error follows.

Since there is no TAAC-1 microcode running for the `MCram` test, the `Scopeloop` option does not apply. Instead, the `Host_scopeloop` option is used from the Interactive Menu.

Video Registers Test

`testvidregs`
The Video Registers test verifies the proper operation of the registers on the video board and their associated circuitry. This test has eight phases.

Phase 1:
The first phase writes a pattern in each address in the **HPAR** (Horizontal Parameters RAM), reads that pattern back, and then compares it with the original value.

Phase 2:
This phase is a unique address test. It writes a changing pattern to the **HPAR** (Horizontal Parameters RAM), and then compares it to the expected values for verification.

Phase 3:
This phase tests the **VPAR** (Vertical Parameters RAM).. A pattern is written, read back, and compared with the original value for each **VPAR** address. For Revision-02 board sets, the "inhibit blanking" bit is also tested in this phase.

Phase 4:
This phase does a unique address test on the **VPAR** (Vertical Parameters RAM). This test writes a changing pattern to the **VPAR**, and then reads it from the beginning address and does a compare for verification.

Phase 5:
This phase does a **HPAR** Register Rotating Bit test. A pattern with a single shifting bit is written, read back, and compared in the **HPAR** register.

Phase 6:
This phase does a **VPAR** Register Rotating Bit test. A pattern is written, read back and compared by shifting each bit of the **VPAR** register. For Revision-02 board sets, the "inhibit blanking" bit is also tested in this phase.

Phase 7:
This phase does Miscellaneous Registers Arbitrary Pattern test. This register set includes the `OVLMSK` (Overlay Mask), `CHLSEL`, `VICON` (Video Control), `IDMSK` (Write Mask Data), and `LBVS` (Local Bus Vector Port Select) registers. A pattern is written to each one of these registers, read back, and compared for verification.

Phase 8:
This phase does a Miscellaneous Registers Rotating Bit test. Each of the registers in the Miscellaneous Register Set is written to with a rotating bit test. All errors are in the following format:

```
testvidregs: error phase ?? Adrs 0x?? Exp 0x?? Rec 0x?? XOR 0x??
```

In all phases `Adrs` is the failing address, `Exp` is the pattern written, `Rec` is the pattern that is read back, and `XOR` is the bitwise *exclusive OR* of `Exp` and `Rec` (identifying error bits).

Since there is no TAAC-1 microcode running for the `MCram` test, the `Scopeloop` option does not apply. Instead, the `Host_scopeloop` option is used from the Interactive Menu.

Sequencer Functionality Test

`testsqcc`
Short code fragments are executed while the TAAC-1 bus read/write register and the micro program counter (MPC) are monitored and compared against expected values. This test exercises all sequencer operations used by the TAAC-1 compiler. Errors are in the following format:

```
     testsqcc: part 1 error testing jufb cctrue.  expected 0x?? found 0x??
```

Where `jufb` is the tested operation with a true condition.

In general, success values are of the form `0x50CCE5nn` (`SUCCESnn`) where **nn** counts up by the operation tested (01, 02,...), and failure values are of the form `0xFA1000nn`, with the same meaning for **nn**.

MPC errors are displayed in the format:

```
testsqcc: part 1 error testing jufb cctrue, loop n
  PC= 0x?? expected one of 0x3F77 0x3F78 0x????
```

For each operation, it is known which MPC values the code fragment should pass through. PC identifies the MPC value observed which is not on the list for that code fragment. This test observes the MPC 100 times before moving on to the next fragment. The loop value just identifies where in that 100 tests the error occurred. This is useful for identifying spurious versus hard errors. The test will automatically move on after 5 such errors on any given instruction.

Since there is only primitive TAAC-1 microcode running for the `SQCc` test, the `Scopeloop` option does not apply. Instead, the `Host_scopeloop` option is used from the Interactive Menu. Toggling the `Host_scopeloop` mode to **on** will cause the diagnostic to enter an infinite loop of the failing test.

ALU Registers Test

`testalureg`
The ALU Registers Test runs a vector of 14 patterns through all 64 cells in both ALU register files. Phases 1, 3, and 5 test the RD ALU, while phases 2, 4 and 6 test the RC ALU.

Phases 1 and 2:
These are a simple pattern write-read tests to each cell. All the cells are tested for a given pattern before going on to the next pattern. Errors are displayed in the following format:

```
     testalureg: error phase 1 alu rd readback test set ? register ?
       wrote 0x?? read 0x??
```

Where `set` identifies the pattern sequence number, `register` is the register that is being tested, `wrote` is the data written to the register, and `read` is the value that was read back.

Phases 3 and 4:
These phases provide a cakewalk tests in which each register's contents are

written to and verified, after writing to the immediately preceding and following register. This is done in a cyclical fashion, with register 0 following register 63. Errors are displayed in the form:

```
testalureg: error phase ?? rd cakewalk following|previous set ?:
  register ? wrote 0x?? then 0x?? read 0x??
```

Phases 5 and 6:
These phases provide a unique address test that test across the entire register file. An incrementing pattern is written to the entire file, and then each register's contents are read and verified. Errors are of the form:

```
testalureg: error phase ?? alu rd readback test register ?
  wrote 0x?? read 0x??
```

ALU Operations Test

`testaluop`
The ALU Operations Test runs a vector of 14 patterns against most of the commonly used unary and binary integer ALU operations. Phases 1, 3, and 5 test the RD ALU. Phases 2, 4, and 6 test the RC ALU. The results are compared by the TAAC with a vector of expected results computed by the host.

Phases 1 and 2:
These phases test the condition code generation of the strictly ALU operations. Errors are displayed as follows:

```
testaluop: error phase 1 ccrd operation 0 "add"
  set 0 patterns 0x?? 0x?? result 0x??
  expected ~car ~neg ~ovr ~zero received car neg ovr zero
```

In these phases, `car` is the carry condition code, `neg i` is the negative code, `ovr` is the overflow code, and `zero` is the "equal to zero" code.

Phases 3 and 4:
These phases test the strictly ALU operations. Errors are displayed as follows:

```
testaluop: error phase 3 rd set 0: operation 0 "add"
  patterns 0x?? 0x?? expected 0x?? readback 0x??
```

This test set identifies the ordinate of the patterns applied. Here, operation 0, the `add` instruction, produced a wrong result. Input values, expected values, and actual values are reported.

Phases 5 and 6:
These test operations involve both the ALU output and the shifter output. Patterns are given both for the operation inputs as well as the initial contents of the shifter register (MQ). Errors are displayed as follows:

```
testaluop: error phase 5 rd set ? aluout: operation 0 "sra"
  patterns 0x?? 0x?? 0x?? expected 0x?? readback 0x??
```

or

```
testaluop: error phase 5 rd set ? mq register: operation 0 "sra"
  patterns 0x?? 0x?? 0x?? expected 0x?? readback 0x??
```

The pattern set is the ordinate of the patterns applied. Three patterns are used for each operation, so there are 14*14*14 sets. The meaning of the reported values is the same as for phases 3 and 4.

This test uses a few of the ALU registers. It assumes that the XOR instruction works in the ALU. Errors which can be attributed to the chip itself, whether in the operations or the register file, all require replacement of the chip.

In all error cases, only an indication that the processor detected an error is reported. If the processor condition code logic is not functioning, it may not be a real error. Both the expected value and the actual readback value are printed to help identify this situation. The processor should first be tested with the Sequencer Functionality Test (`testsqcc`).

Barrel Shifter Test

`testbs`

The Barrel Shifter Test runs a vector of 14 patterns through the barrel shifter, testing every possible shift/rotate function on the patterns. Each of 8 phases test each function, for all 32 shift variations. The results are compared by the TAAC to a vector of expected results computed by the host.

The eight phases of this test are:

Phase 1:	`sll`	-shift left logical
Phase 2:	`srl`	-shift right logical
Phase 3:	`slc`	-shift left circular
Phase 4:	`sra`	-shift right arithmetic
Phase 5:	`sllcn`	-shift left logical using count register
Phase 6:	`srlcn`	-shift right logical using count register
Phase 7:	`slccn`	-shift left circular using count register
Phase 8:	`sracn`	-shift right arithmetic using count register

Errors are in the following form:

```
testbs: error phase 1 operation 0 "sll" shift count 1
  pattern ? 0x?? expected ? 0x?? readback ? 0x??
```

For the shift right instructions, the shift count is the two's complement of the actual shift; thus 31 which is -1, is really 1.

Floating Point Operations Test

testfpop

The Floating Point Operations Test generates a vector of 30 valid floating point numbers, and applies most of the commonly used unary and binary floating point operations.

Phase 1:
This phase tests whether the floating point unit can load and unload its product and sum registers, from inputs, from other registers, and/or from itself. Errors are in the following form:

```
testfpop: error phase 1 set ? part ?
  pattern ? 0x?? expected ? 0x?? readback ? 0x??
```

In this phase, part identifies the failing path:

part 0	input->product->output path
part 1	input->sum->output path
part 2	input->sum->product->output path
part 3	input->product->sum->output path
part 4	input->product->product->output path
part 5	input->sum->sum->output path
part 6	input->sum->Creg->aluA->sum->output path
part 7	input->sum->Creg->aluB->sum->output path
part 8	input->product->Creg->mulA->product->output path
part 9	input->product->Creg->mulB->product->output path

(part 6 through part 9 apply only to **REV5** tests).

Phase 2:
This phase tests the single function operations. Errors are in the following form:

```
testfpop: error phase 2 set ?: operation 0 "absa"
  patterns ? 0x?? ? 0x??
  expected ? 0x?? readback ? 0x??
```

Phase 3:
This phase tests chained or combined function operations. These operations generate both a sum and a product in the same step. The error report identifies which is in error, in the form:

```
testfpop: error phase 3 set ?: operation 0 "muladd" sum register
  patterns ? 0x?? ? 0x??
  expected ? 0x?? readback ? 0x??
```

Phases 4 and 5:
Phase 4 tests the conversion from integer to floating point formats. Phase 5 tests the conversion from floating point to integer formats. Errors are in the form:

```
testfpop: error phase 4|5 set ?
  pattern ? 0x?? expected ? 0x?? readback ? 0x??
```

In phase 4, pattern is integer, and both expected and readback values are reported in floating point and hex formats. In phase 5, pattern is floating point, and both the expected and readback values are reported in integer and hex format.

In many cases the expected result generated by the host software floating point routines may differ from the result generated by the TAAC. By default, if the difference is the value of one mantissa LSB, no error is reported. If the lsb_report flag is turned on, however, this condition is treated as a real error.

Look-up Table RAM and ROM Test

testlu

The Look-up Table RAM and ROM Test checks out the hardware look-up tables. There are two tables: an 8K RAM table, and an 8K ROM table.

Phase 1:
This phase provides a unique address write/read test to the RAM table. This test divides the register file into ranges of address lsb's: lo range 0, hi range 1; lo range 0-1, hi range 2-3; lo range 0-3, hi range 4-7; and so on. Errors are displayed in the form:

```
testlu: error phase 1 address bit ?
  wrote 0x?? address 0x?? readback 0x??
```

Where address bit identifies the range: bit 0 for lo range 0, hi range 1, bit 1 for lo range 0-1, hi range 2-3, and so on. address identifies the cell in which the failure occurred.

Phase 2:
This phase tests the contents of the ROM section against a vector of computed values. The values assist in such computations as division, trigonometry, powers, and roots. The ROM tables are composed of two parts, the mantissa and the exponent. These are further divided into 8 subsections by the setting of the mode register bits 21-23. Errors are displayed in the form:

```
testlu: error phase 2 exponent ROM input value 0x??
  ROM address 0x?? MO 21-23 0x??
  expected exponent 0x?? readback 0x?? XOR 0x??
```

or

```
testlu: error phase 2 mantissa ROM input value 0x?
  ROM address 0x?? MO 21-23 0x??
  expected mantissa 0x?? readback 0x?? XOR 0x??
```

Where ROM input value is the number applied for transformation, address is the actual resultant ROM address, and MO 21-23 reports the subsection of the table.

Multiplier Accumulator Test

`testma`
The Multiplier Accumulator Test uses a vector of 14 integer patterns, and applies several multiply/accumulate sequences to the patterns. The accumulator is loaded with a pattern, then the operation to be tested is executed 16 more times. The final content of the accumulator is then tested against an expected value for the operation (computed in the host). Since the multiplier/accumulator chip generates a 64-bit result, the `mh register` and the `ml register` need to be individually tested.

Phases 1 through 4:
These phases test all four combinations of signed and unsigned inputs against all of the normally used operations:

Phase 1:	signed x, signed y
Phase 2:	unsigned x, signed y
Phase 3:	signed x, unsigned y
Phase 4:	unsigned x, unsigned y

Errors are displayed in the form:

```
testma: error phase 1 ma ml|mh operation 0 "pas"
  patterns ? 0x?? ? 0x?? expected ? 0x?? readback ? 0x??
```

Where `ml|mh` indicates whether it is the low 32 bits or the high 32 bits that are in error.

RDRC Test

`testrdrc`
The RDRC Test incorporates many aspects of `aluop`, `ma`, and `bs`, tests. `testrdrc` tests both integer ALU's, the multiplier/accumulator, and the barrel shifter simultaneously. There is only one phase. Errors are displayed in the form:

```
testrdrc: error phase 1 bs|ma loop ? patterns ? 0x?? ? 0x??
  expected ? 0x?? readback ? 0x??
```

or

```
testrdrc: error phase 1 rd|rc operation 0 "add" patterns ? 0x?? ? 0x??
  expected ? 0x?? readback ? 0x??
```

Sequencer Stack Test

`testsqst`
The Sequencer Stack Test checks out the sequencer's ability to use its stack for other than subroutining purposes. Specifically, it tests for stack depth, integrity of push/pop pairs (using both the A and B registers), and tests setting the stack pointer.

Phase 1:
This phase first pops the stack 256 times, (to absolutely clear it), pushes 100 consecutive MPC addresses on to it , and then compares the top of stack (TOS) with

with the first address pushed. Errors are displayed in the form:

```
testsqst: error phase 1 stack depth shows ?
  tos was ? 0x?? should be ? 0x??
```

Where the observed stack depth is reported, along with both the TOS value and the expected TOS value.

Phase 2:
Like phase 1, this test clears the stack and pushes 100 consecutive MPC addresses onto the stack. The stack pointer is loaded with a number ranging from 0 to 65 to reduce the effective stack depth by a known amount. Errors are displayed in the form:

```
testsqst: error phase 2 loop ? stack depth shows ?
  tos was ? 0x?? should be ? 0x?
```

Where `loop` is the amount of stack reduction.

Phases 3 through 6:
These phases test the register-to-stack push/pop function. Phases 3 and 4 push a count value ranging from 0 to 0xffff. Phases 5 and 6 use a 16-bitrotating bit value (0x0, 0x1, 0x2, 0x4, 0x8....). Phases 3 and 5 test the A register, while phases 4 and 6 test the B register. Errors are displayed in the form:

```
testsqst: error phase ? expected ? 0x?? received ? 0x??
```

Access Processor Test

`testacp`
The Access Processor Test verifies the TAAC's "access processor." This is the TAAC's address register/data register pair, (specifically the AC register).

Phase 1:
This phase tests the AC's integrity by loading them with various patterns. First all three parts of the AC register (x, y, z) are tested, and then just one part at a time: (first x, then y, then z). Errors are displayed in the form:

```
testacp: error phase 1 part 2 "acxld" pattern 0x??
  expected 0x?? received 0x??
```

Phases 2 through 5:
These phases test the ACP's incrementer function in 1D, 2D, 3D dice, and 3D slice modes, respectively. For each of 14 patterns loaded into the AC, each of 64 possible increment/decrement/load/nop combinations are applied, with the results compared with expected results. Errors are displayed in the form:

```
testacp: error phase 2 part 4 "acyd acxd" pattern 0x??
  expected 0x?? received 0x??
```

Where in this instance, in 1D mode, the x and y bytes of the AC were enabled to decrement together.

Phase 6 tests:
This phase tests the "z-buffer" conditional write to image memory. Two rotating bit patterns are slid past each other in the high 16 bits of the data word. A write is expected if the new data z value is *less than or equal to* the current z value in the dr. Errors are displayed in the form:

```
testacp: error phase 6 patterns old 0x?? new 0x??
  expected 0x?? received 0x??
```

Where the z value is in the upper 16 bits of old and new; as well as in expected and received (which should each be one of old or new.)

Phase 7:
This phase tests the wordmask write function of the AC. The wordmask is first tested for basic write/read integrity; and then patterns are written to image memory under all 16 possible wordmask configurations. Finally, the image memory is tested for the expected write's. Errors are displayed in the form:

```
testacp: error phase 7 wordmask expected 0x? received 0x?
```

or

```
testacp: error phase 7 wordmask 0x? address 0x??
  expected 0x?? received 0x??
```

Phase 8:
This phase tests the bitmask write function of the AC. All 16 bitmasks are first tested for basic write/read integrity. A vector of 14 different patterns are then written to image memory under all 32 different bitmask configurations using an incrementing bit pattern as the bitmask. Errors are displayed in the form:

```
testacp: error phase 8 bitmask ?? expected 0x? received 0x?
```

or

```
testacp: error phase 8 bitmask ?? pattern 0x?? mask 0x??
  address 0x?? expected 0x?? received 0x??
```

In phases 6 through 8, the POP is required; if the nopop option is used, these phases will not be run.

Vector Port Test

`testvport`
The Vector Port Test checks out the basic integrity of the vector ports, as well as both the ai and ac random ports. It also provides a separate, independent test of video RAM integrity.

There are 10 phases currently defined in `testvport`:

Phase 1:Bank A / Random Write - Random Read
Phase 2:Bank B / Random Write - Random Read
Phase 3:Bank A / Random Write - Vector Read
Phase 4:Bank B / Random Write - Vector Read
Phase 5:Bank A / Vector Write - Random Read
Phase 6:Bank B / Vector Write - Random Read
Phase 7:Bank A / Computed Write - Random Read
Phase 8:Bank B / Computed Write - Random Read
Phase 9:Bank A / Random Write - Computed Read
Phase 10:Bank B / Random Write - Computed Read

In each phase, 0x0 and 0xffffffff are each written to and then immediately read from the current address in video RAM. The path is as described above. For example, in phase 3, the word is written using the ai random port, and then read back using the A vector port. Errors are displayed in the following format:

```
testvport: error phase 1 Adrs 0x?? Exp 0x?? Rec 0x?? XOR 0x??
  U0001   U0002...
```

Where `Adrs` identifies the address that is failing. The pattern written is `Exp`. The patter read back is `Rec`. `XOR` reports error bits. A list of memory chip U-numbers involved in the error follows.

An occasional vector port failure mode occurs when a write error is missed the first time but caught the during a subsequent pass. After each error is reported, and before the test continues, the failed word is again tested to see if it is still in error. (This only applies if `scope_loop` is set to `on`).

At the end of the `vport` test, a list of all error chips is printed along with a count of errors for each chip. If the list grows too long, it will be truncated. If the `wordy` flag is turned **on**, the contents of the four-word block in the vicinity of the failed word is also reported.

Vector Port Test (Write Only)

`testvec`
This vector port test tests both the vector ports, and the entire 1024 word block, at all 4 possible stride values. `testvec` tests only vector port writes, not reads.

Phases 1 through 8:
Phase 1 through 4 test vector port A, while phases 5 through 8 test vector port B. Phases 1 and 5 test a stride of 1 (all 1024 words specified), phases 2 and 6 test a stride of 2 (512 words specified, every other word), phases 3 and 7 test a stride of 3 (341 words specified, every third word), and phases 4 and 8 test a stride of 4 (256 words specified, every fourth word). Errors are displayed in the form:

```
testvec: error phase ?
  word ? expected 0x?? -> readback 0x??
  word ? expected 0x?? -> readback 0x??
```

Ordinarily, only 5 failed words in the block will be reported; if the wordy flag is **on**, as many failed words as will fit into the message buffer will be reported.

Brooktree RAMDAC Test

testbt
The Brooktree RAMDAC Test checks out the Brooktree RAMDAC's and associated circuitry.

Phases 1 and 2:
These phases do a write/read integrity and unique address test on the color lookup tables in each RAMDAC. Phase 1 errors are displayed in the following format:

```
testbt: error phase ? red ramdac, address 0x?
  expected 0x?? received 0x??
```

Each of the four RAMDAC's is tested in turn, along with all three of the color lookup tables within each RAMDAC. red ramdac could also be green ramdac, blue ramdac, or alpha ramdac. address shows the cell within the specified table which failed.

Phase 2:
This phase places a ramp in one channel at a time in a given RAMDAC, and verifies that all three channels in the RAMDAC are correct. Phase 2 errors are displayed in the format:

```
testbt: error phase 2 ramp in red channel, red ramdac, address 0x?
  expected 0x? received 0x?
```

Where the particular failing channel or table is also identified.

Phase 3:
This phase tests the raster timing against the sequencer instruction clock. Both active display and vertical blanking are timed. If they are not found to be within a certain tolerance of expected values, an error is reported:

```
testbt: error phase 3 blanking|active expected ? received ?
```

Where both the expected duration of the indicated portion of the raster, and its received (measured) value, in instruction clock cycles, are reported.

The expected duration depends on whether the TAAC is tested in internal sync (the default) or external sync, and whether it is tested in **rev4** or **rev5**.

Phase 4:
This phase runs an array of 14 colors through each of the 4 overlay registers in each RAMDAC, and verifies that the ACTUAL COLOR OUTPUT from the

RAMDAC is the specified color. It monitors within the chip for one scan line, at the output of all tables and overlays, right at the inputs to the digital-analog converters. This phase also does a minimal test of overlay selection. Error reporting is similar to phase 1, above.

In Phases 4 and higher, the grabbing of a scan line's color is essentially asynchronous with respect to pixel timing. The nibbles are fetched, one at a time, starting with red low, red high, green low, green high, finishing with blue low, blue high. Dropouts of a single nibble of the received color are significant in this regard, especially at one end of the color spectrum or the other (red low or blue high) since the entire sequence of 6 fetches must start and complete during the first active scan line.

Phase 5:
This phase places a line of data value 0x01010101 in the top of memory, then runs an array of color values through the RAMDAC's for that data value, and verifies that the color output is correct. Error reporting is similar to phase 1, above.

Phase 6:
This phase runs an array of data values in line 1, and verifies that the correct lookup table is selected, and the correct color output. Two very different colors are tested for each data value. Error reporting is similar to phase 1, above.

Phase 7:
This phase tests each of the overlay mask bits against each of the alpha channel data bits, and verifying the correct color output of both the alpha RAMDAC and the RGB RAMDAC's. Error reporting is similar to phase 1, above:

```
testbt: error phase 7 red ramdac, mask ? dv ?
  expected 0x? received 0x?
```

Where `mask` is the value in the overlay mask register, and `dv` is the data drawn into the top line in the alpha channel.

Phase 8:
This phase tests the hardware fill PAL's. A single 4-pixel-wide dot is drawn in the first columns of the first line, and the correct color output is verified. The dot is a rotating bit value. A second dot is drawn at column 8, (toggling the fill) and correct color is again verified. A third dot is drawn at column 16, (toggling again) and color again verified. All four RAMDAC's are tested. Errors are displayed in the format:

```
testbt: error phase 8 red ramdac, address 0x?, part ?,
  expected 0x? received 0x?
```

Where `address` is the data value that failed.

Phase 9:

This phase specifically tests the ability of alpha hardware fill to key overlay. Both alpha fill and overlay are separately tested above. Error reporting is similar to phase 1.

Phase 10:
This phase tests the read mask within each of the RAMDAC's. A line of data value 0xffffffff is drawn, and a rotating bit read mask is applied to the RAMDAC's in turn. The mask is also applied to 0xffffffff, and the resulting look-up table cell is tested for two very different colors. Error reporting is similar to phase 1.

Phase 11:
This phase tests the blink circuitry. Only the alpha RAMDAC is tested. The blink mask register is tested with two bit patterns, for all four blink patterns: occult, fast, medium, and slow. The color output of the RAMDAC is verified to switch between the specified data value and the blinked value, at or near the correct rate. The RAMDAC is further tested for overlay blink at occult rate. Errors are reported in the following format:

```
testbt: error phase 10 alpha ramdac, cell 0x?
  blink mask fast duration color1 0x? count1 ?? color2 0x? count2 ??
```

or

```
  blink mask fast duty cycle color1 0x? count1 ?? color2 0x? count2 ??
```

The RAMDAC data value input is blinked between `0xff` and either `0x5a` or `0x0`; the appropriate look-up table entries are loaded with `unblinked = 0xa6a5a4, blinked = 0xc3c2c1. color1` and `color2` in the error report indicate the actual colors observed, and `counts` gives the number of video frames of each.

Phase 12: This phase tests the blanking color. In Revision-01 board sets, the blanking color for the RGB RAMDAC's is in palette location 0; in Revision-02 board sets, it is in overlay register 1. In either case, the alpha RAMDAC color is fixed at black. Error reporting is similar to phase 1.

Phase 13:
This phase tests each row of display memory for a fixed data value and two colors. Errors are reported in the following way:

```
testbt: error phase 12 alpha ramdac, line 0x??
  expected 0x?? received 0x??
```

Where, again, the `expected` and `received` colors are reported for the given scan line.

Interrupts Test

`testint`

The Interrupts Test checks out some of the interrupt capability of the TAAC. The TAAC is capable of responding to three types of interrupts: *breakpoints,* which are specified in the instruction word; *host interrupts,* which are requested by the host via a bit in the slave mode register; and *vertical interrupts,* invoked near the beginning of the vertical blanking period.

Phase 1:
This phase tests the functionality of the breakpoint interrupt. There are six possible failure modes tested:

0. failed jump to interrupt
1. incorrect interrupt status
2. incorrect interrupt condition code
3. incorrect return address
4. failed return from interrupt
5. incorrect return status

Phase 1 errors are reported in the following format:

```
testint: error phase 1 part 0 failed jump to interrupt
  interrupt status (~B ~H ~V ) mask (~B ~H ~V )
  interrupt return address 0x?? should be 0x??
```

Where the interrupt `status` and `mask` values are reported as their truth value (~B in the status means that this is *not* a breakpoint interrupt, ~H in the mask means host interrupts are *not* enabled.) The `interrupt return address` is really meaningful only in the incorrect return address errors.

Phase 2:
This phase tests the breakpoint inhibit function by first forcing a breakpoint, then requesting another while still in the first. The second should be inhibited. Errors are reported in the following format:

```
testint: error phase 2 failed inhibit
  interrupt status (~B ~H ~V ) mask ( B ~H ~V )
```

Phase 3:
This phase tests the preservation of conditions across a breakpoint interrupt. When such an interrupt occurs, the ALU status conditions are saved; status from further operations requested while in the interrupt are handled by different circuitry. Upon return from interrupt, the original environment should be available in the next instruction, as though the interrupt never occurred.

The TAAC tests this 0x1000 times. The background code adds 0x11111111 plus 0x11111111, with a condition of != 0. The interrupt code xors 0x11111111 and 0x11111111, with a condition of == 0. A failure indicates the returned code found condition == 0.

Phase 3 errors are reported in the following format:

```
testint: error phase 3 failed condition preservation
 loop 0x? result 0x??
```

Where loop is the test count (out of 0x1000) and result is the computed value.

Phases 4 through 6:
These phases are identical to 1 through 3 except that the host break interrupt is tested instead of the breakpoint interrupt.

Phases 7 through 8:
These phases are identical to 1 through 3 except that the vertical interrupt is tested instead of breakpoint.

Frame Buffer Checksum Test

testpoly
The Frame Buffer Checksum Test is an implementation of the poly program from other frame buffer product diagnostics. Each phase first draws a replicated smooth shaded, z-buffered, concave polygon into each bank, then generates the sum and xor checksums of the region, and finally compares the data with the known values.

While testpoly does perform a "real world" test of the hardware, using concave, gouraud shaded, z-buffered polygons from the TAAC-1 graphics library, and will detect single pixel errors, it cannot pinpoint the failing feature in the hardware when such errors occur. It merely calls attention to the need to run more rigorous tests. Errors from all phases are reported in the following format:

```
testpoly: error phase 1 bank A
  sum expected 0x?? received 0x?? diff 0x??
  xor expected 0x?? received 0x??  XOR 0x??
```

Where sum is the 32-bit sum of all pixels in the region, and xor is the 32-bit exclusive-or total of all pixels in the region. Both the difference or the XOR between expected and received is also reported.

18

High Speed Serial Interface Diagnostic

18

High Speed Serial Interface Diagnostic

18.1. General Description

The High Speed Serial Interface (HSI) Controller Board provides two high bandwidth serial communications channels for Sun-3 and Sun-4 workstations. This board uses a 32-bit Address and 32-bit Data VME interface and contains 1 Mbyte of VME addressable memory. The serial channels on the board may be configured for either an RS-449 or a V.35 communications interface.

The HSI Controller Board includes three major components:

- Dynamic RAM
- Mostek 5025 serial communication channels
- Zilog 8536 CIO counter/timer and parallel I/O units

18.2. What This Chapter Contains

Following an overview of the diagnostic and a list of required hardware, the Main Menu and submenus are discussed. Menu selections and optional arguments are listed, along with brief descriptions. The end of the chapter contains a list of completion and error messages and a glossary.

18.3. Overview of the Diagnostic

The HSI Diagnostic has been designed to test the features of the HSI Controller Board to ensure that it performs as specified in the *HSI Theory of Operations* and the *HSI External Reference Specification* (PN 800-3102). It tests each major component and its supporting hardware. Functional sections of the HSI board may be tested individually, and an option is also provided to execute certain tests on the functional sections in a continuous sequence.

The tests may be interrupted at any time during execution. Online help is available. The parameters of each test are given default values upon execution. The HSI Diagnostic also generates and stores meaningful error messages for later retrieval.

18.4. Hardware Requirements

The following hardware is required to run the HSI Diagnostic:

- A Sun-3 or Sun-4 workstation with a cardcage
- A fully populated HSI Controller Board
- A boot device (local disk, local tape, or remote disk over Ethernet)
- 2 RS-449 loopback connectors

- 2 V.35 loopback connectors
- 1 RS-449 to RS-449 cable
- 1 V.35 to V.35 cable

For more information on installing the HSI board, installing loopback connectors, and connecting V.35 devices, refer to *SunLink High-Speed Serial Interface Installation and Service Manual* (Part number 813-1046) and *Cardcage Slot Assignments and Backplane Configuration Procedures for the SunLink High-Speed Serial Interface Board* (Part number 813-2058).

18.5. User Interface

The user interface of the HSI Diagnostic adheres to the standards of the Exec menu system. Each test may be selected from a menu by typing the letter or letters displayed in uppercase in the column on the left side of the menu.

Additional parameters and options may be specified on the command line. The HSI Diagnostic supports the common options and parameters (`pass=`, for example) that may be used with other tests which run under the Exec. For a complete discussion of these options, see Chapter 2, "Using the SunDiagnostic Executive."

Press [ESC] then [RETURN] to move back one level in the menu hierarchy. To exit the diagnostic and return to the Exec, press **x** while the Main Menu is displayed.

To display online Help for any menu option, enter the following on the command line:

? *option_name*

18.6. Starting the Diagnostic

For information on starting the Exec, refer to Chapter 2, "Using the SunDiagnostic Executive." After you have started the Exec, choose the HSI Diagnostic from the Diagnostics Menu.

18.7. The Diagnostic Menus

This section of the chapter provides a modular description of the HSI Diagnostic, beginning with the Main Menu and working down through the options available on each of the submenus. A list of successful completion messages and error messages for each test is given in the section entitled "Messages."

18.8. Main Menu

The Main Menu, which displays when you start the HSI Diagnostic, provides access to the submenus of the individual tests:

```
HSI CONTROLLER BOARD DIAGNOSTIC  Rev: X.X  MM/DD/YY  Main Menu

All   - All Sequential Test
Quick - Quick Sequential Test
Cm    - CIO Menu
Dm    - Dynamic RAM Menu
Sm    - Serial Controller Menu
Pm    - Programmable Divider Menu
x     - Exit

Command ==>
```

All Sequential Test

A

The *All Sequential Test* option on the Main Menu executes all of the tests available for this diagnostic. Each sequence of tests is executed the number of times specified by the Exec `pass=` global option. The All Sequential test is long-running and verifies the functionality of the HSI board with a 95% level of confidence.

Quick Sequential Test

Q

The *Quick Sequential Test* option on the Main Menu executes the Quick Test option within each submenu. Each sequence of tests is executed the number of times specified by the Exec `pass=` global option. The Quick Test is relatively short and verifies the functionality of the HSI board with a 90% level of confidence.

CIO Menu

This section provides descriptions of the tests available to verify the features of both CIO A and CIO B. When you choose **C** from the Main Menu, the CIO Menu is displayed:

```
HSI CONTROLLER BOARD DIAGNOSTIC  Rev: X.X  MM/DD/YY  CIO Menu

All   - All Sequential Test
Quick - Quick Sequential Test
ASm   - CIO A Scope Menu
BSm   - CIO B Scope Menu
ACC   - CIO Access Test
RP    - CIO Port Read
WP    - CIO Port Write
RR    - CIO Control Register Read
WR    - CIO Control Register Write
Vm    - V.35 Modem Control Test Menu
RSm   - RS-449 Modem Control Test Menu
Int   - Interrupt Verification
PI    - Parity Interrupt Verification
MIm   - Modem Control Interrupt Verification Menu

Command ==>
```

Optional Parameters

Some options on the CIO Menu and its submenus take parameters. The meanings and default values of the parameters are shown below.

Parameter	*Description*	*Default*
`pass`	Pass count	1
`pat`	Byte hex write pattern	0
`reg`	Hex address register	None

All Sequential Test

A

The *All Sequential Test* option on the CIO Menu causes options 3 and 8–12 to execute sequentially. The All Sequential Test options of the submenus are executed. The RS-449 and V.35 loopback connectors must be installed on the board under test during execution of this selection.

Quick Sequential Test

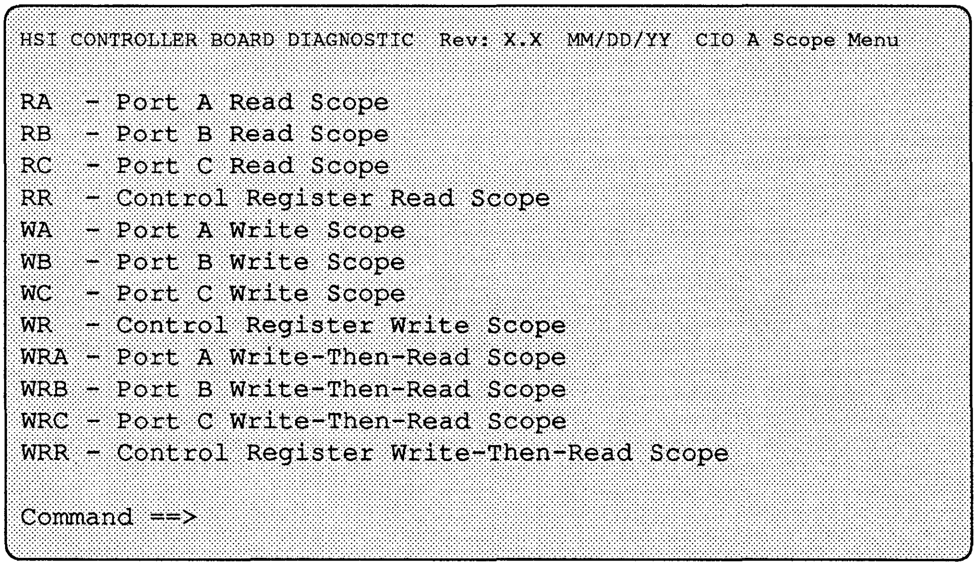
Q

The *Quick Sequential Test* option on the CIO Menu causes options 3, 8, and 9 to execute sequentially. The Quick Sequential Test options of the submenus are executed. The RS-449 and V.35 loopback connectors must be installed on the board under test.

CIO A Scope Menu

AS

When you choose the *CIO A Scope Menu* option, this submenu displays:



```
HSI CONTROLLER BOARD DIAGNOSTIC  Rev: X.X  MM/DD/YY  CIO A Scope Menu

RA  - Port A Read Scope
RB  - Port B Read Scope
RC  - Port C Read Scope
RR  - Control Register Read Scope
WA  - Port A Write Scope
WB  - Port B Write Scope
WC  - Port C Write Scope
WR  - Control Register Write Scope
WRA - Port A Write-Then-Read Scope
WRB - Port B Write-Then-Read Scope
WRC - Port C Write-Then-Read Scope
WRR - Control Register Write-Then-Read Scope

Command ==>
```

RA

The *Port A Read Scope* option on the CIO A Scope Menu permits continuous read access of CIO Port A. This option has no parameters.

RB

The *Port B Read Scope* option on the CIO A Scope Menu permits continuous read access of CIO Port B. This option has no parameters.

RC

The *Port C Read Scope* option on the CIO A Scope Menu permits continuous read access of CIO Port C. This option has no parameters.

RR

The *Control Register Read Scope* option on the CIO A Scope Menu permits continuous read access of the CIO Control Register. This option has no parameters.

WA *pat=*

The *Port A Write Scope* option on the CIO A Scope Menu permits continuous write access to CIO Port A.

WB *pat=*

The *Port B Write Scope* option on the CIO A Scope Menu permits continuous write access to CIO Port B.

WC *pat=*

The *Port C Write Scope* option on the CIO A Scope Menu permits continuous write access to CIO Port C.

WR *pat=*

The *Control Register Write Scope* option on the CIO A Scope Menu permits continuous write access to the CIO Control Register.

WRA *pat=*

The *Port A Write-Then-Read Scope* option on the CIO A Scope Menu permits continuous write-then-read access to CIO Port A. This option provides a tight loop in which the hardware is exercised quickly and continuously. Therefore, for purposes of speed, the value read is not compared with the write pattern.

WRB *pat=*

The *Port B Write-Then-Read Scope* option on the CIO A Scope Menu permits continuous write-then-read access to CIO Port B. The value read is not compared with the write pattern.

WRC *pat=*

The *Port C Write-Then-Read Scope* option on the CIO A Scope Menu permits continuous write-then-read access to CIO Port C. The value read is not compared with the write pattern.

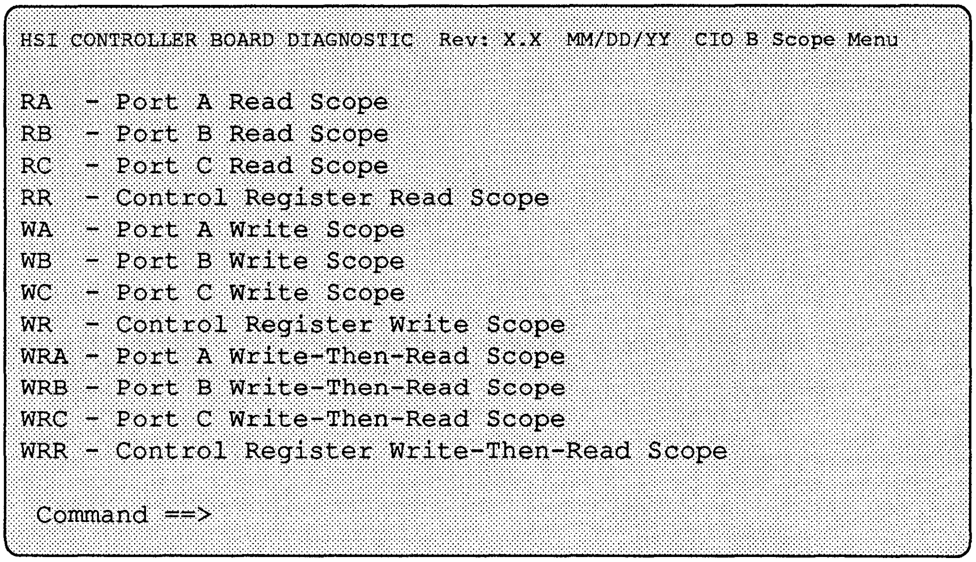
WRR *pat=*

The *Control Register Write-Then-Read Scope* option on the CIO A Scope Menu permits continuous write-then-read access to the CIO Control Register. The value read is not compared with the write pattern.

CIO B Scope Menu

BS

When you choose the *CIO B Scope Menu* option from the CIO Menu, this submenu displays:



```
HSI CONTROLLER BOARD DIAGNOSTIC  Rev: X.X  MM/DD/YY  CIO B Scope Menu

RA  - Port A Read Scope
RB  - Port B Read Scope
RC  - Port C Read Scope
RR  - Control Register Read Scope
WA  - Port A Write Scope
WB  - Port B Write Scope
WC  - Port C Write Scope
WR  - Control Register Write Scope
WRA - Port A Write-Then-Read Scope
WRB - Port B Write-Then-Read Scope
WRC - Port C Write-Then-Read Scope
WRR - Control Register Write-Then-Read Scope

 Command ==>
```

The options on this menu perform the same tests for CIO B as those on the CIO A Scope Menu, just described, perform for CIO A. The parameters and defaults are the same.

CIO Access Test

ACC *pass=*

The *CIO Access Test* option on the CIO Menu provides a verification of the data lines connected to both CIO A and CIO B.

CIO Port Read

RP

The *CIO Port Read* option on the CIO Menu displays the current values read from the ports of both CIO A and CIO B. The option has no parameters.

CIO Port Write

WP *pat= cio= port=*

The *CIO Port Write* option on the CIO Menu allows you to write a hexadecimal bit pattern to a specific CIO port.

Parameter	*Description*	*Values*
`cio`	CIO to which the pattern is written.	A or B
`port`	Port to which the pattern is written.	A or B

CIO Control Register Read

RR

The *CIO Control Register Read* option on the CIO Menu displays the current values read from the control registers of both CIO A and CIO B. The option has no parameters.

CIO Control Register Write

WR *cio= reg= pat=*

The *CIO Control Register Write* option on the CIO Menu allows you to write a hexadecimal bit pattern to a specific CIO control register.

Parameter	*Description*	*Values*
`cio`	CIO to which the pattern is written.	A or B
`reg`	Control register to which the pattern is written.	0–2F

V.35 Modem Control Test Menu

V

When you choose the *V.35 Modem Control Test Menu* option from the CIO Menu, this submenu displays:

```
HSI CONTROLLER BOARD DIAGNOSTIC    Rev: X.X    MM/DD/YY
V.35 Modem Control Test Menu

DCDA      - MK5025 A DTR and DCD Modem Control Verification
CTSA      - MK5025 A RTS and CTS Modem Control Verification
DSRA      - MK5025 A DTR and DSR Modem Control Verification
DCDB      - MK5025 B DTR and DCD Modem Control Verification
CTSB      - MK5025 B RTS and CTS Modem Control Verification
DSRB      - MK5025 B DTR and DSR Modem Control Verification

Command ==>
```

DCDA *pass=*

The *MK5025 A DTR and DCD Modem Control Verification* option on the V.35 Modem Control Test Menu checks the DCD and DTR signals of Serial Controller A for the V.35 configuration.

Data Terminal Ready (DTR) is a signal that the HSI sends out the RS-449 or V.35 port to a remote device. The device waits to receive the signal before sending data.

Data Carrier Detect (DCD) is a signal that the HSI receives from the RS-449 or V.35 port. It indicates that the remote device is present and working correctly.

This test requires external loopback support which connects DCD to DTR of Serial Controller A. The test begins by turning on DTR and verifies that DCD is on. It then turns off DTR and verifies that DCD is off. DTR and DCD are controlled and monitored alternately through Port B of CIO B. Interrupts through CIO A are disabled during this test.

CTSA *pass=*

The *MK5025 A RTS and CTS Modem Control Verification* option on the V.35 Modem Control Test Menu checks the RTS and CTS signals of Serial Controller A for the V.35 configuration.

Request to Send (RTS) is a signal that the HSI sends out a synchronous RS-449 or V.35 port to a remote device. The device responds, when ready, with Clear to Send (CTS).

This test requires external loopback support which connects RTS to CTS of Serial Controller A. The test begins by turning on RTS and verifies that CTS is on. It then turns off RTS and verifies that CTS is off. RTS and CTS are controlled and monitored alternately through Port B of CIO B. Interrupts through CIO A are disabled during this test.

DSRA *pass=*

The *MK5025 A DTR and DSR Modem Control Verification* option on the V.35 Modem Control Test Menu checks the DTR and DSR signals of Serial Controller A for the V.35 configuration.

Data Set Ready (DSR) is a signal received by HSI when a remote device has data to be sent. The HSI should respond, when ready, with Data Terminal Ready (DTR).

This test requires external loopback support which connects DTR to DSR of Serial Controller A. The test begins by turning on DTR and verifies that DSR is on. It then turns off DTR and verifies that DSR is off. DTR and DSR are controlled and monitored alternately through Port B of CIO B. Interrupts through CIO A are disabled during this test.

DCDB *pass=*

The *MK5025 B DTR and DCD Modem Control Verification* option on the V.35 Modem Control Test Menu performs the same tests on Serial Controller B as the MK5025 A DTR and DCD Modem Control Verification option, previously described, performs on Serial Controller A.

CTSB *pass=*

The *MK5025 B RTS and CTS Modem Control Verification* option on the V.35 Modem Control Test Menu performs the same tests on Serial Controller B as the MK5025 A RTS and CTS Modem Control Verification option, previously described, performs on Serial Controller A.

DSRB *pass=*

The *MK5025 B DTR and DSR Modem Control Verification* option on the V.35 Modem Control Test Menu performs the same tests on Serial Controller B as the MK5025 A DTR and DSR Modem Control Verification option, previously described, performs on Serial Controller A.

RS-449 Modem Control Test Menu

RS

When you choose the *RS-449 Modem Control Test Menu* option from the CIO Menu, this submenu displays:

```
HSI CONTROLLER BOARD DIAGNOSTIC   Rev: X.X    MM/DD/YY
RS-449 Modem Control Test Menu

DCDA      - MK5025 A DTR and DCD Modem Control Verification
CTSA      - MK5025 A RTS and CTS Modem Control Verification
DSRA      - MK5025 A DTR and DSR Modem Control Verification
DCDB      - MK5025 B DTR and DCD Modem Control Verification
CTSB      - MK5025 B RTS and CTS Modem Control Verification
DSRB      - MK5025 B DTR and DSR Modem Control Verification

Command ==>
```

All of the options on the RS-449 Modem Control Test Menu function in the same ways as the corresponding options on the V.35 Modem Control Test Menu. See the preceding description of the V.35 Modem Control Test Menu for information.

Interrupt Verification

`I`

The *Interrupt Verification* option on the CIO Menu checks the interrupt capability of CIO A. An interrupt is forced by enabling interrupts on Port A of CIO A and setting the Interrupt Pending (IP) bit of the Port A Command Status Register. Proper interrupt processing on the HSI board occurs when an interrupt is detected across the VME. An error is reported if an interrupt across the VME is not detected within a timeout period. An interrupt level must be enabled on the HSI board through the appropriate DIP switch in order to run this test. For information on the default DIP switch settings, refer to *Configuration Procedures for the SunLink High-speed Serial Interface Board.*

This option has no parameters.

Parity Interrupt Verification

`PI`

The *Parity Interrupt Verification* option on the CIO Menu verifies the ability of the HSI to generate a memory parity error interrupt. It first clears the Parity Error bit in Port A of CIO A. It then programs CIO A Port A to interrupt upon a 0-to-1 transition of the Parity Error bit and enable interrupts from this port. The test then redefines memory for the parity opposite to the current setting and attempts to read memory. A parity error interrupt through CIO A should result when the attempt is made to access memory. An error is reported if a parity error interrupt is not detected within a timeout period.

The even or odd parity is set through Port A of CIO B. The Clear Parity Error bit in Port A of CIO A is asserted upon detection of a parity error. An interrupt level must be enabled on the HSI board through the appropriate DIP switch in order to run this test. For information on the default DIP switch settings, refer to *Configuration Procedures for the SunLink High-speed Serial Interface Board.*

This option has no parameters.

Modem Control Interrupt Verification Menu

MI

When you choose the *Modem Control Interrupt Verification Menu* option from the CIO Menu, this submenu displays:

```
HSI BOARD DIAGNOSTIC   Rev: X.X   MM/DD/YY
Modem Control Int Verification Menu

DCDAR   - MK-A RS-449 DCD Interrupt Verification
DSRAR   - MK-A RS-449 DSR Interrupt Verification
CTSAR   - MK-A RS-449 CTS Interrupt Verification
DCDAV   - MK-A V.35 DCD Interrupt Verification
DSRAV   - MK-A V.35 DSR Interrupt Verification
CTSAV   - MK-A V.35 CTS Interrupt Verification
DCDBR   - MK-B RS-449 DCD Interrupt Verification
DSRBR   - MK-B RS-449 DSR Interrupt Verification
CTSBR   - MK-B RS-449 CTS Interrupt Verification
DCDBV   - MK-B V.35 DCD Interrupt Verification
DSRBV   - MK-B V.35 DSR Interrupt Verification
CTSBV   - MK-B V.35 CTS Interrupt Verification

Command ==>
```

DCDAR *pass=*

The *MK-A RS-449 DCD Interrupt Verification* option on the Modem Control Interrupt Verification Menu verifies the ability of the DCD modem control line to generate interrupts for MK5025-A through RS-449. This test requires the external loopback support which connects DCD to DTR of Serial Controller A for RS-449.

This test first programs CIO A Port B to interrupt upon a 0-to-1 transition of the DCD input signal. The DTR output signal is then turned on to induce a DCD interrupt. An error is reported if a DCD interrupt across the VME is not detected within a timeout period. An interrupt level must be enabled on the HSI board through the appropriate DIP switch in order to run this test. For information on the default DIP switch settings, refer to *Configuration Procedures for the SunLink High-speed Serial Interface Board.*

DSRAR *pass=*

The *MK-A RS-449 DSR Interrupt Verification* option on the Modem Control Interrupt Verification Menu verifies the ability of the DSR modem control line to generate interrupts for MK5025-A through RS-449. This test requires the external loopback support which connects DSR to DTR of Serial Controller A for RS-449.

This test first programs CIO A Port B to interrupt upon a 0-to-1 transition of the DSR input signal. The DTR output signal is then turned on to induce a DSR interrupt. An error is reported if a DSR interrupt across the VME is not detected within a timeout period. An interrupt level must be enabled on the HSI board through the appropriate DIP switch in order to run this test. For information on the default DIP switch settings, refer to *Configuration Procedures for the SunLink High-speed Serial Interface Board.*

CTSAR *pass=*

The *MK-A RS-449 CTS Interrupt Verification* option on the Modem Control Interrupt Verification Menu verifies the ability of the CTS modem control line to generate interrupts for MK5025-A through RS-449. This test requires the external loopback support which connects CTS to RTS of Serial Controller A for RS-449.

This test first programs CIO A Port B to interrupt upon a 0-to-1 transition of the CTS input signal. The RTS output signal is then turned on to induce a CTS interrupt. An error is reported if a CTS interrupt across the VME is not detected within a timeout period. An interrupt level must be enabled on the HSI board through the appropriate DIP switch in order to run this test. For information on the default DIP switch settings, refer to *Configuration Procedures for the SunLink High-speed Serial Interface Board.*

DCDAV *pass=*

The *MK-A V.35 DCD Interrupt Verification* option on the Modem Control Interrupt Verification Menu performs the same tests for the V.35 configuration that the MK-A RS-449 DCD Interrupt Verification option, just described, performs for the RS-449 configuration.

DSRAV *pass=*

The *MK-A V.35 DSR Interrupt Verification* option on the Modem Control Interrupt Verification Menu performs the same tests for the V.35 configuration that the MK-A RS-449 DSR Interrupt Verification option, just described, performs for the RS-449 configuration.

CTSAV *pass=*

The *MK-A V.35 CTS Interrupt Verification* option on the Modem Control Interrupt Verification Menu performs the same tests for the V.35 configuration that the MK-A RS-449 CTS Interrupt Verification option, just described, performs for the RS-449 configuration.

DCDBR *pass=*

The *MK-B RS-449 DCD Interrupt Verification* option on the Modem Control Interrupt Verification Menu verifies the ability of the DCD modem control line to generate interrupts for MK5025-B through RS-449. This test requires the external loopback support which connects DCD to DTR of Serial Controller B for RS-449.

This test first programs CIO A Port B to interrupt upon a 0-to-1 transition of the DCD input signal. The DTR output signal is then turned on to induce a DCD interrupt. An error is reported if a DCD interrupt across the VME is not detected within a timeout period. An interrupt level must be enabled on the HSI board through the appropriate DIP switch in order to run this test. For information on the default DIP switch settings, refer to *Configuration Procedures for the SunLink High-speed Serial Interface Board.*

DSRBR *pass=*

The *MK-B RS-449 DSR Interrupt Verification* option on the Modem Control Interrupt Verification Menu verifies the ability of the DSR modem control line to generate interrupts for MK5025-B through RS-449. This test requires the external loopback support which connects DSR to DTR of Serial Controller B for RS-449.

This test first programs CIO A Port B to interrupt upon a 0-to-1 transition of the DSR input signal. The DTR output signal is then turned on to induce a DSR interrupt. An error is reported if a DSR interrupt across the VME is not detected within a timeout period. An interrupt level must be enabled on the HSI board through the appropriate DIP switch in order to run this test. For information on the default DIP switch settings, refer to *Configuration Procedures for the SunLink High-speed Serial Interface Board.*

CTSBR *pass=*

The *MK-B RS-449 CTS Interrupt Verification* option on the Modem Control Interrupt Verification Menu verifies the ability of the CTS modem control line to generate interrupts for MK5025-B through RS-449. This test requires the external loopback support which connects CTS to RTS of Serial Controller B for RS-449.

This test first programs CIO A Port B to interrupt upon a 0-to-1 transition of the CTS input signal. The RTS output signal is then turned on to induce a CTS interrupt. An error is reported if a CTS interrupt across the VME is not detected within a timeout period. An interrupt level must be enabled on the HSI board through the appropriate DIP switch in order to run this test. For information on the default DIP switch settings, refer to *Configuration Procedures for the SunLink High-speed Serial Interface Board.*

DCDBV *pass=*

The *MK-B V.35 DCD Interrupt Verification* option on the Modem Control Interrupt Verification Menu performs the same tests for the V.35 configuration that the MK-B RS-449 DCD Interrupt Verification option, just described, performs for the RS-449 configuration.

DSRBV *pass=*
The *MK-B V.35 DSR Interrupt Verification* option on the Modem Control Interrupt Verification Menu performs the same tests for the V.35 configuration that the MK-B RS-449 DSR Interrupt Verification option, just described, performs for the RS-449 configuration.

CTSBV *pass=*
The *MK-B V.35 CTS Interrupt Verification* option on the Modem Control Interrupt Verification Menu performs the same tests for the V.35 configuration that the MK-B RS-449 CTS Interrupt Verification option, just described, performs for the RS-449 configuration.

Dynamic RAM Menu

This section describes the tests available to verify the functionality of the 1 Mbyte of Dynamic RAM on the HSI board. The tests include several pattern verification tests and a parity error test. Each pattern verification test may be customized with the following options:

- □ Byte, word, or long data sizes. High-order data bits are truncated according to the selected size.

  Default: Byte

- □ A range of Dynamic RAM addresses to test.

  Default: All RAM locations

- □ A pattern mask to mask out specific bits during pattern verification.

  Default: 0xFFFFFFFF

When you choose **D** from the Main Menu, the Dynamic RAM Menu is displayed:

```
HSI CONTROLLER BOARD DIAGNOSTIC   Rev: X.X   MM/DD/YY
Dynamic RAM Menu

All      - All Sequential Test
Quick    - Quick Sequential Test
PARm     - Set Parity Menu
FPAR     - Force Parity Error Test
ADR      - Address Test (unique pattern)
PAT      - Pattern Test (constant pattern)
CHK      - Checker Pattern Test
UNIQ     - Unique Pattern Test
MATS     - MATS+ Pattern Test
RAND     - Random Pattern Test
NTA      - NTA Test
TRI      - Triangle Pattern Test
DIAG     - Diagonal Pattern Test
ACCm     - DRAM Access Menu

Command ==>
```

'Optional Parameters

Most of the options on the Dynamic RAM Menu and its submenus take parameters. The meanings and default values of the parameters are shown below.

Parameter	*Description*	*Default*
`pass`	Pass count	1
`pat`	Hex test pattern	0x55555555
`sa`	Hex start address	First DRAM location
`ea`	Hex end address	Last DRAM location
`dsize`	Byte/Word/Long word	Byte
`Mask`	Hex bit pattern	0xFFFFFFFF

All Sequential Test

A

The *All Sequential Test* option on the Dynamic RAM Menu causes options 3–11 to execute sequentially for byte, word, and long data modes.

Quick Sequential Test

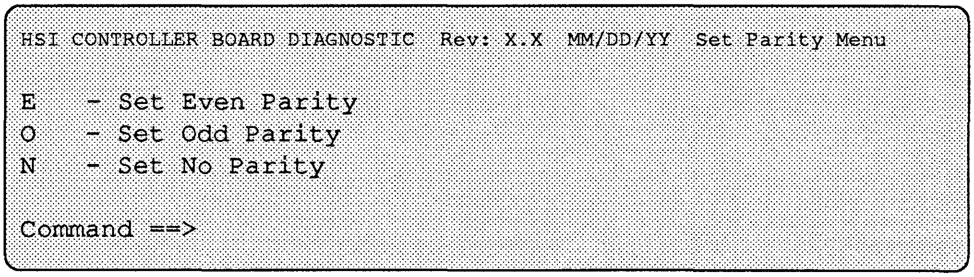
Q

The *Quick Sequential Test* option on the Dynamic RAM Menu causes options 3, 4, and 5 to execute sequentially for byte, word, and long data modes.

Set Parity Menu

PAR

When you choose the *Set Parity Menu* option from the Dynamic RAM Menu, this submenu displays:



```
HSI CONTROLLER BOARD DIAGNOSTIC  Rev: X.X  MM/DD/YY  Set Parity Menu

E    - Set Even Parity
O    - Set Odd Parity
N    - Set No Parity

Command ==>
```

The options on this menu manipulate the DRAM parity.

E

The *Set Even Parity* option on the Set Parity Menu changes the Parity Bit for the DRAM to Even. This option has no parameters.

O

The *Set Odd Parity* option on the Set Parity Menu changes the Parity Bit for the DRAM to Odd. This option has no parameters.

N

The *Set No Parity* option on the Set Parity Menu disables parity checking during the DRAM tests. Disabling parity results in only read-back verification of written patterns without verification of the Parity Error bit. This option has no parameters.

Force Parity Error Test

FPAR

The *Force Parity Error Test* option on the Dynamic RAM Menu verifies the ability of the HSI to indicate if a memory parity error has occurred.

The test first clears the Parity Error bit in Port A of CIO A. It then redefines memory for the parity opposite to the current setting and attempts to read memory. An error is reported if the Parity Error bit is not set within a timeout period after the attempt to access memory. The Even or Odd parity is set through Port A of CIO B. The 'Clear Parity Error' bit in Port A of CIO A is asserted upon detection of a parity error.

This option has no parameters.

Address Test

ADR *pass= sa= ea= dsize= mask=*

The *Address Test (unique pattern)* option on the Dynamic RAM Menu first fills each location to be tested with its corresponding address. It then sequentially verifies each location for the expected contents.

Pattern Test

PAT *pass= sa= ea= dsize= mask= pat=*

The *Pattern Test (constant pattern)* option on the Dynamic RAM Menu first fills each location to be tested with the pattern you specify. It then sequentially verifies each location for the expected contents.

Checker Pattern Test

CHK *pass= sa= ea= dsize= mask= pat=*

The *Checker Pattern Test* option on the Dynamic RAM Menu alternates between two different patterns as it fills each location to be tested. It then sequentially verifies each location for the expected contents. One of the two patterns used is the pattern specified by you. The other is the pattern resulting from inverting the bits of the specified pattern. The pattern 0x55555555 and its inverse, 0xAAAAAAAA, are used if no pattern is specified.

Unique Pattern Test

UNIQ *pass= sa= ea= dsize= mask= pat=*

The *Unique Pattern Test* option on the Dynamic RAM Menu first fills each location to be tested with a unique pattern. It then sequentially verifies each location for the expected contents. The unique pattern used at each location is generated by adding the value 0x08 to the pattern used to fill the previous location. The initial pattern used is the pattern you specify.

MATS+ Pattern Test

MATS *pass= sa= ea= dsize= mask= pat=*

The *MATS+ Pattern Test* option on the Dynamic RAM Menu first fills each location to be tested with a sequence of three patterns. The same three patterns are repeated in each group of three locations. The test then sequentially verifies each location for the expected contents.

The first pattern used is the pattern specified by you. The other two patterns are those resulting from inverting the bits of the specified pattern. The pattern 0x55555555 is used for the first pattern, and its inverse, 0xAAAAAAAA, is used for the remaining patterns if no pattern is specified.

Random Pattern Test

RAND *pass= sa= ea= dsize= mask= pat=*

The *Random Pattern Test* option on the Dynamic RAM Menu first fills each location to be tested with values produced by a random number generator. It then sequentially verifies each location for the expected contents. The pattern specified by you is used as the starting point for the random number generator.

NTA Test

NTA *pass= sa= ea= mask=*

The *NTA Test* option on the Dynamic RAM Menu is a long word test. It uses a series of patterns to verify the locations under test. The locations are initially filled with 0x00000000. As the expected contents of a location are successfully verified, the location is refilled with the next pattern in the series. The verification of a filled pattern may proceed from the lowest test address to the highest or from highest to lowest. The following table shows the address direction and patterns used during each sequential pattern test.

From Addr	*To Addr*	*Current Pattern*	*Next Pattern*
Low	High	0x00000000	0x01010101
High	Low	0x01010101	0xFFFFFFFF
Low	High	0xFFFFFFFF	0xF1F1F1F1
High	Low	0xF1F1F1F1	0x33333333
High	Low	0x33333333	0xF0F0F0F0
Low	High	0xF0F0F0F0	0x0F0F0F0F
High	Low	0x0F0F0F0F	0x55555555
Low	High	0x55555555	0xAAAAAAAA
Low	High	0xAAAAAAAA	0x05050505 then 0x88888888
High	Low	0x88888888	0x11111111
High	Low	0x11111111	0x00000000
Low	High	0x00000000	

Triangle Pattern Test

TRI *pass= sa= ea= dsize= mask= pat=*

The *Triangle Pattern Test* option on the Dynamic RAM Menu first fills each location to be tested with a unique pattern. It then sequentially verifies each location for the expected contents. The unique pattern used in each location is generated from the pattern in the previous location by shifting it left once and incrementing by one. The original pattern used in the first location under test is reinstated after each series of ten patterns.

Diagonal Pattern Test

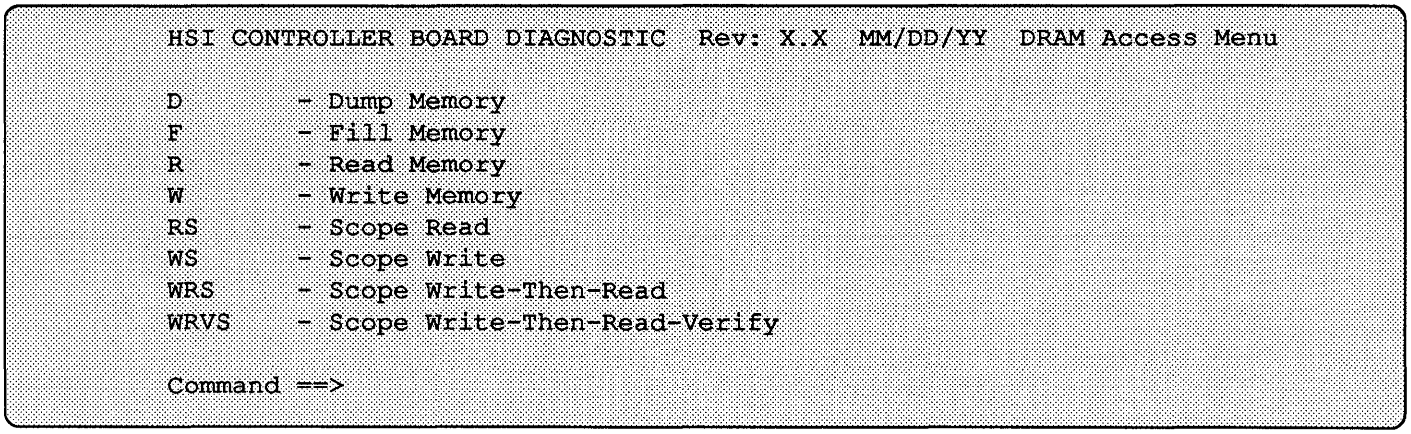
DIAG *pass= sa= ea= dsize= mask= pat=*

The *Diagonal Pattern Test* option on the Dynamic RAM Menu fills each location to be tested with a series of eight patterns. The same eight patterns are repeated through every eight locations of memory. The patterns are generated by shifting the pattern used in the previous location left one bit position. The starting pattern of the eight-pattern series is the pattern specified by you.

DRAM Access Menu

ACC

When you choose the *DRAM Access Menu* option on the Dynamic RAM Menu, this submenu displays:



```
HSI CONTROLLER BOARD DIAGNOSTIC  Rev: X.X  MM/DD/YY  DRAM Access Menu

D        - Dump Memory
F        - Fill Memory
R        - Read Memory
W        - Write Memory
RS       - Scope Read
WS       - Scope Write
WRS      - Scope Write-Then-Read
WRVS     - Scope Write-Then-Read-Verify

Command ==>
```

The options on this menu manipulate the 1 Mbyte of addressable memory on the HSI board.

D *sa= ea=*

The *Dump Memory* option on the DRAM Access Menu provides the ability to request a long word display of the contents of memory. The range of memory locations to be accessed is specified by you.

Parameter	*Description*	*Default*
sa	Hex start address	First DRAM location
ea	Hex end address	Last DRAM location

F *sa= ea=dsize= pat=*

The *Fill Memory* option on the DRAM Access Menu provides the ability to fill a range of memory locations with a specified value. The size of the memory fill may be either byte, word, or long word. The range of memory locations to be accessed is specified by you. A read-back verification is performed upon successful execution of each fill.

Parameter	*Description*	*Default*
sa	Hex start address	First DRAM location
ea	Hex end address	Last DRAM location
dsize	Byte/Word/Long word	Byte
pat	Hex fill value/0x	

R *sa=*

The *Read Memory* option on the DRAM Access Menu provides the ability to perform a long word read operation from a specified location in memory. Read access continues sequentially from the long word specified until you terminate the operation.

To display the contents of the next memory location, press [RETURN]. To terminate the read operation, enter a period (.).

Parameter	*Description*	*Default*
`sa`	Hex start address	First DRAM location

W *sa= ea= dsize=*

The *Write Memory* option on the DRAM Access Menu provides the ability to perform a byte, word, or long word write operation to a specified location in memory. Write access continues sequentially from the byte, word, or long word specified until you terminate the operation. A read-back verification is performed upon successful execution of each write.

After the contents of the current memory location are displayed, you may do one of the following:

- □ Enter new data to be written to the current location.
- □ Press [RETURN] to continue to the next location without changing the contents of the current location.
- □ Enter a period (.) to terminate the option.

Parameter	*Description*	*Default*
`sa`	Hex start address	First DRAM location
`ea`	Hex end address	Last DRAM location
`dsize`	Byte/Word/Long word	Byte

RS *sa= dsize=*

The *Scope Read* option on the DRAM Access Menu provides the ability to initiate a continuous byte, word, or long word read loop on a specified location in memory.

Parameter	*Description*	*Default*
`sa`	Hex scope address	First DRAM location
`dsize`	Byte/Word/Long word	Byte

WS *sa= dsize= pat=*

The *Scope Write* option on the DRAM Access Menu provides the ability to initiate a continuous byte, word, or long word write loop on a specified location in memory.

Parameter	*Description*	*Default*
`sa`	Hex scope address	First DRAM location
`dsize`	Byte/Word/Long word	Byte
`pat`	Hex write value	0

WRS *sa= dsize= pat=*

The *Scope Write-Then-Read* option on the DRAM Access Menu provides the ability to initiate a continuous byte, word, or long word write-then-read loop on a specified location in memory. This test provides a tight loop in which the hardware is exercised quickly and continuously. Therefore, for purposes of speed, the value read is not compared to the write pattern.

Parameter	*Description*	*Default*
`sa`	Hex scope address	First DRAM location
`dsize`	Byte/Word/Long word	Byte
`pat`	Hex write value	0

WRVS *sa= dsize= pat=*

The *Scope Write-Then-Read-Verify* option on the DRAM Access Menu provides the ability to initiate a continuous byte, word, or long word write-then-read loop on a specified location in memory. Following each write-then-read loop, the pattern read is verified against the write pattern.

Parameter	*Description*	*Default*
`sa`	Hex scope address	First DRAM location
`dsize`	Byte/Word/Long word	Byte
`pat`	Hex write value	0

Serial Controller Menu

This section describes the tests available to verify the features of both Serial Controllers A and B. The following is the menu summarizing the tests for this section of the HSI Controller Board. The Serial Controllers will be set to operate under Transparent Mode for these tests.

When you choose **S** from the Main Menu, the Serial Controller Menu is displayed:

```
HSI CONTROLLER BOARD DIAGNOSTIC   Rev: X.X   MM/DD/YY
Serial Controller Menu

All      - All Sequential Test
Quick    - Quick Sequential Test
ASCm     - Serial Controller A Scope Menu
BSCm     - Serial Controller B Scope Menu
Im       - Internal Loopback Menu
Em       - External Loopback Menu
Pm       - Port To Port Menu
ACC      - MK5025 Access Verification
ASLf     - MK5025 A Selftest
BSLf     - MK5025 B Selftest
RMK      - Read MK5025
WMK      - Write MK5025
AFcs     - MK5025 A FCS Check
BFcs     - MK5025 B FCS Check
AI       - MK5025 A Interrupt Verification
BI       - MK5025 B Interrupt Verification

Command ==>
```

All Sequential Test

A

The *All Sequential Test* option on the Serial Controller Menu causes options 3, 6–8, and 11–14 to execute in order.

Quick Sequential Test

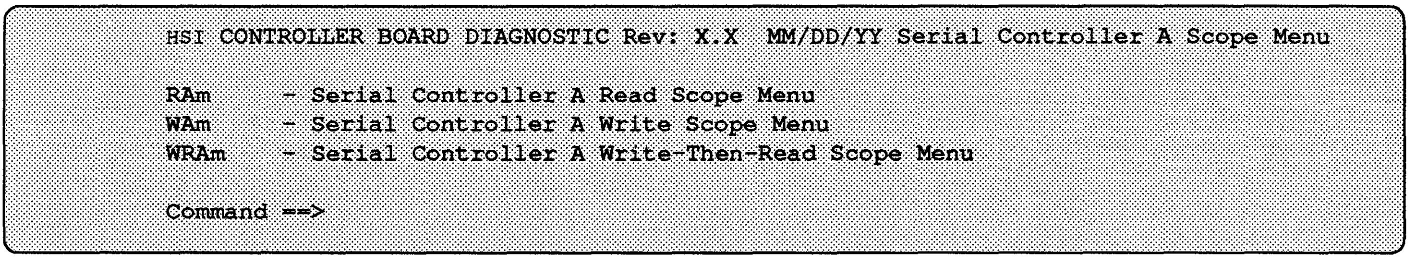
Q

The *Quick Sequential Test* option on the Serial Controller Menu causes options 7 and 8 to execute in order.

Serial Controller A Scope Menu

ASC

When you choose the *Serial Controller A Scope Menu* option, this submenu displays:

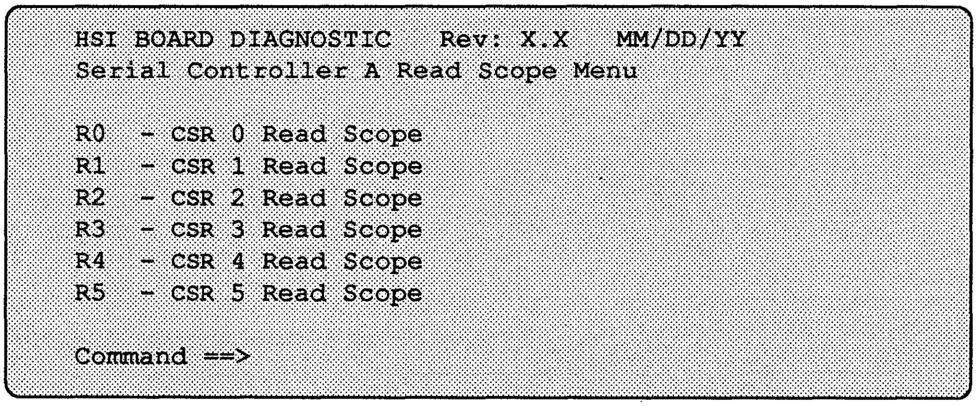


```
HSI CONTROLLER BOARD DIAGNOSTIC Rev: X.X  MM/DD/YY Serial Controller A Scope Menu

RAm     - Serial Controller A Read Scope Menu
WAm     - Serial Controller A Write Scope Menu
WRAm    - Serial Controller A Write-Then-Read Scope Menu

Command ==>
```

The options on this menu allow for continuous read, write, or write-then-read access of the MK5025 A Control and Status Registers.

RA

When you choose the *Serial Controller A Read Scope Menu* option from the Serial Controller A Scope Menu, this submenu displays:



```
HSI BOARD DIAGNOSTIC   Rev: X.X   MM/DD/YY
Serial Controller A Read Scope Menu

R0  - CSR 0 Read Scope
R1  - CSR 1 Read Scope
R2  - CSR 2 Read Scope
R3  - CSR 3 Read Scope
R4  - CSR 4 Read Scope
R5  - CSR 5 Read Scope

Command ==>
```

The options on this menu allow for continuous read access of the MK5025 A Control and Status Registers. Scope access of Control and Status Registers 0–5 is available. These options have no parameters.

WA *pat=*

When you choose the *Serial Controller A Write Scope Menu* option from the Serial Controller A Scope Menu, this submenu displays:

```
HSI BOARD DIAGNOSTIC   Rev: X.X   MM/DD/YY
Serial Controller A Write Scope Menu

W0  - CSR 0 Write Scope
W1  - CSR 1 Write Scope
W2  - CSR 2 Write Scope
W3  - CSR 3 Write Scope
W4  - CSR 4 Write Scope
W5  - CSR 5 Write Scope

Command ==>
```

The options on this menu allow for continuous write access of the MK5025 A Control and Status Registers. Scope access of Control and Status Registers 0–5 is available.

Parameter	*Description*	*Default*
pat	Word hex write pattern	0

WR *pat=*

When you choose the *Serial Controller A Write-Then-Read Scope Menu* option from the Serial Controller A Scope Menu, this submenu displays:

```
HSI BOARD DIAGNOSTIC   Rev: X.X   MM/DD/YY
Serial Controller A Write-Then-Read Scope Menu

WR0 - CSR 0 Write-Then-Read Scope
WR1 - CSR 1 Write-Then-Read Scope
WR2 - CSR 2 Write-Then-Read Scope
WR3 - CSR 3 Write-Then-Read Scope
WR4 - CSR 4 Write-Then-Read Scope
WR5 - CSR 5 Write-Then-Read Scope

Command ==>
```

The options on this menu allow for continuous write-then-read access of the MK5025 A Control and Status Registers. Scope access of Control and Status Registers 0–5 is available. This test provides a tight loop in which the hardware is exercised quickly and continuously. Therefore, for purposes of speed, the value read is not compared to the write pattern.

Parameter	*Description*	*Default*
pat	Word hex write pattern	0

Serial Controller B Scope Menu

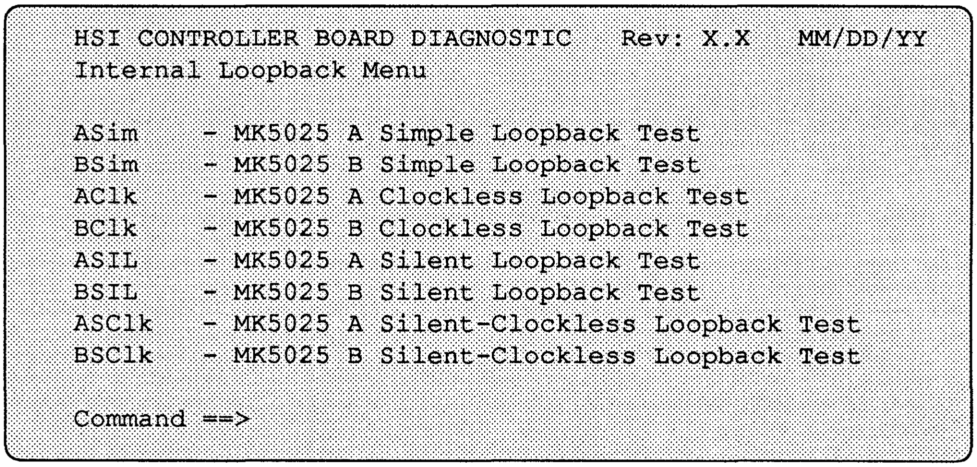
BSC

When you choose the *Serial Controller B Scope Menu* option from the Serial Controller Menu, a choice of submenus is displayed. The options on the submenus perform the same tests for Controller B that the Serial Controller A Scope Menu options, just described, perform for Controller A.

Internal Loopback Menu

I

When you choose the *Internal Loopback Menu* option from the Serial Controller Menu, this submenu displays:



```
HSI CONTROLLER BOARD DIAGNOSTIC    Rev: X.X    MM/DD/YY
Internal Loopback Menu

ASim     - MK5025 A Simple Loopback Test
BSim     - MK5025 B Simple Loopback Test
AClk     - MK5025 A Clockless Loopback Test
BClk     - MK5025 B Clockless Loopback Test
ASIL     - MK5025 A Silent Loopback Test
BSIL     - MK5025 B Silent Loopback Test
ASClk    - MK5025 A Silent-Clockless Loopback Test
BSClk    - MK5025 B Silent-Clockless Loopback Test

Command ==>
```

The options on this menu perform a series of internal Transmission and Reception of Information frame(s) through each of the Serial Controllers A and B. The controllers are placed in several internal loopback configurations, and a transmit and receive attempt is made with each loopback configuration. The following loopback configurations are used:

- Simple Loopback
- Clockless Loopback
- Silent Loopback
- Silent Clockless Loopback

The on-board clock is used for the Simple Loopback and Silent Loopback tests. The SYSCLK of the serial controllers is used for the Clockless Loopback and the Silent Clockless Loopback tests.

You may specify the test data to be sent with the frame. You may either specify the test data pattern or select a sequence pattern to be sent with the frame. The default test mode sends a 1 Kbyte frame containing a test pattern of sequential numerical data.

The following options may be specified with these tests:

Parameter	*Description*	*Default*
pat	Transmit pattern	Sequential numeric pattern starting at 0 and ending at transmit length-1
cnt	Transmit length	1024

If the *transmit pattern* is specified, the *transmit length* is equal to the byte count of the *transmit pattern.*

External Loopback Menu

E

When you choose the *External Loopback Menu* option from the Serial Controller Menu, this submenu displays:

```
HSI CONTROLLER BOARD DIAGNOSTIC   Rev: X.X   MM/DD/YY
External Loopback Menu

AR  - RS-449 External Loopback Thru Port A
BR  - RS-449 External Loopback Thru Port B
AV  - V.35 External Loopback Thru Port A
BV  - V.35 External Loopback Thru Port B

Command ==>
```

This test attempts to transmit and receive information frame(s) through the same port. The transmission may be performed through either RS-449 or V.35 ports.

The test requires an external loopback connection which feeds the transmit clock and transmit data lines of the port under test into the receive clock and receive data lines of the same port. It also requires that the on-board clock be used.

You may specify the test data to be sent with the frame. You may either specify the test data pattern or select a sequence pattern to be sent with the frame. The default test mode sends a 1 Kbyte frame containing a test pattern of sequential numerical data.

The following options may be specified with these tests:

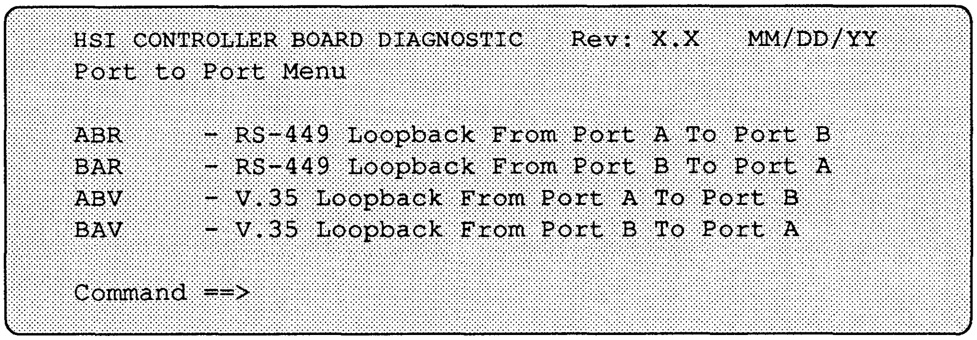
Parameter	*Description*	*Default*
pat	Transmit pattern	Sequential numeric pattern starting at 0 and ending at transmit length-1
cnt	Transmit length	1024

If the *transmit pattern* is specified, the *transmit length* is equal to the byte count of the *transmit pattern.*

Port To Port Menu

P

When you choose the *Port To Port Menu* option from the Serial Controller Menu, this submenu displays:



```
HSI CONTROLLER BOARD DIAGNOSTIC   Rev: X.X   MM/DD/YY
Port to Port Menu

ABR      - RS-449 Loopback From Port A To Port B
BAR      - RS-449 Loopback From Port B To Port A
ABV      - V.35 Loopback From Port A To Port B
BAV      - V.35 Loopback From Port B To Port A

Command ==>
```

This test attempts to transmit information frame(s) from one HSI port to another. The transmission may be performed through either RS-449 or V.35 ports.

The test requires an external loopback connection which feeds the transmit clock and transmit data lines of one port into the receive clock and receive data lines of the other port. It also requires that the on-board clock be used for both Serial Controllers A and B.

You may specify the test data to be sent with the frame. You may either specify the test data pattern or select a sequence pattern to be sent with the frames. The default test mode sends a 1 Kbyte frame containing a test pattern of sequential numerical data.

The following options may be specified with these tests:

Parameter	*Description*	*Default*
pat	Transmit pattern	Sequential numeric pattern starting at 0 and ending at transmit length-1
cnt	Transmit length	1024

If the *transmit pattern* is specified, the *transmit length* is equal to the byte count of the *transmit pattern*.

ACC

The *MK5025 Access Verification* option on the Serial Controller Menu provides verification of the data lines connected to Serial Controllers A and B. The data lines are checked for proper connections by a sequence of write-then-read operations on the Control and Status Registers of each MK5025. The write-then-read operations are performed with a series of patterns designed to help identify possible misconnections.

This option has no parameters.

ASL

The *MK5025 A Selftest* option on the Serial Controller Menu initiates the Mostek 5025 built-in selftest. It sends a selftest primitive to the controller and waits for a successful or unsuccessful selftest response through Control and Status Register 1 (CSR 1).

This option has no parameters.

BSL

The *MK5025 B Selftest* option on the Serial Controller Menu initiates the Mostek 5025 built-in selftest. It sends a selftest primitive to the controller and waits for a successful or unsuccessful selftest response through Control and Status Register 1 (CSR 1).

This option has no parameters.

RMK

The *Read MK5025* option on the Serial Controller Menu provides a utility which allows you to display the contents of the Command and Status Registers for Serial Controllers A and B. This option has no parameters.

WMK *mk= csr= pat=*

The *Write MK5025* option on the Serial Controller Menu provides a utility which allows you to write a 16-bit value into the Command and Status Registers of either Serial Controller A or B. The parameters are required.

Parameter	*Description*	*Default*
mk	Controller A or B	None
csr	0–5	None
pat	Hex word write pattern	None

AF

The *MK5025 A FCS Check* option on the Serial Controller Menu verifies the capability of Serial Controller A to detect and report a Frame Checksum error.

First the Disable Transmit FCS (DTFCS) bit of the Initialization Block Mode Register is asserted. Asserting this bit results in packets being transmitted without a Checksum.

Next an Information Frame is transmitted in Silent Internal Loopback mode. An FCS error should be reported upon reception of this packet. An error is reported if the FCS error count in the Serial Controller's Initialization Block does not increase upon receiving this packet.

This option has no parameters.

BF

The *MK5025 B FCS Check* option on the Serial Controller Menu performs the same test on Serial Controller B that the MK5025 A FCS Check option performs on Controller A. The option has no parameters.

AI

The *MK5025 A Interrupt Verification* option on the Serial Controller Menu verifies the capability of Serial Controller A to generate an interrupt.

The test attempts to generate an interrupt by internally transmitting a packet in Simple Loopback mode. A Transmit interrupt from the Serial Controller is expected when the packet is transmitted. An error is reported if the Serial Controller interrupt is not detected within a timeout period. The on-board clock is used during this test. An interrupt level must be enabled on the HSI board through the appropriate DIP switch in order to run the test. For information on the default DIP switch settings, refer to *Configuration Procedures for the SunLink High-speed Serial Interface Board.*

This option has no parameters.

BI

The *MK5025 B Interrupt Verification* option on the Serial Controller Menu performs the same test on Serial Controller B that the MK5025 A Interrupt Verification option performs on Controller A. The option has no parameters.

Programmable Divider Menu

This section describes the tests available to exercise the Programmable Timer on the HSI board.

When you choose **P** from the Main Menu, the Programmable Divider Menu is displayed:

```
HSI CONTROLLER BOARD DIAGNOSTIC   Rev: X.X   MM/DD/YY
Programmable Divider Menu

SW  - Programmable Divider Clock Sweep Verification
SA  - Set Programmable Divider A
SB  - Set Programmable Divider B

Command ==>
```

SW

The *Programmable Divider Clock Sweep Verification* option on the Programmable Divider Menu causes a sweep of clock rates within a specific range to be generated from each of the Programmable Dividers A and B. The difference between one clock rate and the next in the sweep is a fixed value. The time interval between the setting of one clock rate and the next change is also a fixed value. The sweep is made from high to low clock rates.

The purpose of the clock sweep is to provide a change in the Programmable Divider rates which may be monitored externally with an instrument such as an oscilloscope. The monitoring of such a change would verify that the Programmable Dividers may be influenced by changes in their divide values. It would also verify the "Select On-Board Clocks for Serial Port A/B" signals in Port C of CIO A and the "Divider A/B External Trigger" signals in Port A of CIO A.

This option has no parameters.

SA *val=*

The *Set Programmable Divider A* option on the Programmable Divider Menu allows you to change the divide value of Programmable Divider A. You may use the following parameter:

Parameter	*Description*	*Default*
`val`	Word hex divide value	None

SB *val=*

The *Set Programmable Divider B* option on the Programmable Divider Menu allows you to change the divide value of Programmable Divider B. You may use the following parameter:

Parameter	*Description*	*Default*
`val`	Word hex divide value	None

Exit

X

The *Exit* option on the Main Menu exits the HSI Diagnostic and returns to the Exec Main Menu.

18.9. Messages

The HSI Diagnostic tests return the following successful completion or error messages.

CIO Ports A, B, and C and Control Registers Access Verification

Successful Completion Messages

```
CIO A Access Verification Successful.  Pass=#.
CIO B Access Verification Successful.  Pass=#.
```

Error Messages

```
3000  CIO Access Test  Pass=#  Error Accessing CIO A/B Port A
With Pattern #
3000  CIO Access Test  Pass=#  Error Accessing CIO A/B Port B
With Pattern #
3000  CIO Access Test  Pass=#  Error Accessing CIO A/B Port C
With Pattern #
3000  CIO Access Test  Pass=#  Error Accessing CIO A/B CONTROL
REG With Pattern #
```

V.35 Modem Control Test Menu

Successful Completion Messages

```
MK-A Modem Control Verification of DTR-DCD Thru V.35 Successful.
Pass=#.
MK-A Modem Control Verification of RTS-CTS Thru V.35 Successful.
Pass=#.
MK-A Modem Control Verification of DTR-DSR Thru V.35 Successful.
Pass=#.
MK-B Modem Control Verification of DTR-DCD Thru V.35 Successful.
Pass=#.
MK-B Modem Control Verification of RTS-CTS Thru V.35 Successful.
Pass=#.
MK-B Modem Control Verification of DTR-DSR Thru V.35 Successful
Pass=#.
```

Error Messages

```
3060 DTR-DCD Test Thru V.35   Pass=#  MK A DCD Did Not Set.
3060 RTS-CTS Test Thru V.35   Pass=#  MK A DCD Did Not Set.
3060 DTR-DSR Test Thru V.35   Pass=#  MK A DCD Did Not Set.
3060 DTR-DCD Test Thru V.35   Pass=#  MK A DCD Did Not Clear.
3060 RTS-CTS Test Thru V.35   Pass=#  MK A DCD Did Not Clear.
3060 DTR-DSR Test Thru V.35   Pass=#  MK A DCD Did Not Clear.
3060 DTR-DCD Test Thru RS-449  Pass=#  MK B DCD Did Not Set.
3060 RTS-CTS Test Thru RS-449  Pass=#  MK B DCD Did Not Set.
3060 DTR-DSR Test Thru RS-449  Pass=#  MK B DCD Did Not Set.
3060 DTR-DCD Test Thru RS-449  Pass=#  MK B DCD Did Not Clear.
3060 RTS-CTS Test Thru RS-449  Pass=#  MK B DCD Did Not Clear.
3060 DTR-DSR Test Thru RS-449  Pass=#  MK B DCD Did Not Clear.
```

RS-449 Modem Control Test Menu

Successful Completion Messages

```
MK-A Modem Control Verification of DTR-DCD Thru RS-449 Successful.
Pass=#.
MK-A Modem Control Verification of RTS-CTS Thru RS-449 Successful.
Pass=#.
MK-A Modem Control Verification of DTR-DSR Thru RS-449 Successful.
Pass=#.
MK-B Modem Control Verification of DTR-DCD Thru RS-449 Successful.
Pass=#.
MK-B Modem Control Verification of RTS-CTS Thru RS-449 Successful.
Pass=#.
MK-B Modem Control Verification of DTR-DSR Thru RS-449 Successful.
Pass=#.
```

Error Messages

```
3060 DTR-DCD Test Thru RS-449  Pass=#  MK A DCD Did Not Set.
3060 RTS-CTS Test Thru RS-449  Pass=#  MK A DCD Did Not Set.
3060 DTR-DSR Test Thru RS-449  Pass=#  MK A DCD Did Not Set.
3060 DTR-DCD Test Thru RS-449  Pass=#  MK A DCD Did Not Clear.
3060 RTS-CTS Test Thru RS-449  Pass=#  MK A DCD Did Not Clear.
3060 DTR-DSR Test Thru RS-449  Pass=#  MK A DCD Did Not Clear.
3060 DTR-DCD Test Thru RS-449  Pass=#  MK B DCD Did Not Set.
3060 RTS-CTS Test Thru RS-449  Pass=#  MK B DCD Did Not Set.
3060 DTR-DSR Test Thru RS-449  Pass=#  MK B DCD Did Not Set.
3060 DTR-DCD Test Thru RS-449  Pass=#  MK B DCD Did Not Clear.
3060 RTS-CTS Test Thru RS-449  Pass=#  MK B DCD Did Not Clear.
3060 DTR-DSR Test Thru RS-449  Pass=#  MK B DCD Did Not Clear.
```

Interrupt Verification

Successful Completion Messages

```
Interrupt Verification Successful.  Pass=#.
```

Error Messages

```
3080  Interrupt Verification Test  Pass=#  Interrupt Did Not Occur.
```

Parity Interrupt Verification

Successful Completion Messages

```
Parity Interrupt Verification Successful.  Pass=#.
```

Error Messages

```
3090 Parity Interrupt Verification Pass=# Parity Error Bit will not clear.
3090 Parity Interrupt Verification Pass=# Interrupt Did Not Occur.
3090 Parity Interrupt Verification Pass=# Parity Error Bit Did Not Set.
```

Modem Control Interrupt Verification Menu

Successful Completion Messages

```
DCD on MK-A Thru RS-449 Interrupt Verification Successful.
Pass=#.
DSR on MK-A Thru RS-449 Interrupt Verification Successful.
Pass=#.
CTS on MK-A Thru RS-449 Interrupt Verification Successful.
Pass=#.
DCD on MK-A Thru V.35 Interrupt Verification Successful.
Pass=#.
DSR on MK-A Thru V.35 Interrupt Verification Successful.
Pass=#.
CTS on MK-A Thru V.35 Interrupt Verification Successful.
Pass=#.
DCD on MK-B Thru RS-449 Interrupt Verification Successful.
Pass=#.
DSR on MK-B Thru RS-449 Interrupt Verification Successful.
Pass=#.
CTS on MK-B Thru RS-449 Interrupt Verification Successful.
Pass=#.
DCD on MK-B Thru V.35 Interrupt Verification Successful.
Pass=#.
DSR on MK-B Thru V.35 Interrupt Verification Successful.
Pass=#.
CTS on MK-B Thru V.35 Interrupt Verification Successful.
Pass=#.
```

Error Messages

```
3100 RS-449 Modem Interrupt Verification On MK-A Pass=#
Unable to clear DCD.
3100 RS-449 Modem Interrupt Verification On MK-A Pass=#
DCD Did Not Set.
3100 RS-449 Modem Interrupt Verification On MK-A Pass=#
DCD Did Not Set But Interrupt Occurred.
3100 RS-449 Modem Interrupt Verification On MK-A Pass=#
Interrupt Did Not Occur.

(Same form of Error Messages for V.35, DSR, CTS, MK-B.)
```

Dynamic RAM Address, Pattern, and NTA Tests

Error Messages

```
3140 "Test Name" Memory Test  Parity Error  Pass=#
Start Addr= #, Ending Addr = #, Current Addr = #
Expected Value = #, Actual Value = #
3140 "Test Name" Memory Test  Access Error  Pass=#
Start Addr= #, Ending Addr = #, Current Addr = #
Expected Value = #, Actual Value = #
```

MK5025 Register Access Verification

Successful Completion Messages

```
MK5025 A Access Verification Successful.  Pass=#.
MK5025 B Access Verification Successful.  Pass=#.
```

Error Messages

```
3170 MK Access Verification  Pass=#  Error Accessing MK-A With
Pattern #
3170 MK Access Verification  Pass=#  Error Accessing MK-B With
Pattern #
```

MK5025 A/B Selftest

Successful Completion Messages

```
MK A Selftest Successful.
MK B Selftest Successful.
```

Error Messages

```
1030  MK Selftest  Pass=#  Unsuccessful MK5025 A selftest.
1040  MK Selftest  Pass=#  PAV not set during MK5025 A selftest.
3110  MK Selftest  Pass=#  Unable to RE_INIT MK-A after sefltest.
1030  MK Selftest  Pass=#  Unsuccessful MK5025 B selftest.
1040  MK Selftest  Pass=#  PAV not set during MK5025 B selftest.
3110  MK Selftest  Pass=#  Unable to RE_INIT MK-B after sefltest.
```

MK5025 A/B FCS Check

Successful Completion Messages

```
FCS Verification on MK5025 A Successful.  Pass=#.
FCS Verification on MK5025 B Successful.  Pass=#.
```

Error Messages

```
3020  FCS Check  Unable to RE_INIT MK5025 A Before FCS Check
Attempt.
3020  FCS Check  Pass=#  Unable to force FCS error on MK5025 A.
3020  FCS Check  Unable to RE_INIT MK5025 B Before FCS Check
Attempt.
3020  FCS Check  Pass=#  Unable to force FCS error on MK5025 B.
```

Serial Controller A/B Interrupt Verification

Successful Completion Messages

```
MK-A Interrupt Verification Successful.  Pass=#.
MK-B Interrupt Verification Successful.  Pass=#.
```

Error Messages

```
3110 MK-A Interrupt Verification Pass=# Unable to RE-INIT MK Before Attempt.
3050 MK-A Interrupt Verification Pass=# Unable to transmit.
3050 MK-A Interrupt Verification Pass=# Interrupt Causing Condition Did Not Occur.
3050 MK-A Interrupt Verification Pass=# Interrupt Bit Did Not Set.
3050 MK-A Interrupt Verification Pass=# Interrupt Did Not Occur.
3110 MK-B Interrupt Verification Pass=# Unable to RE-INIT MK Before Attempt.
3050 MK-B Interrupt Verification Pass=# Unable to transmit.
3050 MK-B Interrupt Verification Pass=# Interrupt Causing Condition Did Not Occur.
3050 MK-B Interrupt Verification Pass=# Interrupt Bit Did Not Set.
3050 MK-B Interrupt Verification Pass=# Interrupt Did Not Occur.
```

18.10. Glossary

HSI	High Speed Serial Interface Controller Board.
MK5025 A	HSI Serial Controller A.
MK5025 B	HSI Serial Controller B.

19

IPI Disk Subsystem Diagnostic

19

IPI Disk Subsystem Diagnostic

19.1. General Description

The Intelligent Peripheral Interface (IPI) Subsystem Diagnostic tests the IPI disk controller and disk drive. The diagnostic runs under the Exec, and the interface of the IPI Disk Subsystem Diagnostic is consistent with the menu system and command syntax supported by the Exec.

19.2. What This Chapter Contains

Following an overview of the diagnostic and a list of required hardware, the Main Test Menu and submenus are discussed. Menu selections and optional arguments are listed, along with brief descriptions. The end of the chapter contains a list of error messages and a glossary.

19.3. Overview of the Diagnostic

The IPI Disk Subsystem Diagnostic has been designed to test both the IPI controller and the IPI drive. The diagnostic includes two sets of tests: the controller tests with no drive required, and the combined controller/drive tests. Each set of tests verifies approximately 80% of the hardware under test. In combination, both sets of tests verify the IPI Disk Subsystem with a 95% level of confidence.

The parameters of each test are given default values upon execution. Online help is provided. The IPI Disk Subsystem Diagnostic also generates and stores error messages for later retrieval.

19.4. Hardware Requirements

The following hardware is required to run the IPI Disk Subsystem Diagnostic:

- A Sun-3 or Sun-4 system with extra VME interface 9U slot.
- A monitor.
- A keyboard.
- A working IPI-2 disk controller board. (As many as four may be installed.)
- An IPI disk drive. (As many as ten per controller may be connected.)
- A boot device (local disk, local tape, or remote disk over Ethernet).

19.5. User Interface

The user interface of the IPI Disk Subsystem Diagnostic adheres to the standards of the Exec menu system. Each test may be selected from a menu by typing the letter or letters displayed in uppercase in the column on the left side of the menu.

Additional parameters and options may be specified on the command line. For a complete discussion of the Exec command line syntax, refer to Chapter 2, "Using the SunDiagnostic Executive."

A Main Test Menu and three submenus are provided. The options on the submenus provide for testing the controller separately, as well as both the controller and disk combined. A submenu also provides a subset of the IPI-3 command package for special testing and configuration. All, Default, and Quick Test sequences are provided on the Main Test Menu.

To move back one level in the menu hierarchy, press ESC then press RETURN. To leave the IPI Disk Subsystem Diagnostic and return to the Exec, enter **ex** when the Main Test Menu is displayed.

During any test, if an error occurs and the environment variable sco= is set to ON, the test will display an error message then enter a scope loop. The following message is displayed during the scope loop:

```
Scope On Error (Press Any Key To Exit Loop) ...
```

Whenever a controller panic error occurs, the host will try to reset the controller and continue, unless the environment variable sto= is set to 1. In that case, the test will pause before resetting, to allow you to capture the controller panic condition. The following message is displayed:

```
Halt Before Reset Controller (Press Any Key To Continue) ...
```

At the end of each test, if the command variable ba= is set to ON, the screen will be cleared to redisplay the menu, but if ba= is set to OFF, the following message is displayed:

```
Pause (Press Any Key To Continue) ...
```

You can terminate a test in progress by pressing any key on the keyboard. The pressed key takes effect between transfers of command packets from the diagnostic to the controller.

To display online Help for any menu option, enter the following on the command line:

? *option_name*

19.6. Starting the Diagnostic

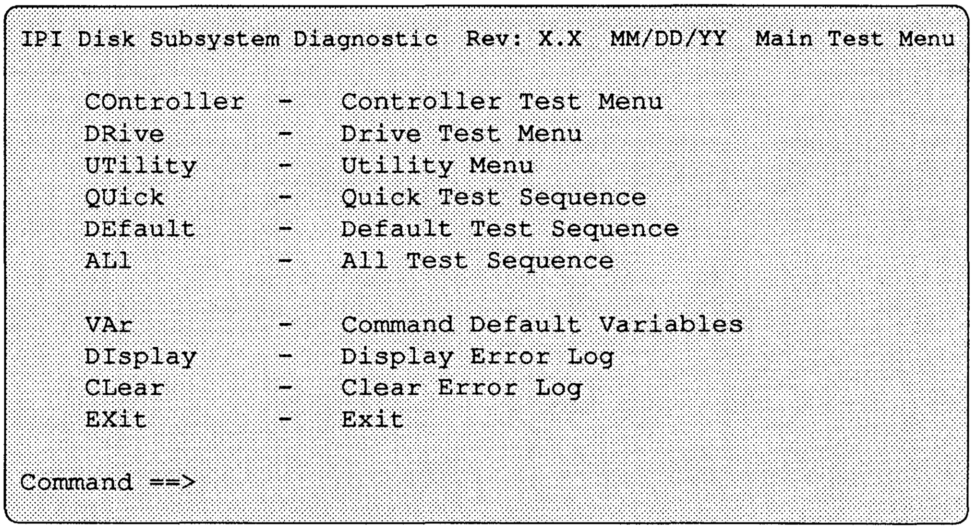
For information on starting the Exec, refer to Chapter 2, "Using the SunDiagnostic Executive." After you have started the Exec, choose the IPI Disk Subsystem Diagnostic from the Diagnostics Menu.

19.7. The Diagnostic Menus

This section of the chapter provides a modular description of the IPI Disk Subsystem Diagnostic, beginning with the Main Test Menu and working down through the options available on each of the submenus. A list of error messages for each test is given in the section entitled "Error Messages."

Main Test Menu

The Main Test Menu, which displays when you start the IPI Disk Subsystem Diagnostic, provides access to the submenus of the individual tests:



```
IPI Disk Subsystem Diagnostic  Rev: X.X  MM/DD/YY  Main Test Menu

    COntroller   -   Controller Test Menu
    DRive        -   Drive Test Menu
    UTility      -   Utility Menu
    QUick        -   Quick Test Sequence
    DEfault      -   Default Test Sequence
    ALl          -   All Test Sequence

    VAr          -   Command Default Variables
    DIsplay      -   Display Error Log
    CLear        -   Clear Error Log
    EXit         -   Exit

Command ==>
```

Command Variables

The options on the submenus, as well as the Quick, Default, and All Test options on the Main Test Menu, take optional parameters, or command variables. The variables may be displayed or set by using the Command Default Variables command on the Main Test Menu. You can also specify the variables by typing them as parameters on the command line when you run individual tests. If no parameters are specified, the default values are used. The meanings of the command variables, and their default values, are shown in the following table:

Parameter	*Description*	*Default*
`COntroller`	Controller reference address (0–3).	0
`DRive`	Drive reference address (0–7).	0
`CYlinder`	Range of cylinders. DIAG (diagnostic cylinders) ALL (all cylinders) *<range of cylinders>* (for example, 11–7D09)	DIAG
`WArning`	Mode which displays a warning when a non-diagnostic disk area will be overwritten (can be ON or OFF).	ON
`BAtch`	When ON, batch mode allows you to run a sequence of tests from a script file without intervention. If this mode is OFF, a pause occur after the execution of each command before the screen is cleared to redisplay the menu.	OFF

Parameter	*Description*	*Default*
`INfo`	Level of verbosity of informational messages displayed during test (1–3). The higher the level, the more information displayed during testing. All level 2 messages are preceded by "**," and all level 3 messages are preceded by "***." Level 3 messages require knowledge of IPI-3 commands and response interface.	1
`TEst`	Test number (meaning specific to command).	0 (all)
`DAta`	Data pattern (0–FFFFFFFF).	Varies by test
`BLock`	Logical address block in a drive, specified from 0 to the last logical block in the drive. Each drive may contain a different number of logical blocks.	None
`NUmber`	Number of logical blocks to be accessed, from 1 to the maximum number of logical blocks in a drive.	None
`MOde`	Mode of operation (meaning specific to command).	None

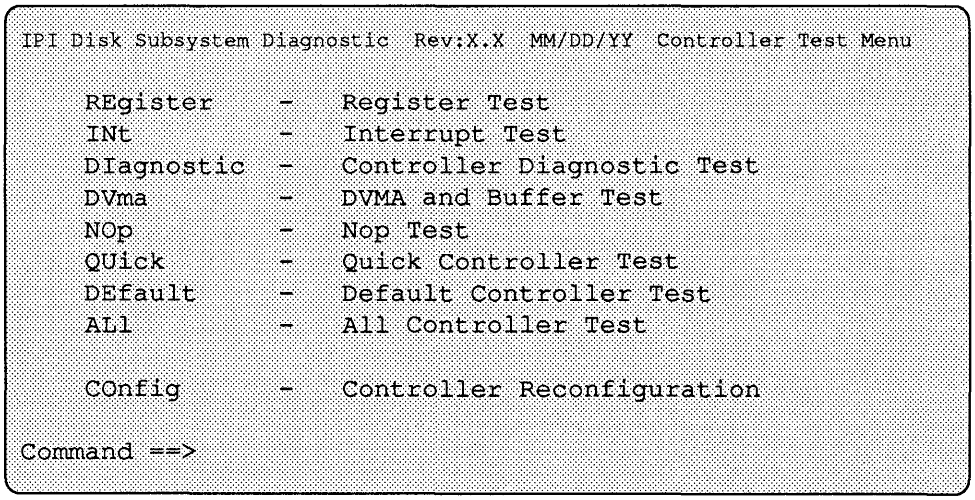
Local Environmental Variables

The commands in the IPI Subsystem Diagnostic are designed to run with the Exec global environmental variables. You also have the option of using a subset of those global environmental variables to control the execution of tests in the IPI Subsystem Diagnostic. These local environmental variables may be entered on the command line in the same way that you enter command variables. The local variables affect the tests in the IPI Disk Subsystem only; they do not change the Exec global environmental variables. The following table summarizes the local environmental variables.

Parameter	*Description*	*Default*
`PASs`	Pass count (1–7FFFFFFF).	1
`SOFt`	Soft error retry count (0–7FFFFFFF).	0
`SCOpe`	Scope on error (ON or OFF).	OFF
`STOp`	Stop on error count (1–7FFFFFFF).	1

Controller Test Menu

When you choose **CO** from the Main Test Menu, the Controller Test Menu is displayed:



```
IPI Disk Subsystem Diagnostic  Rev:X.X  MM/DD/YY  Controller Test Menu

     REgister     -    Register Test
     INt          -    Interrupt Test
     DIagnostic   -    Controller Diagnostic Test
     DVma         -    DVMA and Buffer Test
     NOp          -    Nop Test
     QUick        -    Quick Controller Test
     DEfault      -    Default Controller Test
     ALl          -    All Controller Test

     COnfig       -    Controller Reconfiguration

Command ==>
```

The options on the Controller Test Menu allow you to test functions of the IPI-2 Disk Controller that do not require a disk to be attached.

RE *co= pas= sof= sco= sto=*

The *Register Test* option on the Controller Test Menu performs a write-and-read test on the shared VME registers resident on the IPI controller board.

IN *co= pas= sof= sco= sto=*

The *Interrupt Test* option on the Controller Test Menu verifies the ability of the controller to generate an interrupt to the host correctly when enabled and not to generate an interrupt when disabled. The test is run with two status conditions: Command Register Not Busy and Response Register Valid.

DI *co= pas= sof= sco= sto=*

The *Controller Diagnostic Test* option on the Controller Test Menu issues a request to the controller to execute the on-board PROM diagnostic selftests. This diagnostic includes the following tests:

1. LEDs Walking Test
2. EPROM Checksum Test
3. LEDs Register Test
4. SCC Port A Register Test
5. SCC Port B Register Test
6. SCC Port A Loopback Test
7. SCC Port B Loopback Test
8. Exec RAM Inverse Test
9. Exec RAM Address Test

10. Exec RAM 3-Pattern Test
11. Exec RAM March Test
12. Exec RAM Byte Alignment Test
13. CIO Register Test
14. Read-Ahead Buffer Inverse Test
15. Read-Ahead Buffer Address Test
16. Read-Ahead Buffer 3-Pattern Test
17. Read-Ahead Buffer March Test
18. VME Register Test
19. IPI LCA Configuration Test
20. IPI Register Test
21. IPI DMA Loopback Test
22. IPI Quick Control Sequence Loopback Test
23. IPI Bus Control Sequence Loopback Test
24. EEPROM Checksum Test
25. Bus Error Interrupt Test
26. Address Error Interrupt Test
27. SCC Interrupt Test
28. CIO Interrupt Test
29. Read-Ahead Buffer Parity Error Interrupt Test
30. IPI Interrupt Test

During the selftest, the test messages are displayed only to a terminal connected to the SCC port A of the controller. Any error that occurs before the CIO Register Test is considered as fatal error; in this case, the controller will not respond to the host. Any error that occurs after the CIO Register Test will be reported to the host accordingly. For information on error messages associated with this test, see the *IPI-2 Disk Controller Boot PROM Diagnostic Design Document.*

DV *co= pas= sof= sco= sto=*

The *DVMA and Buffer Test* option on the Controller Test Menu verifies that the controller can successfully perform the VME DMA transfers between the CPU host main memory (VME space) and the controller read-ahead buffer.

This test issues the DVMA read buffer command to transfer a block of data from the host to the controller, and then issues the DVMA buffer write command to transfer data from the controller back to the host. Finally, it verifies that data were transferred properly. The sizes of data transfer used are 4, 8, 10, 20, 40, 80, 100, 200, 400, 800, 1000, 2000, 4000, 8000, and FC00 hexadecimal. The hexadecimal patterns used for each transfer are 00000000, FFFFFFFF, 55555555, AAAAAAAA, 5A972C5A, and an incremental pattern. All transfers are performed at buffer locations 0, 40000, 80000, and C0000 hexadecimal.

NO *co= pas= sof= sco= sto=*
The *Nop Test* option on the Controller Test Menu issues an NOP command to verify that the controller can receive a command and respond to the host correctly.

QU *co= pas= sof= sco= sto=*
The *Quick Controller Test* option on the Controller Test Menu performs a quick sequence of three controller tests:

1. Controller Register Test
2. Controller Interrupt Test
3. Controller Nop Test

DE *co= pas= sof= sco= sto=*
The *Default Controller Test* option on the Controller Test Menu performs a default sequence of four tests:

1. Controller Register Test
2. Controller Interrupt Test
3. Controller DVMA and Buffer Test
4. Controller Nop Test

AL *co= pas= sof= sco= sto=*
The *All Controller Test* option on the Controller Test Menu performs a sequence of all the controller tests, in order:

1. Controller Register Test
2. Controller Interrupt Test
3. Controller Diagnostic Selftest
4. Controller DVMA and Buffer Test
5. Controller Nop Test

CO *co=*
The *Controller Reconfiguration* option on the Controller Test Menu allows you to reconfigure the functional characteristics of the IPI controller board. When you choose this command, the following messages are displayed:

```
Controller Current Configuration Values:

@ DMA Burst Size Is ?? Long Words
@ Inter-Burst Latency Is ? × 3.2 Microseconds
@ Bus Request Level Is ?
@ VME Interrupt Level Is ?
@ VME Block Mode Is ?

>Do you wish to change these values (Y/N)?
```

If you want to leave the values unchanged, type **N**. You return to the Controller Test Menu. If you do want to change any of the values, type **Y**. The following messages are displayed:

```
Enter new value, or press <Return> to leave unchanged

>DMA Burst Size (8,10,20,40,80,100,200,400)?
>Inter-Burst Latency (0-F)?
>Bus Request Level (0-3)?
@ VME Interrupt Level (1-7)?
@ VME Block Mode (ON/OFF)?
```

Following each message, you can enter a new hexadecimal value to change the configuration. New values go into effect only when the controller is reset. To reset the controller, do one of the following:

- Choose the Controller Diagnostic Test option on the Controller Test Menu.
- Perform a **k1** or **k2** system reset.
- Perform a "hard" reset by turning the system off then on.

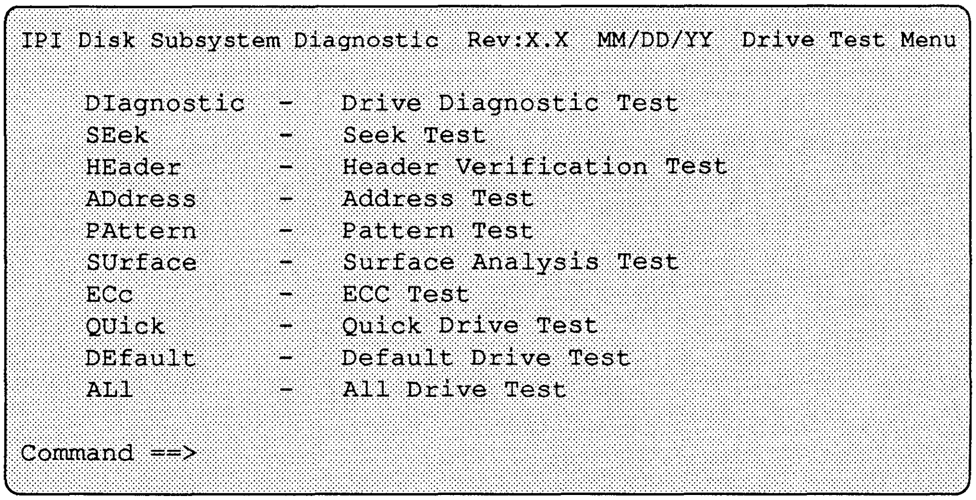
CAUTION **If you choose to change the bus request level, a warning message will display. A change to the bus request level can cause the controller to arbitrate incorrectly for the bus when DMA operations are attempted. Therefore, it is recommended that you exercise caution when deciding to change the bus request level.**

On most systems, the bus arbiter is set to request level 3 and cannot be reset. If you do reset the bus request level to an incompatible value, a timeout error will be generated. To correct the error, first reset the bus request level to match the bus request level of the CPU. After the bus level is changed, reset the controller by choosing the Controller Diagnostic Test option on the Controller Test Menu or by performing a **k1** or k2 system reset.

To leave a configuration value unchanged and move to the next value, press [RETURN].

Drive Test Menu

When you choose **DR** from the Main Test Menu, the Drive Test Menu is displayed:



```
IPI Disk Subsystem Diagnostic  Rev:X.X  MM/DD/YY  Drive Test Menu

     DIagnostic    -   Drive Diagnostic Test
     SEek          -   Seek Test
     HEader        -   Header Verification Test
     ADdress       -   Address Test
     PAttern       -   Pattern Test
     SUrface       -   Surface Analysis Test
     ECc           -   ECC Test
     QUick         -   Quick Drive Test
     DEfault       -   Default Drive Test
     ALl           -   All Drive Test

Command ==>
```

The options on the Drive Test Menu allow you to test the IPI disk drives through functions of the IPI-2 Disk Controller.

DI *co= dr= pas= sof= sco= sto=*

The *Drive Diagnostic Test* option on the Drive Test Menu issues a request to the specified drive to perform a selftest. The selftest capability depends upon the specific drive under test.

SE *co= dr= te= pas= sof= sco= sto=*

The *Seek Test* option on the Drive Test Menu performs several seek tests and a seek timing test. The value of `te=` may be one of the following numbers:

Test Number	*Description*
0	All seek tests
1	Sequential seek test
2	Butterfly seek test
3	Random seek test
4	Timing seek test

- □ A value of 0 (all seek tests) is the default.
- □ The sequential seek test begins at the first cylinder and performs the seek, incrementing by one until it reaches the last cylinder. It then continues in reverse, performing the seek from the last cylinder, decrementing by one until it reaches the first cylinder.
- □ The butterfly seek test performs the seek test at locations *start plus K* and *end minus K*, where K begins at zero and extends to *end minus start plus one*. For example, if there are ten cylinders under test, the seek operation will be performed on the cylinders in the following order: 0–9, 1–8, 2–7, 3–6, 4–5, 5–4, 6–3, 7–2, 8–1, and 9–0.
- □ The random seek test performs the random cylinder seek N times, where N is the number of cylinders on a drive.

- The timing seek test records the seeking time elapsed from the first cylinder to the last, as the test steps through each cylinder.

HE *co= dr= pas= sof= sco= sto=*
The *Header Verification Test* option on the Drive Test Menu performs a read operation on all the logical blocks in the drive to verify that the header information is correct.

AD *co= dr= cy= pas= sof= sco= sto=*
The *Address Test* option on the Drive Test Menu first writes the address pattern to all logical blocks of the specified cylinders. It then reads the pattern back, verifying the address of each block.

If the specified cylinder range (cy=) includes cylinders outside the diagnostic cylinders, and if the warning mode (wa=) is set to ON, the following message is displayed:

```
WARNING: DATA IN CYLINDERS ???? THRU ???? WILL BE DESTROYED.
>PROCEED (Y/N)?
```

If you do not want the data in the specified cylinders to be destroyed, type **N**. You return to the Drive Test Menu. If you do want to proceed with the test, destroying the data in the non-diagnostic cylinders, type **Y**.

PA *co= dr= cy= da= pas= sof= sco= sto=*
The *Pattern Test* option on the Drive Test Menu writes and verifies data patterns on the first, middle, and last logical blocks of the first, middle, and last cylinders of the specified cylinder range.

The default pattern for da= is 0, which is translated to all of the following patterns:

```
0x00000000
0xFFFFFFFF
0xEBD6EBD6
0xD7ADD7AD
0xAF5BAF5B
0x5EB75EB7
0x6DB66DB6
```

If the specified cylinder range (cy=) includes cylinders outside the diagnostic cylinders, and if the warning mode (wa=) is set to ON, the following message is displayed:

```
WARNING: DATA IN CYLINDERS ???? THRU ???? WILL BE DESTROYED.
>PROCEED (Y/N)?
```

If you do not want the data in the specified cylinders to be destroyed, type **N**. You return to the Drive Test Menu. If you do want to proceed with the test, destroying the data in the non-diagnostic cylinders, type **Y**.

SU *co= dr= cy= da= pas= sof= sco= sto=*

The *Surface Analysis Test* option on the Drive Test Menu writes and verifies the specified data patterns on all logical blocks of the specified cylinders.

The default pattern for `da=` is 0x6DB66DB6. If the specified cylinder range (`cy=`) includes cylinders outside the diagnostic cylinders, and if the warning mode (`wa=`) is set to ON, the following message is displayed:

```
WARNING: DATA IN CYLINDERS ???? THRU ???? WILL BE DESTROYED.
>PROCEED (Y/N)?
```

If you do not want the data in the specified cylinders to be destroyed, type **N**. You return to the Drive Test Menu. If you do want to proceed with the test, destroying the data in the non-diagnostic cylinders, type **Y**.

EC *co= dr= te= pas= sof= sco= sto=*

The *ECC Test* option on the Drive Test Menu performs the ECC test on a sector of a diagnostic cylinder. The value of `te=` may be one of the following numbers:

Test Number	*Description*
0	ECC tests 1 and 2 (default setting)
1	ECC error detection test
2	ECC error correction test
3	ECC extensive error correction test

With support of the firmware, this test can detect all errors as long as 89 bits, and correct all errors as long as 17 bits in a single burst. The ECC error detection test consists of the following steps:

1. Verify a diagnostic sector with random data.
2. Read sector with ECC field included.
3. Force an ECC error that exceeds two double-octets but is within an 89-bit length at a random location within the sector.
4. Write sector to disk with ECC error included.
5. Read back sector and verify that the ECC detects the error but does not correct it.
6. Repeat steps 1 through 5 with ECC error burst from 2 to 89 bits.

The ECC error correction test consists of the following steps:

1. Verify a diagnostic sector with random data.
2. Read sector with ECC field included.
3. Force a random ECC error within a 17-bit length at a random location within the sector.

4. Write sector to disk with ECC error included.
5. Read back sector and verify that the ECC detects and corrects the error.
6. Repeat steps 1 through 5 with ECC error burst from 1 to 17 bits in length. Each bit group is done once randomly.

The ECC extensive error correction test includes the following steps:

1. Verify a diagnostic sector with random data.
2. Read sector with ECC field included.
3. Force a random ECC error within a 17-bit length at a random location within the sector.
4. Write sector to disk with ECC error included.
5. Read back sector and verify that the ECC detects and corrects the error.
6. Repeat steps 1 through 5 with all combinations of the 17-bit error burst, beginning at a random bit position.

`QU` *co= dr= cy= pas= sof= sco= sto=*
The *Quick Drive Test* option on the Drive Test Menu performs a quick sequence of the following drive tests:

1. Drive Sequential Seek Test
2. Drive Address Test
3. Drive Pattern 00000000 Test
4. Drive Pattern FFFFFFFF Test
5. Drive Pattern 6DB66DB6 Test
6. Drive Pattern EBD6EBD6 Test
7. Drive Pattern D7ADD7AD Test
8. Drive Pattern AF5BAF5B Test
9. Drive Pattern 5EB75EB7 Test

`DE` *co= dr= cy= pas= sof= sco= sto=*
The *Default Drive Test* option on the Drive Test Menu performs the default sequence of eleven drive tests:

1. Drive Diagnostic Selftest
2. Drive Butterfly Seek Test
3. Drive Address Test
4. Drive Pattern 00000000 Test
5. Drive Pattern FFFFFFFF Test

6. Drive Pattern 6BD66BD6 Test
7. Drive Pattern EBD6EBD6 Test
8. Drive Pattern D7ADD7AD Test
9. Drive Pattern AF5BAF5B Test
10. Drive Pattern 5EB75EB7 Test
11. ECC Error Correction Test

AL *co= dr= cy= pas= sof= sco= sto=*

The *All Drive Test* option on the Drive Test Menu performs a sequence of all drive tests in the following order:

1. Drive Diagnostic Selftest
2. Drive Sequential Seek Test
3. Drive Butterfly Seek Test
4. Drive Random Seek Test
5. Drive Timing Seek Test
6. Drive Header Verification Test
7. Drive Address Test
8. Drive Pattern 00000000 Test
9. Drive Pattern FFFFFFFF Test
10. Drive Pattern 6BD66BD6 Test
11. Drive Pattern EBD6EBD6 Test
12. Drive Pattern D7ADD7AD Test
13. Drive Pattern AF5BAF5B Test
14. Drive Pattern 5EB75EB7 Test
15. Drive Surface Analysis Test
16. ECC Error Detection Test
17. ECC Error Correction Test

Utility Menu

When you choose **UT** from the Main Test Menu, the Utility Menu is displayed:

```
IPI Disk Subsystem Diagnostic  Rev:X.X  MM/DD/YY  Utility Menu

    ATtribute    -   Read/Initialize/Load Attributes
    FOrmat       -   Format Drive
    OPerate      -   Set Disk Operating Mode
    SEek         -   Drive Position Control
    TEll         -   Report Drive Position
    WRite        -   Write Pattern to Drive
    REad         -   Read and Verify Pattern
    STatus       -   Report Addressee Status
    RDefect      -   Read Defective List
    WDefect      -   Write Defective List
    MAp          -   Reallocate Disk Block
    VMe          -   VME Register Write/Read
    CYlinder     -   Cylinder Address of DataBlock

Command ==>
```

The options on the Utility Menu provide a subset of IPI-3/VME commands for special test and configuration purposes. For a complete discussion of the IPI-3/VME commands, see the *Sun IPI-3/VME Command Set for the IPI-2 Disk Controller*.

AT *co= dr= mo=*

The *Read/Initialize/Load Attributes* option on the Utility Menu allows you to read, initialize, or change the attributes of the drive controller. The possible values of mo= are shown in the following table:

Value	*Description*
0	Read attributes (default setting)
1	Load new attributes

If you specify mo=0, the following messages are displayed for you to select a specific attribute:

```
1) Vendor ID
2) Size of Data Blocks
3) Size of Physical Blocks
4) Total Number of Disk Data Blocks
5) Total Number of Disk Physical Blocks
6) Disk Data Block Size Supported
7) Disk Physical Block Size Supported
8) Pad With Fill Characters
9) Physical Disk Configuration
A) Addressee Configuration
B) Facilities Attached to Slave
C) IPI Controller Reconfiguration

Select Attribute to Read (1-C) ==>
```

If you specify mo=1, the following messages are displayed:

```
1) Pad With Fill Characters
2) IPI Controller Reconfiguration

Select Attribute to Load (1-2) ==>
```

You may select the specific attribute for which to load new values. All of the fields of the attribute you select are displayed for you to enter the new values.

FO *co= dr= mo= cy=*

The *Format Drive* option on the Utility Menu allows you to format all or part of a drive, using either the working defect list or the manufacturer's defect list. Formatting with the manufacturer's defect list is equivalent to initializing the drive. The drive is formatted using only the drive's factory defect list. Formatting with the working defect list uses the manufacturer's defect list, as well as defects found during normal operation or diagnostic testing.

If you choose to format the entire disk using the working defect list, you must specify a range of all cylinders, either by using the *Command Default Variables* option on the Main Menu or by typing cy=all on the command line. If you format with the manufacturer's defect list, any range specified is ignored, and the entire disk is formatted.

The meanings of parameters specific to this command are shown in the following table:

Parameter	*Description*	*Default*
mo	Format using working defect list (0). Format using manufacturer's defect list (1).	0

Parameter	*Description*	*Default*
`cy`	Range of cylinders to format when using working defect list. Ignored when using manufacturer's defect list.	DIAG

When formatting begins, the following message is displayed:

```
Format Using defect_list
Cylinder Range:  start_cylinder to end_cylinder
```

If you format with the manufacturer's list, the following message alerts you that the entire disk is about to be formatted:

```
All cylinders are formatted when using Mfg.'s list!
```

If the specified number of cylinders includes blocks outside the diagnostic area, and if the warning mode (`wa=`) is set to ON, the following message is displayed:

```
WARNING: DATA IN BLOCKS ???? THRU ???? WILL BE DESTROYED.
>PROCEED (Y/N)?
```

If you do not want the data to be destroyed, type **N**. You return to the Utility Menu. If you do want to proceed with the formatting, type **Y**.

OP *co= dr= mo=*

The *Set Disk Operating Mode* option on the Utility Menu allows you to force a drive into a specific operating mode. The possible values of `mo=` are shown in the following table:

Value	*Description*
0	Reset the drive to zero (default mode)
1	Spin down
2	Spin up

SE *co= dr= bl=*

The *Drive Position Control* option on the Utility Menu performs a seek to a specified logical block in a drive.

TE *co= dr=*

The *Report Drive Position* option on the Utility Menu reports the current drive position in a logical block address.

NOTE *If you enter a Report Drive Position command immediately after a Drive Position Control command, a different logical address block is reported because the disk is continuously spinning.*

WR *co= dr= bl= nu= da=*

The *Write Pattern to Drive* option on the Utility Menu allows you to write a specified pattern to a number of logical blocks in a drive. The meanings of some parameters specific to this command are shown in the following table:

Parameter	*Description*	*Default*
`bl`	Logical block address to be read.	0
`nu`	Number of logical blocks to be read.	1
`da`	32-bit hex pattern to be written.	00000000

RE *co= dr= bl= nu= da=*

The *Read and Verify Pattern* option on the Utility Menu allows you to read a number of logical blocks in a drive and compare them to a specified pattern. The meanings of some parameters specific to this command are shown in the following table:

Parameter	*Description*	*Default*
`bl`	Logical block address to be read.	0
`nu`	Number of logical blocks to be read.	1
`da`	32-bit hex pattern to be compared to blocks read.	00000000

ST *co= dr=*

The *Report Addressee Status* option on the Utility Menu reports the current status condition of a drive. One or more of the following conditions is reported: operational, not operational, ready, not ready, drive switched to another port, port neutral, available, busy, status pending, active, and inactive.

RD *co= dr=*

The *Read Defective List* option on the Utility Menu allows you to read a defective list from a drive. When you select this option, the following menu is displayed:

```
1) Manufacturer Defect List
2) Working Defect List
3) Formatted Defect List
4) Alternate Sector List

Select Defect List To Read (1-4) ==>
```

- □ Manufacturer Defect List — Defects found by the drive manufacturer.
- □ Working Defect List — Defects found during normal operation or diagnostic testing. Defects can be added to this list with the *Write Defective List* or *Reallocate Disk Block* utilities or by using the SunOS formatting utility.

- Formatted Defect List — Defects that had been on the working defect list and that were added after a format operation. Immediately after a drive is formatted, the formatted defect list should include all the defects on the manufacturer's defect list plus the defects from the working list.

- Alternate Sector List — Any defective sectors that have been reassigned with the *Reallocate Disk Block* command.

Select one of the defective list formats or press RETURN to return to the Utility Menu. If you select a defective list, the following message is displayed:

```
Sequential (0) Or Chronological (1)?
```

If the defective list that you select is not an alternate sector list, the following message is displayed:

```
Track (0) Or Sector (1) Format?
```

WD *co= dr=*

The *Write Defective List* option on the Utility Menu allows you to clear, append, or create the working defective list. You cannot use this command to write to the manufacturer's defective list, however. When you select this option, the following menu is displayed:

```
1) Create Working Defect List
2) Append To Working Defect List
3) Clear Working Defect List

Select Action (1-3) ==>
```

If you choose to create or append to the working defective list, the following message is displayed:

```
Track (0) Or Sector (1) Format?
```

If you select track format, the following messages are displayed:

```
>Cylinder?
>Track?
>Byte Offset Into Track?
>Length Of Defect?
```

If you select sector format, the following messages are displayed:

```
>Cylinder?
>Track?
>Sector After Index?
>Byte Offset Within Sector?
>Length Of Defect?
```

These requests for information repeat until you press [RETURN] without entering a value. You must supply all of the information requested before the working defective list can be written.

MA *co= dr= bl=*

The *Reallocate Disk Block* option on the Utility Menu allows you to reallocate a bad logical disk block. The bad block is remapped to another physical disk sector. The value of `bl=` specifies the block to be reallocated; the default is 0.

VM *co=*

The *VME Register Write/Read* option on the Utility Menu allows you to perform scope write and read operations to a VME register offset address a specified number of times. The command ignores any bus errors while writing or reading the specified register. When you select this option, the following menu is displayed:

```
1) Read VME Register
2) Write To VME Register
3) Write Then Read VME Register

Select Action (1-3) ==>
```

After you make a selection, the following message is displayed:

```
VME Register Offset Address (0-3FC)?
```

If you choose to write or write then read the VME register, the following message is displayed:

```
Value To Write (0-FFFFFFFF)?
```

Finally, you are asked the pass count for the scope operation:

```
Number Of Times To Scope (1-*)?
```

If you choose to read or write then read the VME register, only the first value read is displayed, even if more than one read operation is performed.

CY *co= dr= bl=*

The *Cylinder Address of DataBlock* option on the Utility Menu allows you to determine the cylinder address of a data block. This information is useful when you want to use the Format Drive option to format only the cylinder containing a bad data block. If you specify the number of the bad block as the value of `bl=`, this option will return the number of the cylinder in which the block is located.

Quick Test Sequence

QU *co= dr= cy= pas= sof= sco= sto=*

The *Quick Test Sequence* option on the Main Test Menu executes the following sequence of drive and controller tests:

1. Controller Register Test
2. Controller Nop Test
3. Controller Interrupt Test
4. Drive Sequential Seek Test
5. Drive Address Test
6. Drive Pattern Test (all default patterns)

Default Test Sequence

DE *co= dr= cy= pas= sof= sco= sto=*

The *Default Test Sequence* option on the Main Test Menu executes the following sequence of controller and drive tests:

1. Controller Register Test
2. Controller Nop Test
3. Controller Interrupt Test
4. Controller Diagnostic Selftest
5. Drive Diagnostic Selftest
6. Drive Butterfly Seek Test
7. Drive Address Test
8. Drive Pattern Test (all default patterns)
9. ECC Error Correction Test

All Test Sequence

AL *co= dr= cy= pas= sof= sco= sto=*

The *All Test Sequence* option on the Main Test Menu executes all controller and drive tests in the following sequence:

1. Controller Register Test
2. Controller Nop Test
3. Controller Interrupt Test

4. Controller Diagnostic Selftest
5. Controller DVMA and Buffer Test
6. Drive Diagnostic Selftest
7. Drive Sequential Seek Test
8. Drive Butterfly Seek Test
9. Drive Random Seek Test
10. Drive Timing Seek Test
11. Drive Header Verification Test
12. Drive Address Test
13. Drive Pattern Test (all default patterns)
14. Drive Surface Analysis Test
15. ECC Error Detection Test
16. ECC Error Correction Test

Command Default Variables

VA *co= dr= cy= wa= ba= in= pas= sof= sco= sto=*

The *Command Default Variables* option on the Main Test Menu allows you to set or display the current default values of the command variables. If no parameter is specified, the default values of all command variables are displayed.

You can specify values for command variables with this command or, if a specific command takes a command variable, you can specify the value of the variable when you issue that command. If you specify the value of a command variable when you issue the command, the value is in effect for that command only. You can change the default setting of a command variable only by using the Command Default Variables command.

Display Error Log

DI

The *Display Error Log* option on the Main Test Menu displays the contents of the IPI Subsystem Diagnostic error log.

Clear Error Log

CL

The *Clear Error Log* option on the Main Test Menu clears the IPI Subsystem Diagnostic error log.

Exit

EX

The *Exit* option on the Main Test Menu leaves the IPI Subsystem Diagnostic and returns to the Exec.

19.8. Error Messages

This section contains a listing and interpretation of the error messages produced by the IPI Disk Subsystem Diagnostic.

```
Error: controller ???? does not exist
```

Interpretation: Controller specified for test does not seem to exist. A bus error occurs when probing an address of the controller.

```
Error: controller ???, ID expected ??, ID read ???
```

Interpretation: Some controller exists but the ID register value does not indicate that it is an IPI board.

```
Error: controller ??? indicates that drive ??? does not exist
```

Interpretation: The specified controller indicates that the specified drive is not attached to the controller.

```
Error: VME register at VME base address + ???????? not accessible
```

Interpretation: An unexpected bus error occurs when attempting to read or write to a VME register with the specified offset address.

```
Error: VME register at VME base address + ????, write ????, read ????
```

Interpretation: The Controller Register Test failed, and the data read back does not compare with the data written.

```
Error: command register write ?????, response register read ?????
```

Interpretation: The Controller Register Test failed, and data read from the response register does not compare with data written into the command register. Only the least significant 24 bits are checked.

```
Error: DVMA buffer address ???, write ????, read ????
```

Interpretation: The Controller DVMA Test failed, and that the DVMA data read does not match with the DVMA data write.

```
Error: fail to compare block ????, byte ????, exp ????, obs ????
```

Interpretation: A disk write read operation failed. The data read back from the specified block does not match the data written into the block at the specified byte position.

```
Error: controller interrupt occurs when disabled
```

Interpretation: The Controller Interrupt Test failed, and an unexpected interrupt occurred.

```
Error: controller interrupt did not occur exactly one time
```

Interpretation: The Controller Interrupt Test failed. No interrupt occurs or two or more interrupts occur when exactly one interrupt is expected.

```
Error: expecting response packet not received
```

Interpretation: A response status has been received from the controller with no error indication, but no response packet is indicated when it is expected by a given command.

```
Error: command packet ID ????, response register ????
```

Interpretation: A command was given to the controller, but a wrong response has been received.

```
Error: controller not ready, status ????
```

Interpretation: The controller indicates that it is not ready to communicate with the host.

```
Error: command register busy timeout, status ????????
```

Interpretation: The controller status register indicates that the command register is busy for too long.

```
Error: controller response timeout, status ????????
```

Interpretation: A command has been sent to the controller, and the controller does not respond to the host for a given time.

```
Error: VME DMA bus error occurs
```

Interpretation: Controller panic — a bus error occurs during DVMA transfer.

```
Error: VME DMA timeout error occurs
```

Interpretation: Controller panic — VME DMA has not been acknowledged.

```
Error: controller panic fault code ????????
```

Interpretation: Controller panic with specified condition.

```
Error: fail to write block ????????
```

Interpretation: The controller indicates that it encounters an error upon writing to the indicated logical block address after repeated attempts.

```
Error: fail to read block ????????
```

Interpretation: The controller indicates that it encounters an error upon reading the indicated logical block address after repeated attempts.

```
Error: ECC status does not indicate detection of error
```

Interpretation: An ECC error was generated, but the controller does not indicate a detection of the ECC error.

```
Error: ECC status does not indicate correction of error
```

Interpretation: A correctable ECC error was generated, but the controller does not indicate the correction of an ECC error.

```
Error: ECC bad correction, burst ????, exp ????, obs ????
```

Interpretation: A correctable ECC error was generated and the controller did indicate ECC error correction, but the data was not properly corrected.

```
Error: fail to map physical address ????????
```

Interpretation: The host was unable to map the specified physical address to VME space.

```
Error: fail to unmap virtual address ????????
```

Interpretation: The host was unable to unmap a virtual address.

```
Error: fail to map virtual address ????????
```

Interpretation: The host was unable to map a virtual address.

```
Error: fail to allocate ???????? memory bytes
```

Interpretation: The host was unable to allocate enough memory for test.

```
Error: fail to allocate ???????? bytes of DVMA space
```

Interpretation: The host was unable to allocate enough DVMA space for test.

```
Error: diagnostic exception ????????
```

Interpretation: The controller response indicates that the drive has failed the selftest diagnostic. This is an IPI-3 error response packet.

```
Error: alternate port exception ????????
```

Interpretation: The controller response indicates that the error occurs because of a dual port situation. This is an IPI-3 error response packet.

```
Error: command exception ????????
```

Interpretation: The controller response indicates that the host command packet has a format error, usually caused by failure of a DVMA command packet transfer. This is an IPI-3 error response packet.

```
Error: machine exception ????????
```

Interpretation: The controller response indicates failure of a hardware function. This is an IPI-3 error response packet.

```
Error: intervention required ????????
```

Interpretation: The controller response indicates intervention is required for this command. This is an IPI-3 error response packet.

```
Error: conditional success ????????
```

Interpretation: The controller response indicates that there was an error during disk operation but it was recovered. This is an IPI-3 error response packet.

19.9. Glossary

CIO The Zilog 8536 counter/timer and I/O chip. Signals from DMA controllers, as well as certain synchronous communication signals, if enabled in the CIO, can cause the CIO chip to generate a VMEbus interrupt.

DMA Direct Memory Access.

ECC Error Checking and Correction.

IPI	Intelligent Peripheral Interface.
Nop	No operation.
SCC	Serial Communications Controller chip.
LCA	Logic Cell Array.

20

Keyboard Diagnostic

20

Keyboard Diagnostic

The keyboard test program runs under the SunDiagnostic Executive. It tests the functionality of the keyboard on Sun-3 and Sun-4 systems.

20.1. Requirements

The diagnostic requires a standard Sun monitor or a 1024 x 1024 monitor, and a Type-3 or Type-4 keyboard.

The test takes about 3 minutes, and requires the operator to participate.

20.2. Description

To use the diagnostic:

Enter **k** from the SunDiagnostic Executive diagnostics menu.

When the diagnostic starts, it clears the screen, then displays a drawing of a keyboard. The examples below depict the Type-3 and Type-4 keyboard layout.

Figure 20-1 *Type-3 Keyboard*

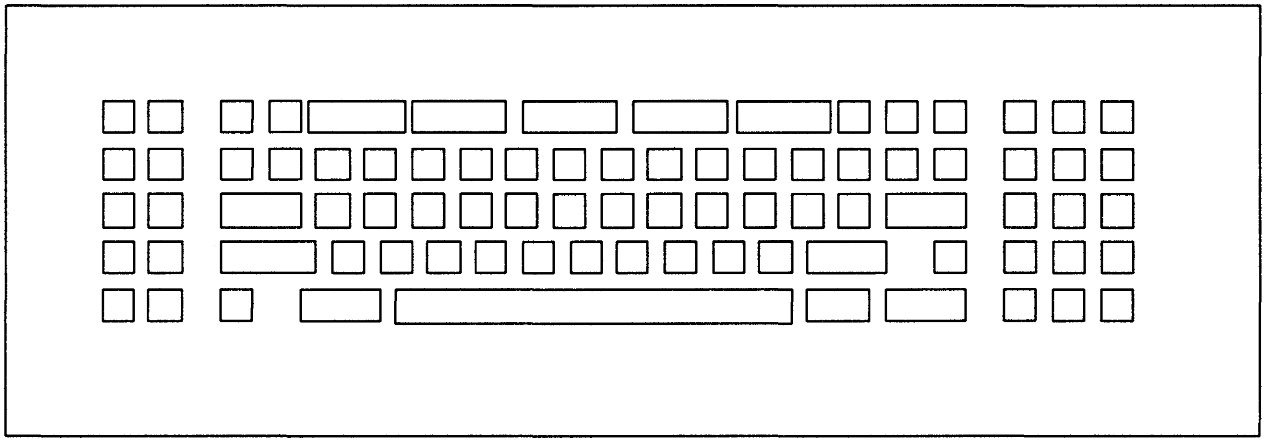
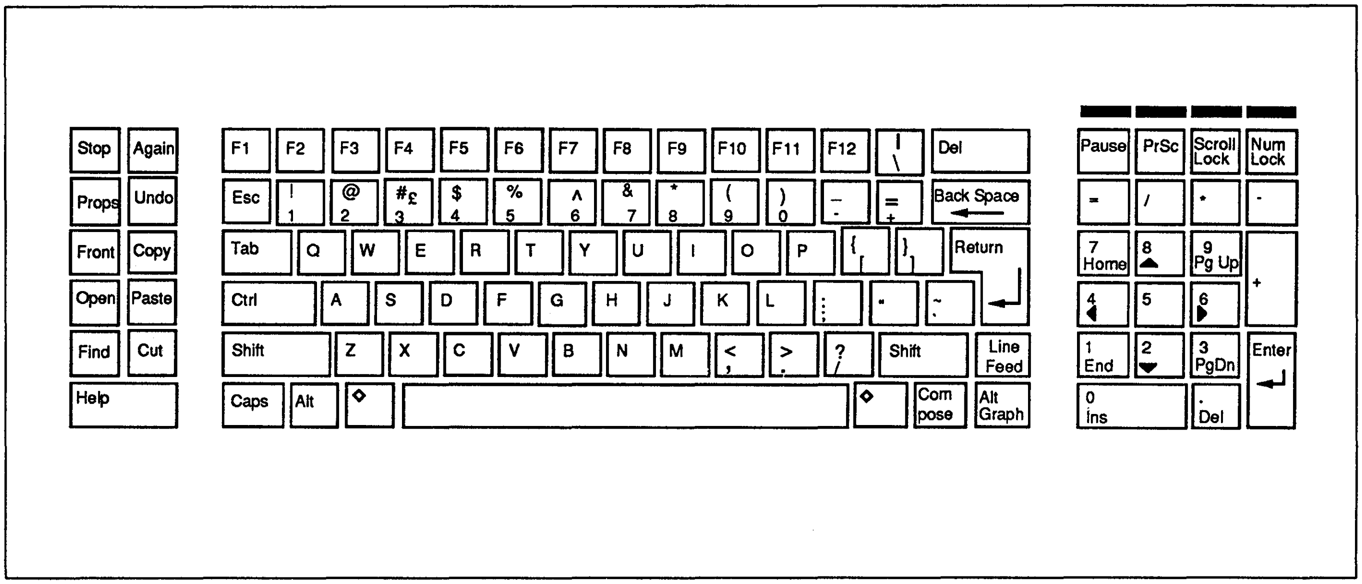


Figure 20-2 *Type-4 Keyboard*



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. On a Type-4 keyboard, the `L1` and `R3` designations are on the sides of the keys.

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.

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.)*

When testing a Type-4 keyboard, after the keyboard click test, the diagnostic tests the LED indicators that appear in the upper right section of the keyboard. The indicators are used to signal when the "Capslock", "Compose" "Scrollock" and "Numlock" Type-4 keyboard features are in use.

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.**

21

MCP/ALM2 Diagnostic

21

MCP/ALM2 Diagnostic

21.1. 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.

21.2. 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 21-1 *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 21-2 *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		

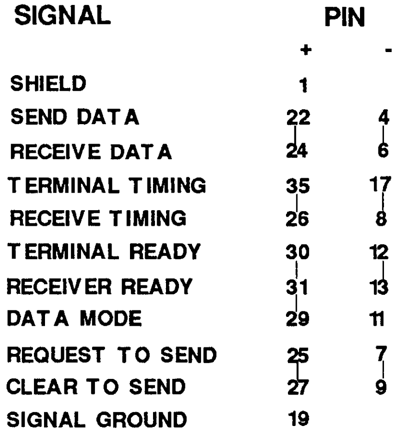
The following figure shows the RS232 signals on synchronous/asynchronous port 0 with the DB50 pin numbers. Ports 0 through 3 are synchronous/asynchronous ports on the MCP board. Please note that although this illustration seems to indicate a single port looping back to itself, you may also use these signals in paired cables. On the MCP version of the board, synchronous RS232 signals are available on ribbon cable connectors J9 and J11.

Figure 21-3 *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 21-4 *RS449 Loopback Signals, The Two RS449 Synchronous Ports*

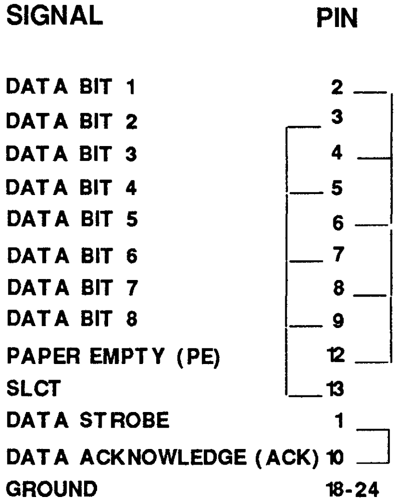


SIGNAL	PIN +	PIN -
SHIELD	1	
SEND DATA	22	4
RECEIVE DATA	24	6
TERMINAL TIMING	35	17
RECEIVE TIMING	26	8
TERMINAL READY	30	12
RECEIVER READY	31	13
DATA MODE	29	11
REQUEST TO SEND	25	7
CLEAR TO SEND	27	9
SIGNAL GROUND	19	

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 21-5 *Parallel Printer Port Signals, DB25 Plug With Loopback*



SIGNAL	PIN
DATA BIT 1	2
DATA BIT 2	3
DATA BIT 3	4
DATA BIT 4	5
DATA BIT 5	6
DATA BIT 6	7
DATA BIT 7	8
DATA BIT 8	9
PAPER EMPTY (PE)	12
SLCT	13
DATA STROBE	1
DATA ACKNOWLEDGE (ACK)	10
GROUND	18-24

21.3. 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.

21.4. 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.

21.5. 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.

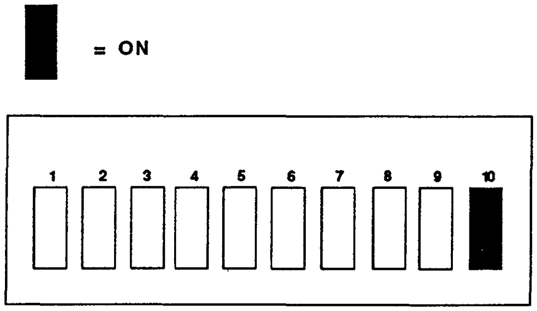
The Main Menu

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:

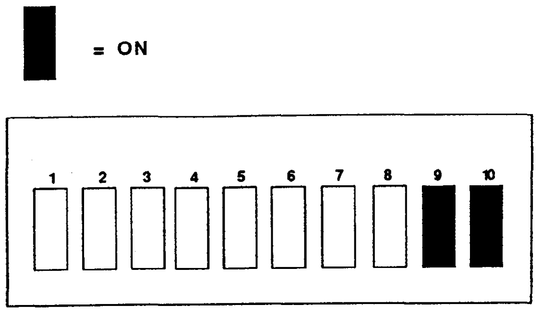
```
Hoe     Halt On Error.           [current]
GOe     Go On Error (don't halt).
A       Test Board 0.  Phys address hex 01000000
B       Test Board 1.  Phys address hex 01010000
C       Test Board 2.  Phys address hex 01020000
D       Test Board 3.  Phys address hex 01030000
Va      Show Virtual Addrsses.
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).

Figure 21-6 *ALM-2/MCP DIP Switches*



The highest 8 bits of a 32-bit physical address (showing 01).



The next highest 8 bits of a 32-bit physical address (showing 03).

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.*

Hoe

The *halt on error* selection from the main menu causes the diagnostic to halt when an error occurs.

GOe

The *go on error* selection from the main menu causes the diagnostic to continue running when an error occurs.

A

The *Test Board 0* selection from the main menu executes tests on board 0.

B

The *Test Board 1* selection from the main menu executes tests on board 1.

C

The *Test Board 2* selection from the main menu executes tests on board 2.

D

The *Test Board 3* selection from the main menu executes tests on board 3.

M

The *Manufacturing default* selection from the main menu attempts to test up to four boards.

va

The *Show Virtual Addresses* selection from the main menu is intended for use in debugging. This option displays the virtual address the diagnostic executive has given to certain addressable parts of the ALM-2/MCP.

ba=

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. If successful, 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.
Xoff        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.
```

The following test describes the Basic Test Menu choices.

M

This option selects the *Manufacturing Board Test Defaults* menu. The default configuration pairs all sixteen RS-232 ports, with port 0 paired with port 1, port 2 paired with port 3, and so on for all ports. It also simultaneously pairs ports 0 and 1 as RS-449 ports. The RS-449 signal drivers and receivers on the board are separate from those for RS-232, making this arrangement possible. The Manufacturing default also tells the diagnostic that the parallel printer port has its special loopback plug on it, rather than a real printer.

DLF

Selecting *Describe Loopback Fixtures* from the basic test menu, displays a menu of loopback descriptions. The loopback description menu and options follow this section.

DEF

Selecting *Default Tests* from the basic test menu, selects and executes the default tests.

D

If you select the *DEVCTL Register Test* from the Basic Test Menu, you invoke the Device Control Register test. The ALM-2/MCP has a virtual DEVCTL (Device Control) Register for each port. For the asynchronous-only ports, only two bits of the 8-bit register are valid. For the RS232 synchronous or asynchronous ports, three bits are valid. For the RS449 synchronous or asynchronous ports, five bits are valid. The DEVCTL Register test sets to 0 and 1 each valid DEVCTL Register bit in turn and reads them back to verify it.

I

This option executes the *Interrupt Vector Register Test.* This test writes all possible patterns to the 8-bit Interrupt Vector Register and reads them back to compare with expected values.

R

Entering **R** invokes the *Common RAM Test.* Common RAM is 14K x 32-bit static random access memory. Data to be transmitted out a port goes from the workstation to this buffer. Then, with the help of DMA, the SCC s move the data out to the serial ports. For synchronous operation, the SCC s move received data to Common RAM for the workstation to read.

This test does memory testing of every location. Address-unique patterns test the ALM-2/MCP's address bus bits 2 through 14. Data patterns test data bus bits 0 through 21.

X

Entering **X** invokes the *X-Off File Test.* The X-Off File on the ALM-2/MCP board handles certain protocols used by different terminals. It is sixteen bytes of random access memory, readable and writable from the workstation.

F

Entering **F** from the Basic Test Menu invokes the *FIFO Test.* The 4 Kbyte First-In-First-Out memory (FIFO) holds asynchronous received data from the SCCs until the workstation has time to read it. With every 16-bit word of data, it holds up to six bits of ID information so the workstation knows which port it came from.

Hardware implements the FIFO as part of Common RAM, but for software it is virtually separate.

First, this test resets the ALM-2/MCP and expects a read of the FIFO to return the hexadecimal value ffff. That value comes from the FIFO controller hardware whenever it believes the FIFO is empty.

Next, the test does pattern writes then immediate reads to the empty FIFO. This tests the FIFOs data path as well as the FIFO controller for the ability to hold the same address in Common RAM for quick writes and reads. The test then fills the FIFO with patterns and reads them back.

DM

Entering **DM** from the Basic Test Menu invokes the *DMA Chip Addressing, Data Test.* The ALM-2/MCP uses DMA to address locations in Common RAM for the SCCs. The board has five 8237 DMA Controller chips to do this.

The 8237 chip has a program condition during which a program can write and read certain internal registers. This test uses that condition to write and read patterns to an internal register of each DMA Controller chip. The patterns are different for each chip, and the test covers addressing, partial testing for functionality, and the 8-bit data path to and from each chip.

S

Entering **S** from the Basic Test Menu invokes the *SCC Chip Addressing, Data Test.* The eight Serial Communication Controller (SCC) chips provide some of the serial protocols and the data encoding/decoding needed to read and write to the serial ports. The ALM-2/MCP provides a 3-bit wide bus, called the SCCID bus, for referencing the eight SCC chips. Another line selects between the A and B channel of each one (A/B), and still another line selects between SCC Data mode and Control mode (D/C). Depending on the transaction, the SCCID bus has one of three sources and one of four destinations.

This test uses some internal SCC registers (Register 12A and 12B) to test partial functionality of those chips and the functionality of the chip that decodes address bus bits 0 through 5 into SCCID, A/B and D/C.

SL

Entering **SL** from the Basic Test Menu invokes the *Scope Loops* option. 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.

MT

Entering **MT** from the Basic Test Menu selects the *More Tests* option. Selecting this option from the Basic Menu displays the Middle menu.

DLF Option Menu

If you select **DLF** from the basic test menu, the diagnostic displays the following menu of loopback descriptions.

```
Please choose a loopback description for the board.

CP       MCP Defaults, same as MPF (MCP + four single loopbacks).
M        ALM-2 Manufacturing Defaults (MCP + ALM-2, printer plug).
Pairs    8 ALM-2 pair cables.
All      16 ALM-2 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 run 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.
```

Selecting *Pairs* configures the board with asynchronous-only loopbacks. For this reason they are labeled as ALM configurations, rather than MCP.

A

Selecting *All* configures the board with asynchronous-only loopbacks. For this reason they are labeled as ALM configurations, rather than MCP.

MDf, MDT, MPf and **MPT**

The four MCP choices refer to the four synchronous/asynchronous ports on the MCP board. The diagnostic will know that only ports 0-3 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 following text describes the tests offered by the Middle Test Menu.

A

Option **A** of the Middle Test Menu, invokes the *SCC Asynchronous Data Flow Test.* The SCC provides an internal loopback capability that facilitates testing SCCs, their supporting hardware and the data path between them and Common RAM without involving the port line drivers and receivers. It also covers the data path from the SCCs to the FIFO.

This test fills Common RAM with a pattern and causes data to flow from it through the SCCs, using their internal loopback. It then reads the data in the FIFO and compares with the expected data. The test repeats this sequence for each SCC. At this point, the test does not use synchronization. It fills Common RAM with the same pattern in every location and then uses DMA transmit circuitry to cause the SCC to transfer data.

S

Entering **S** from the Middle Test Menu, invokes the *Sync Data Flow Test.* This test uses the internal loopback capability of the SCC chips to test the four synchronous ports for data transfer between the transmit area of Common RAM to the synchronous receive area of Common RAM. It therefore tests the same hardware as the Asynchronous Data Transfer test, but in synchronous mode. It also partially tests the DMA controller chip that is responsible for moving data into common RAM from the SCCs and the CIO chip's ability to catch the EOP signal.

NOTE *This test is for MCP boards only (with four Sync Serial ports).*

It fills half of Common RAM with a pattern and causes data to flow from it through the SCCs, using their internal loopback. It then reads the data in the other half of Common RAM and compares with the expected data.

D

Entering D from the Middle Test Menu, invokes the *DMAs Addressing Test.* The first part of this test behaves like the SCC Data test, except that it uses address-unique data in Common RAM. This process uncovers any addressing problems with the 8237 DMA Controller chips.

One of the DMA controllers is to be used for synchronous receiving. For this reason, ports 0 through 2 are first tested asynchronously, to verify the transmitter DMA controller, then synchronously to test the receiving DMA controller.

One DMA controller is used for transmitting data out of the parallel printer port. The diagnostic tests that controller during the parallel port test (where possible).

L

Entering **L** from the Middle Test Menu invokes the *Line Drivers/Receivers test.* This procedure allows a pair of ports to pass data between them, or each port to send data to itself. This test requires the loopback cables or connectors to the serial ports. Contact Sun Customer Support for information on obtaining a loopback connector kit.

This test is very similar to the DMA Addressing test, except that it uses an external loopback. If the previous tests passed, a failure here must be in the drivers or receivers of the RS-232C or RS-449 ports.

DT

Entering **DT** invokes the *DTR and DCD Control Test.* Data Terminal Ready (DTR) is a signal that the ALM-2/MCP sends out the RS232 port to the remote device. If the port is RS449, call that signal Terminal Ready (TR). The device is supposed to wait for that signal before attempting to send data.

Data Carrier Detect (DCD) is a signal that the ALM-2/MCP receives from the RS232 port. If the port is RS449, call that signal Receiver Ready (RR). It means that the remote device is there and working correctly.

This test uses the external loopback fixtures, which connect DCD to DTR on RS232, and connect TR to RR on Rs449. It turns DTR (TR) on and expects DCD (RR) to be on, then it turns DTR (TR) off and expects DCD (RR) to be off. It uses the DEVCTL Register to turn that signal on and it uses the SCCs to detect DCD or RR.

R

Entering **R** invokes the *RTS and CTS Control.Test.* Request To Send (RTS) is a signal that the MCP sends out a synchronous RS232 port to the remote device. On RS449 ports, call the signal RS. The device is supposed to respond, when ready, with Clear To Send (CTS on RS232, CS on RS449).

NOTE *This test is for MCP boards only (with four sync.serial ports).*

The Manufacturing test fixture or a loopback plug or cable connects these two signals together. This test turns on RTS (RS) and expects CTS (CS) to be on. Then it turns off RTS (RS) and expects CTS (CS) to be off. It uses the SCC's to control RTS/RS and to detect CTS/CS.

RP

If the printer loopback plug is not available, the only way to test the printer port is with a printer. The **RP** option from the Middle Test Menu invokes the *Real Printer Test.* The test attempts to print some lines of data to the printer while echoing the lines on the console. It informs the operator that the printer may not be receiving the ACK handshaking signal if the line did not show up on the If it cannot print the line because of one of the status signals, that information is also displayed on the console.

NOTE *This test is not available for the synchronous-only MCP, board because it does not have a parallel printer port.*

PL

Entering **PL** from the Middle Test Menu invokes the *Printer Port Loopback Test.* The parallel printer port is an output-only port. The only inputs to it are one handshake signal and two status signals. In order to test that port without using a real printer, you must use a loopback connector that connects four of the data outputs to one of the status signal inputs, and the other four to the other status input. The pin assignments for that connector are described at the beginning of this chapter. Contact Sun Customer Support for a loopback connector kit.

Expecting this connector to be installed on the printer port, the test sends patterns to the port and reads the status signals in the CIO. All four of the data lines must be "1" in order to assert the status signal. The test uses this fact to check for active and unshorted data lines and working status inputs.

MT

Entering **MT** from the Middle Test Menu selects the *More Tests* option. This option displays the Advanced Test Menu, described next.

21.6. 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.)
```

Following are descriptions of the Advanced Menu tests.

B

This option invokes the *Baud Rate Accuracy Test.* Although a previous test selected various baud rates, the rates of both the sending ports and receiving ports depended on the same clock crystal on the ALM-2/MCP. The tests would not detect an incorrect clock crystal. This test uses the CPU clock to measure the actual character flow through the ports to verify that the baud rate is true. A fast or slow character rate would indicate a wrong clock crystal.

X

This option invokes the *X-Off Function Test* This test puts all possible patterns into the X-Off File, then uses all possible patterns in the data stream. It expects the X-Off Controller to turn off CTS to the appropriate SCC channel only when it detects the correct X-Off character, and only when X-Off is enabled.

S

Entering **S** invokes the *SCC Interrupt Vectors Test.* Software can set the Disable Interrupts bit in the CIO and thereby disable ALM-2/MCP interrupts to the workstation. This test disables ALM-2/MCP interrupts, but enables them in each SCC chip in turn, after loading a different interrupt vector in each one. It then uses loopback to cause an SCC interrupt condition in every SCC chip, expecting only the enabled interrupt's vector to make it to the readable Device Interrupt Vector (DEVVECTOR). Even with ALM-2/MCP interrupts disabled, DEVVECTOR should be valid. This process tests each the internal device interrupt daisy chain for each SCC and ALM-2/MCP.

F

This option invokes the *CIO FIFO Interrupt Vectors Test* Certain FIFO signals go to the CIO chip so that it may generate interrupts to the workstation, if software has enabled them. These signals are: FIFO Half Full, FIFO Full and FIFO Empty.

This test uses the SCC internal loopback to move data into the empty FIFO until it is half full, with ALM-2/MCP interrupts to the workstation disabled (in the CIO), and FIFO Half Full interrupts (in the CIO) enabled. An interruption would, therefore, indicate that the CIO chip failed to disable interrupts to the workstation. If the workstation does not get an interrupt, the diagnostic examines DEVVECTOR for the expected vector from the CIO.

Next, the test empties the FIFO with FIFO Empty interrupt enabled in the workstationCIO,with If the workstation does not get an interrupt, the diagnostic examines DEVVECTOR for the expected vector from the CIO. sp The test repeats this for FIFO Full.

E

This option invokes the *CIO EOP Interrupt Vectors Test* The CIO allows each of the six DMA controller EOP signals to generate a VME interrupt, if software enabled them.

This test clears any pending EOP interrupt, disables workstation interrupts in the CIO, then enables CIO interrupts for EOP0, EOP1, EOP2, EOP3, EOP4 and EOP5. It causes the DMA chips to generate each EOP signal (where possible, in the case of EOP5, from the parallel port) and expects the CIO to show those pending interrupt signals, and DEVVECTOR to show the correct pattern.

D

Entering **D** invokes the *CIO DSR/DM Interrupt Vectors Test.* Synchronous communication uses the handshake input signal called Data Set Ready (DSR) on RS232 ports, and Data Mode (DM) on RS449 ports. The Manufacturing test fixture will loop back the DTR (TR
on RS449) output signal to DSR (DM on RS449) on ports 0 through 2, as well as to the input signal DCD (RR on RS449) For each port, this test disables ALM-2/MCP interrupts and turns on DTR/TR in the DEVCTL register to cause a DSR/DM event. It then checks the CIO for showing that signal and for the correct value in DEVVECTOR. An interrupt would indicate a CIO failure. No signal would be a failure either of the CIO or the DSR/DM line receiver.

V

Option **V** invokes the *VME Interrupts from SCC, CIO* option. By now, the diagnostic has confidence that the SCCs and the CIO can process interrupt conditions and assert their interrupt vectors. This test enables workstation interrupts in the CIO and repeats some of the previously described interrupt signal tests to make sure that each device or event can cause the ALM-2/MCP to generate interrupts to the workstation.

M

Entering **M** selects the *Multiport Test*. This test attempts to stress the ALM-2/MCP with multi-tasking. It sets up a transmit data area in Common RAM for each of the sixteen ports, with different data in each area. It also sets up an area in workstation memory for taking received data from the FIFO.

The test sets up an interrupt handler in the workstation to handle MCP interrupts, and sets the MCP to generate interrupts on FIFO -not-empty. That handler reads the FIFO data for the number of the port that sent the character to the FIFO, then puts the character into that port's save area. If the port number is invalid (negative or greater than 15), the handler generates an error message. The handler also keeps count of the number of characters it moved into each port's data area and generates an error message if there were too many.

The handler will not return until the CIO says the FIFO is empty.

The test then initializes each port for internal, asynchronous data transfer, from Common RAM to the FIFO. It starts each port for data transfer, then begins polling the CIO for EOP's from the four transmit DMA controllers. The FIFO -not-empty interrupt handling should be going on now. When the poller sees an EOP, it reads the status register of the corresponding DMA chip, accumulating the results in a data array. That status register shows Terminal Count (TC) information for each channel of the chip. The test ANDS all four of the DMA controller chips' status registers.

When the result of the ANDing is all ones, it means all sixteen of the ports have sent all of their data. The test then leaves the polling loop and checks the resulting data in workstation memory for correctness. It generates a message if the data is not correct.

21.7. 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.

21.8. 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 didn't compare. 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 compare error. 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 compare error. 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 compare error. 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 checked each SCC chip's ability to hold all possible patterns. Suspect the chip or the data lines. Each chip is responsible for two ports. Channel A on the chip is the odd-numbered port and Channel B is the even-numbered port. Therefore, SCC 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.**

NEED DESCRIPTION

**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 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 mismatch.

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 the expected data. 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.

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 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 mismatch.

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 mismatch.

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.

First, suspect the path between the DCD input line and the receiving port's SCC input. Also, 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.

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 produced 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 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. Of 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 loopback 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 at the beginning of this chapter. 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 "no vector" sign.

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 necessary number of bytes into the FIFO 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. You may have installed the wrong loopback fixture, and as a result, the test expected the DSR from the wrong port. For example, the port really has a single loopback connector, but you entered the main menu DLF option "loopback pair" setting.

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 SunDiagnostic Executive feature 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 or reboot the Exec.

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 SunDiagnostic Executive feature 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 or reboot the Exec.

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 many 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, there is either 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 SunDiagnostic Executive feature 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 or reboot the Exec.

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. The test checking for the Terminal Count indications from every channel of the transmit DMA controllers to determine that the movement was ended, then 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 SunDiagnostic Executive, but that function failed. Try the test again or reboot the Exec.

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" 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 under 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 under 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.

21.9. 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 for troubleshooting 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 that 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.

21.10. Glossary

ALM-2 Asynchronous Line Multiplexer-2 board — produced in two different products. One of them is the "ALM-2," a 16-port asynchronous device, which also has one parallel printer port. The other one is the "MCP."

MCP The Multiprotocol Communications Processor (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 standard 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 implies the connecting of 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.

22

MTI/ALM Board Diagnostic

22

MTI/ALM Board Diagnostic

22.1. 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.

22.2. 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.

22.3. 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.

22.4. 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 sequence 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.

22.5. 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.

22.6. 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, as opposed to a parallel port, 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.

22.7. 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

23

Sun Memory Diagnostic

23

Sun Memory Diagnostic

23.1. General Description

The Sun Memory Diagnostic is a menu-driven diagnostic designed to detect main memory errors in Sun-3, Sun-4 and SPARCsystem CPU boards. It is a group of tests that can be executed 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.

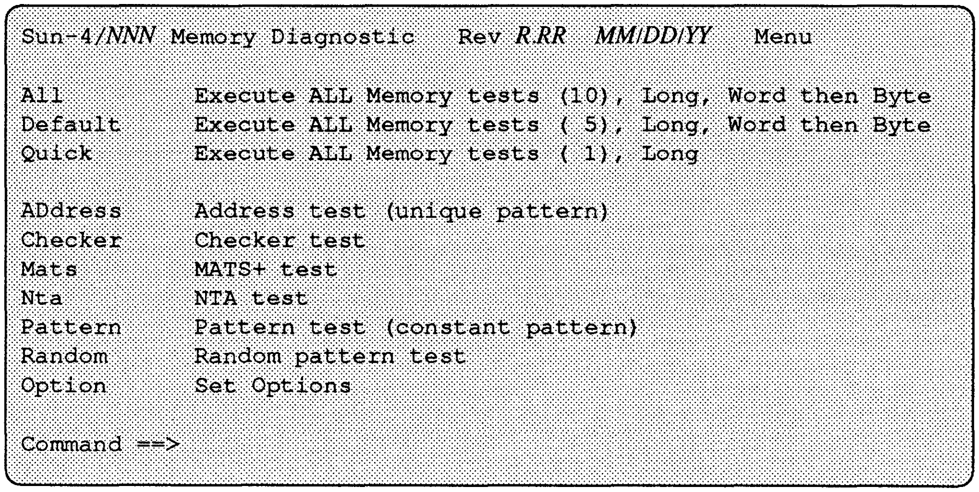
NOTE *This diagnostic does not test cache on any system CPU board. It is a main memory diagnostic; if the CPU board has cache memory, it will be disabled during testing.*

23.2. Hardware Requirements

- □ 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.

23.3. The Main Menu

The Main Menu is shown below:



```
Sun-4/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`.

`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`.

`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`.

`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 that 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 the 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.

```
B=yte 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
```

23.4. 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.

Stuck-at Fault
An error in which a data bit is stuck at either the logic 1 or logic 0 state.

24

Parity Diagnostic

24

Parity Diagnostic

24.1. General Description

The Parity Diagnostic is designed to verify the error-checking mechanism of the Sun-3/80. The diagnostic verifies the functionality of the parity circuitry and of parity memory.

24.2. What This Chapter Contains

Following an overview of the diagnostic and a list of required hardware, the Main Menu and options are discussed. Optional parameters and their defaults are shown. The end of the chapter contains a list of error messages and a glossary.

24.3. Overview of the Diagnostic

The parity error-checking mechanism detects data faults in main memory. It ensures that data integrity is maintained and that any corrupted data is detected before it is used. Although parity cannot detect every error that may occur, it is an efficient mechanism for detecting the most common kinds of memory failures.

The Parity Diagnostic is designed to verify that the parity circuitry is functioning properly. It does so by "forcing" errors through main memory on its initial pass. On the second pass, it scans all of memory again, to verify that parity errors were generated in those locations where it seeded the bad data. If the diagnostic does not receive a parity error in each location, it generates a message indicating possible bad parity circuitry.

The Parity Diagnostic was created specifically for the Sun-3/80 because the parity circuitry in the Sun-3/80 differs slightly from the implementation of parity on other Sun-3 systems. Although the difference is minor, it should be noted by manufacturing technicians and repair personnel, so that debugging time of parity circuitry can be minimized.

When the parity circuitry of a Sun-3/80 detects a parity error, it causes a bus error to be generated. Other Sun-3 systems generate a Level 7 Interrupt when detecting the same error. Because the Parity Diagnostic forces parity errors into main memory, it is programmed to receive bus errors on its second pass, not Level 7 Interrupts.

Bus errors are used instead of Level 7 Interrupts because they provide the software with more information about the error that occurred and the state of the system when it occurred. SunOS software can use this extra information to analyze the error and take appropriate action.

24.4. Hardware Requirements

The following hardware is required to run the Parity Diagnostic:

□ A Sun-3/80 with verified functional memory

□ A monitor

□ A keyboard

□ A boot device (local disk, local tape, or remote disk over Ethernet)

24.5. Firmware Requirements

The standard power-up boot PROM is also required to run the Sun-3/80 Parity Diagnostic. The power-up diagnostics perform the main memory verification that the Parity Diagnostic requires before it can run properly. The boot PROM is also used to load and begin execution of the Parity Diagnostic.

24.6. User Interface

The user interface of the Parity Diagnostic adheres to the standards of the Exec menu system. Each test may be selected from a menu by typing the letter or letters displayed in uppercase in the column on the left side of the menu.

Additional parameters and options may be specified on the command line. If a parameter is not specified on the command line, the default value is used.

To exit the diagnostic and return to the Exec, press [ESC] then press [RETURN] when the Main Menu is displayed.

To display online Help for any menu option, enter the following on the command line:

? *option_name*

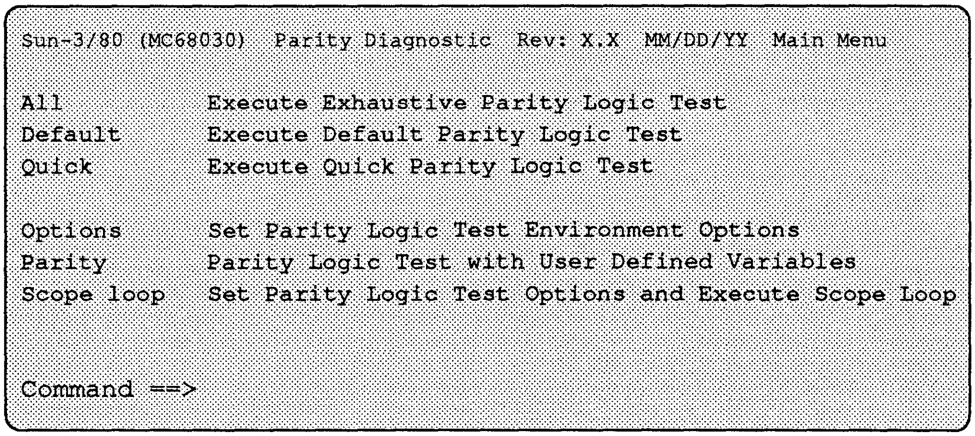
24.7. Starting the Diagnostic

For information on starting the Exec, refer to Chapter 2, "Using the SunDiagnostic Executive." After you have started the Exec, choose the Parity Diagnostic from the Diagnostics Menu.

24.8. The Main Menu

This section of the chapter provides a modular description of the Parity Diagnostic. It discusses each of the options available on the Main Menu and lists the parameters that may be used, together with their default values. A list of messages generated by the diagnostic can be found in the section entitled "Error Messages."

The Main Menu, which displays when you start the diagnostic from the Exec, provides access to the individual tests:



```
Sun-3/80 (MC68030)  Parity Diagnostic  Rev: X.X  MM/DD/YY  Main Menu

All          Execute Exhaustive Parity Logic Test
Default      Execute Default Parity Logic Test
Quick        Execute Quick Parity Logic Test

Options      Set Parity Logic Test Environment Options
Parity       Parity Logic Test with User Defined Variables
Scope loop   Set Parity Logic Test Options and Execute Scope Loop


Command ==>
```

Optional Parameters

The All, Default, and Quick options on the Main Menu take some common parameters. The meanings of the parameters are shown below.

Parameter	*Description*
`PASs`	Pass count
`Option`	Byte, word, or long word modes
`Pattern`	Data pattern

All Test

A

The *Execute Exhaustive Parity Logic Test* option on the Main Menu executes ten passes of the Parity Logic Test in long word, word, and byte modes.

Default Test

D

The *Execute Default Parity Logic Test* option on the Main Menu executes five passes of the Parity Logic Test in long word, word, and byte modes.

Quick Test

Q

The *Execute Quick Parity Logic Test* option on the Main Menu executes the Parity Logic Test once in long word mode.

Options

O

The *Set Parity Logic Test Environment Options* command on the Main Menu sets the testing environment. It controls the mode in which the Parity Logic and Scope Loop tests run and sets the size of access for these tests. It also allows you to specify if the address of memory under test is incremented or decremented during testing.

The parameters that may be specified on the command line are shown below. Parameters listed on the same line are mutually exclusive. For example, each test may be run in byte, word, or long word mode. Enabling one of these modes disables the other two.

```
[Byte] [Word] [Long]
[INCrement] [NOINCrement]
[DECrement] [NODECrement]
```

The Options command syntax is as follows:

Options *[Byte][Word][Long]*
[INCrement][DECrement]
[NOINCrement][NODECrement]

The following example sets byte mode and increments the address under test:

```
o b inc
```

The default is as follows:

```
o l
```

To display the current options, choose the Options command with no parameters specified.

Parity

P *ad= si= st= pas= p=*

The *Parity Logic Test with User Defined Variables* option on the Main Menu initiates the Parity Logic Test. This test writes data with incorrect parity into different areas of memory to verify that the parity circuitry is able to detect errors in all parts of the memory range.

The Parity Logic Test first divides the memory range under test into blocks of equal size; each block is specified by the `st=` option. The test then writes the specified pattern with incorrect parity to the first location of each block. After writing the pattern across the entire range, it reads the first location of each block. In each block, a bus error should have been generated as a result of the parity error. If the Parity Logic Test does not detect a bus error, an error message is displayed.

The meanings and default values of the parameters are shown in the following table:

Parameter	*Description*	*Default*
`ADdress`	Base address for this test	0x00000000
`SIze`	Size of memory to be tested	Environment size
`STep`	Block specification	0x2000
`PASs`	Pass count for this test	1
`Pattern`	Data pattern for this test	0x80808080

Consider this example, in long word mode:

```
p ad=100000 st=1000 pas=3 p=80808080
```

This command writes `0x80808080` with inverted or "bad" parity into location 0x100000, 0x101000, 0x102000, and so on. Writing continues until the address reaches the current environment size (the default value of `si=`). The test then returns to address 0x100000 and reads the location that it wrote to previously — this time, however, with correct ("good") parity. Each read should generate a bus error. Any read that does not generate a bus error causes an error message to be displayed. The test is executed three times.

Scope Loop

s *ad= p=*

The *Set Parity Logic Test Options and Execute Scope Loop* option on the Main Menu initiates scope loop testing. A specific memory location is tested continuously by setting up a scope loop on that location.

The base address and pattern to be used may be specified on the command line. The size of the access (long word, word, or byte) must be specified with the Options command. After the scope loop is started, it continues until a key is pressed.

The meanings and default values of the parameters are shown in the following table:

Parameter	*Description*	*Default*
`ADdress`	Base address for this test	0
`Pattern`	Data pattern for this test	0x80808080

Consider this example:

```
s l o; ad=1C00 p=0x5555AAAA
```

The Options command is entered first, to set the access size to long word. The location 0x1C00 is then written repeatedly, using the pattern 0x5555AAAA and inverted parity. Next, the location is read with correct parity. Each read access to this location should generate a bus error. If a bus error is not detected, an error message is displayed.

24.9. Error Messages

The following error messages may be displayed by the Parity Diagnostic:

```
Enter Options Again.

Unable to Map Memory.

Memory Size specified > Available memory.

Unable to UNMap Memory.

Parity Bus Error Expected but NOT Received.

Bus Error Received, but cannot detect Parity Error.

Memory Size must be specified.

Operator Error.

Unable to Map Register at Phys: 0x<x>

Unable to UnMap Register at Phys: 0x<x>

Memory Size specified is greater than available memory.

Unable to Install Bus Error Trap Handler.

Unable to Remove Bus Error Trap Handler.

Parity Bus Error Received but NOT Expected.

Parity Error Bits in Mem_Err_Ctl_Reg do NOT match bytes with Bad parity.

Extra Parity Bus Error Received but NOT Expected.
```

24.10. Glossary

DRAM	Dynamic Random Access Memory.
Parity	A hardware-based memory error detection mechanism.
Parity Memory	An area of memory, set aside from main memory, used for error detection information.

25

Sun Mouse Diagnostic

25

Sun Mouse Diagnostic

25.1. 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-4 system.

25.2. 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).

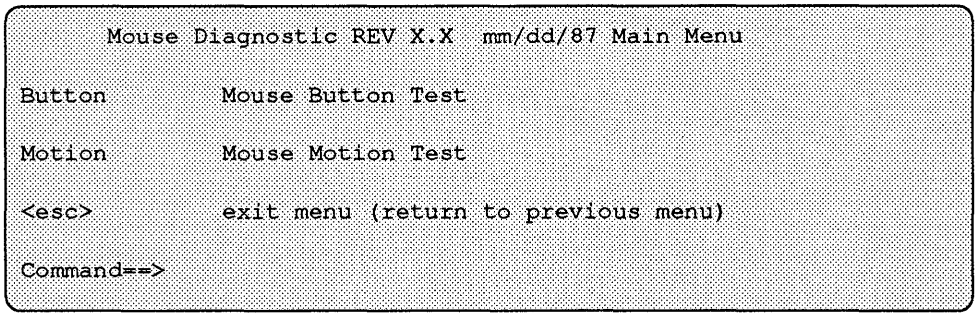
25.3. Mouse Data

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.

25.4. 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 you to the previous menu. Pressing ? displays the Mouse Diagnostic Help Menu.

Figure 25-1 *Mouse Diagnostic Main Menu*



```
      Mouse Diagnostic REV X.X  mm/dd/87 Main Menu

Button         Mouse Button Test

Motion         Mouse Motion Test

<esc>          exit menu (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 mouse 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.

25.5. 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

26

Sun Ethernet Diagnostic

26

Sun Ethernet Diagnostic

26.1. General Description

The Ethernet Interface provided on Sun-3 and Sun-4 CPU Boards connects the 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 Ethernet Diagnostic tests the Ethernet Interface on Sun-3/1xx, Sun-3/2xx, Sun-3/80, Sun-3/4xx and Sun-4 CPU boards.

26.2. Hardware Requirements

The minimum hardware configuration required is:

1. A Sun cardcage.
2. A Sun power supply.
3. A Sun CPU board.
4. A Sun Memory board if there is no memory on the CPU 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.

26.3. The Main Menu

The user interface of the Ethernet diagnostic consists of a Main Menu and five 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.

Local copies of the global environmental parameters may be set and used by those tests which have a double asterisk, **, in their parameter list. See the section on Local Environmental Parameters for details.

26.4. Local Environmental Parameters

The local environmental parameters are a subset of the global environmental parameters that are available on the Options Menu. Any of these local environmental 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 that it follows. If a local environmental parameter is not entered, the current value of the equivalent global environmental parameter, as displayed on the Options Menu, will be the default value. Here is an example entry:

```
testcommand Pass= SCope= SOft= STop= WAit=
```

Possible parameters are:

Pass
: Set pass count limit. Number of times a test is to be executed.

SCope
: Set flag. 0 = off, 1 = on.

SOft
: Set soft error retry limit.

STop

Set stop on nth error limit.

WAit

Set wait flag. 0 = off, 1 = on.

INFO

Set info message level. 0 = off, 1 = terse, 2 = verbose, 3 = more verbose.

STATUS

Set status message level. 0 = off, 1 = terse, 2 = verbose, 3 = more verbose.

ICAche

Enable internal cache(s) 0 = off, 1 = instruction, 2 = data, 3 = both.

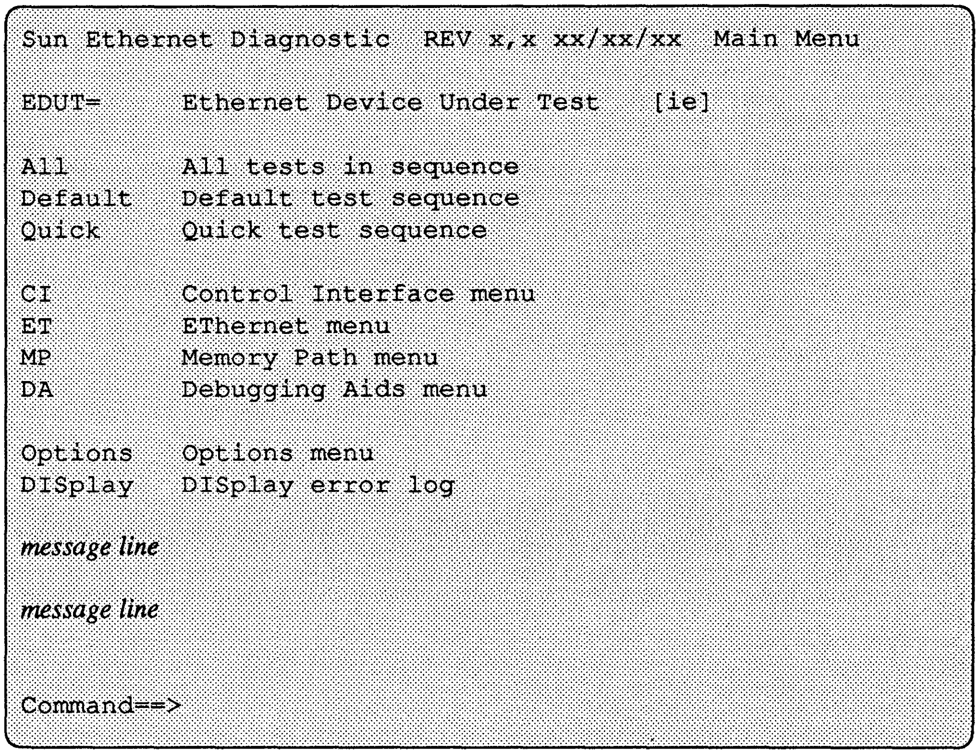
XCAche

Enable external cache(s) 0 = off, 1 = central, 2 = io, 3 = both.

The defaults and ranges are:

parameter	*default*	*range*	*base (radix)*
Pass	1	1__MAXINT	dec
SCope	0	0__1	dec
SOft	0	0__MAXINT	dec
STop	0	0__MAXINT	dec
WAit	0	0__1	dec
INFO	1	0__3	dec
STATUS	0	0__3	dec
ICAche	1	0__3	dec
XCAche	3	0__3	dec

26.5. Ethernet Diagnostic Main Menu



```
Sun Ethernet Diagnostic  REV x,x xx/xx/xx  Main Menu

EDUT=     Ethernet Device Under Test   [ie]

All       All tests in sequence
Default   Default test sequence
Quick     Quick test sequence

CI        Control Interface menu
ET        EThernet menu
MP        Memory Path menu
DA        Debugging Aids menu

Options   Options menu
DISplay   DISplay error log

message line

message line


Command==>
```

EDUT=

Enter the Ethernet chip set you wish to test after this command. For example:

```
edut=le
```

A

If the `all` option is selected, a sequence containing all of the tests will run.

D

If you select `default` a sub-set of non-interactive tests will execute.

DQ

If the `quick` test is selected, a sub-set of the default test sequence will execute.

CI

This option brings up the Control Interface submenu, described later.

ET

This option brings up the Ethernet Chip Set submenu, which is described later.

MP

This option brings up a submenu of Memory Path tests, described later.

DA

This option brings up a Debugging Aids menu, described later.

o

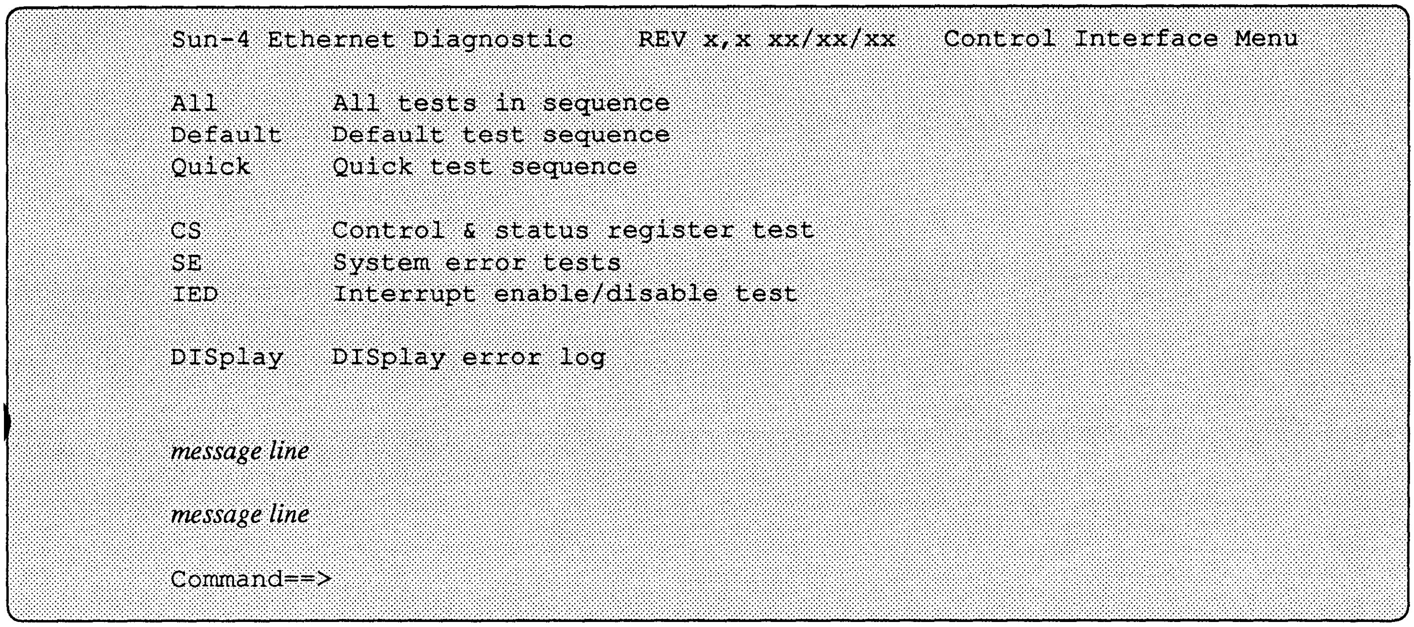
If you enter **o**, the options menu, described later, appears.

DIS

This option displays the error log. If the *control interface* menu is selected, the diagnostic displays a sub-menu containing the control interface tests.

26.6. The Control Interface Menu

The Control Interface tests check 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-4 Ethernet Diagnostic    REV x,x xx/xx/xx   Control Interface Menu

All       All tests in sequence
Default   Default test sequence
Quick     Quick test sequence

CS        Control & status register test
SE        System error tests
IED       Interrupt enable/disable test

DISplay   DISplay error log


message line

message line

Command==>
```

A

The *All tests* command runs all of the tests available on this menu. It executes them in the sequence shown on the menu.

Default

The *Default tests* command runs a subset of the tests available on this menu.

QUick

The *Quick tests* command runs a small set of tests that complete in a short time.

CS

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 any of the parameters shown under "Local Environmental Parameters".

SE

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 any of the parameters shown under "Local Environmental Parameters".

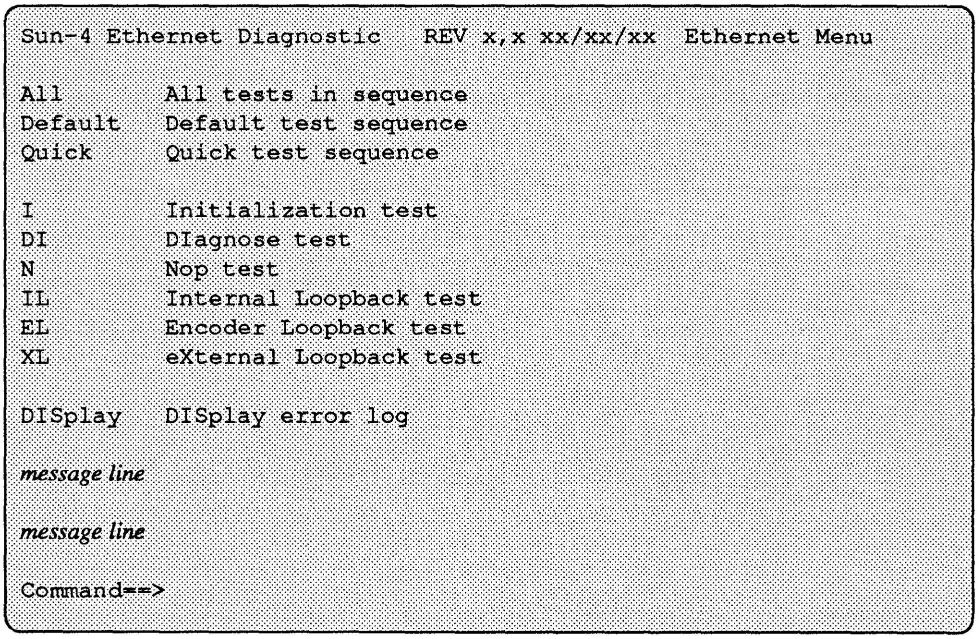
IED

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.

Use any of the parameters shown under "Local Environmental Parameters".

26.7. The Ethernet Menu

The Ethernet tests check the functionality of the Ethernet chip set.



```
Sun-4 Ethernet Diagnostic   REV x,x xx/xx/xx  Ethernet Menu

All       All tests in sequence
Default   Default test sequence
Quick     Quick test sequence

I         Initialization test
DI        DIagnose test
N         Nop test
IL        Internal Loopback test
EL        Encoder Loopback test
XL        eXternal Loopback test

DISplay   DISplay error log

message line

message line

Command==>
```

A

The *All tests* command runs all of the tests available on this menu. It executes the following commands in sequence:

```
I ; DI ; N ; IL ; EL ; XL
```

D

The *Default tests* command runs a subset of the tests available on this menu. It executes the following commands in sequence:

```
I ; DI ; N ; IL
```

Q

The *Quick tests* commands runs a small set of tests that complete in a short time. It executes the following commands in sequence: `I ; DI`

I

When you select the Initialization test from the Ethernet Menu, the Ethernet chip set control blocks are initialized; the chips are reset; and a channel attention is issued for the Intel chip. Correct operation is then verified.

DI

The Diagnose test runs a self-diagnostic operation for the Intel Ethernet chip only.

N

The NOP test runs a "No Operation" command for the Intel Ethernet chip, and reports the results.

IL *fifo= blk=*

The Internal Loopback test verifies performance of the Ethernet chip set buffer management, address recognition, CRC generation and detection, interrupt and retransmit features. To do this, the test configures the chip set in internal loopback mode, transmits and receives blocks, analyzes control blocks for the correct state, and compares transmitted and received data.

fifo=

This parameter specifies the size of the internal FIFO for the Intel chip only.

blk=

This parameter specifies the size of the transmit and receive buffers.

**=

Any parameters shown under "Local Environmental Parameters" may also be used with the Internal Loopback command.

Defaults and the acceptable range of entries for the FIFO and blocksize parameters are as follows:

parameter	*default*	*range*	*base (radix)*
fifo	8	1__0xf	hex
blksize	0x12	1__0xa000	hex

EL *fifo= blk=* — Intel only

The Encoder Loopback command verifies correct operation of the Intel Ethernet chip set through the Serial Interface Adapter, without going to the transceiver.

You may specify the internal FIFO size and the transmit and receive buffer sizes as described for the Internal Loopback command, in addition to the parameters shown under "Local Environmental Parameters".

XL *fifo= blk=*

The External Loopback test verifies proper operation of the Serial Interface Adapter and the transceiver interface. To do this, the test configures the chip set in full external loopback mode and performs the sequence of tests described for the internal loopback.

You may use the parameters described for the Internal Loopback command.

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 chapter
- a transceiver cable
- a transceiver with two terminating connectors

If your system boots with the

```
le
```

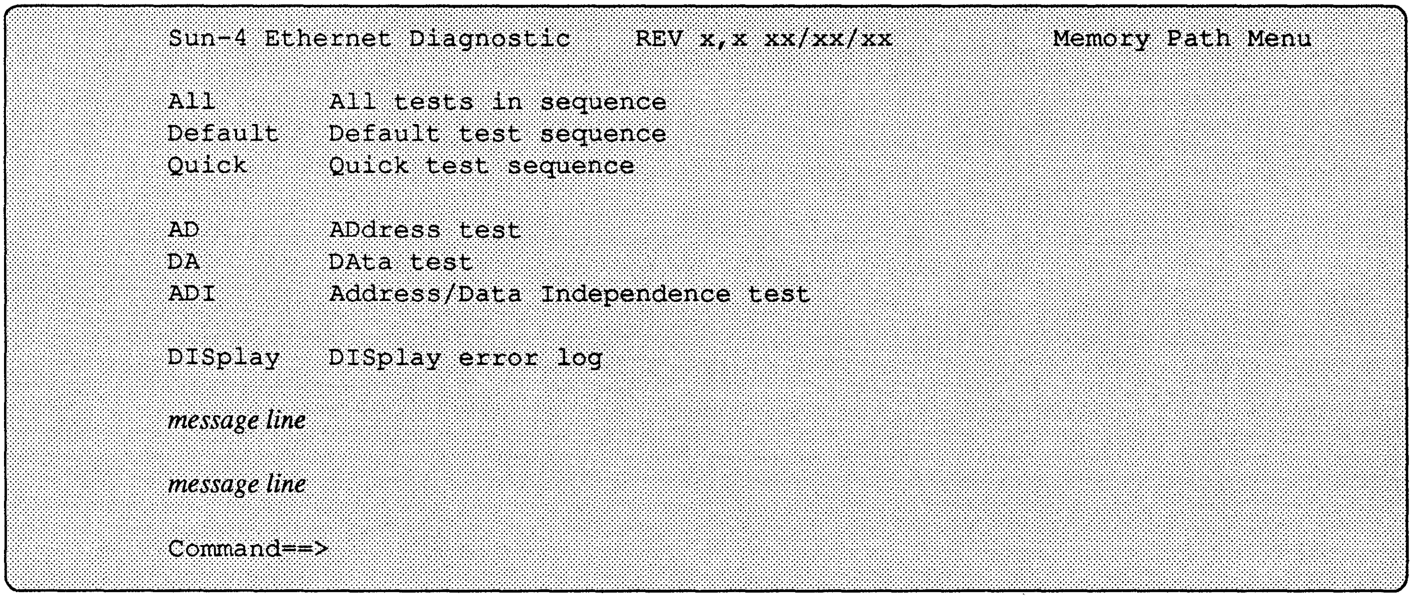
boot device command, the message

```
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.*

26.8. 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-4 Ethernet Diagnostic    REV x,x xx/xx/xx          Memory Path Menu

All        All tests in sequence
Default    Default test sequence
Quick      Quick test sequence

AD         ADdress test
DA         DAta test
ADI        Address/Data Independence test

DISplay    DISplay error log

message line

message line

Command==>
```

A

The *All tests* command runs all of the tests available on this menu.

D

The *Default tests* command runs a subset of the tests available on this menu.

Q

The *Quick tests* commands runs a small set of tests that complete in a short time.

AD *fifo= blk=*

The purpose of the *Address* test is to check the Ethernet chip set's ability to access memory within its address range. 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. This test accepts these parameters in addition to those shown under "Local Environmental Parameters".

fifo=

specifies the size of the internal FIFO for the Intel chip set.

blk=

specifies the size of the transmit and receive buffers. The range and defaults for these two parameters are:

parameter	*default*	*range*	*base (radix)*
FIFOlim	8	1__0xf	hex
BLKsize	0x12	1__0x12	hex

DA *fifo= blk=*

The purpose of the *Data Lines* test is to check the Ethernet chip's ability to write and read data to/from memory.

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.

The parameters are the same as those shown for the Address test.

ADI *fifo= blk=*

The *Address/Data*Independence test verifies the Ethernet chip set's ability to write and read data in memory independent of the memory address accessed. (Note: both chip sets have multiplexed address and data lines) The test writes and reads memory using data patterns different from the addresses accessed, then verifies the data.

This test accepts the same parameters as the Address test.

26.9. 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.

```
Sun-4 Ethernet Diagnostic  REV x dd/mm/yy  Debugging Aids Menu

RS       ReSet toggle
CA       Channel Attention toggle
DCB      Dump control blocks
DDS      Display Data Structure addresses

DISplay  DISplay error log

message line

message line

Command==>
```

RRS

The *Reset Toggle* test toggles the Ethernet chip's set/reset line so it can be checked with an oscilloscope. This test accepts parameters shown under "Local Environmental Parameters".

CA *mode=*

The *Channel Attention (Intel only)* test toggles the Ethernet chip's channel attention line so it can be checked with an oscilloscope.

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.

DCB *lev= ind=*

The *Dump Control Blocks* test displays for analysis the contents of the control and status registers, the control blocks, and any buffers.

The defaults are `lev=-1 ind=-1`. Parameters shown under "Local Environmental Parameters" may also be used.

The test displays the specified control blocks on the screen. The flag bits described below allow you to control the contents of the display. Level specifies which control blocks are to be displayed. Each bit in level is assigned to a particular block using power of 2 encoding. Using -1 causes all blocks to be displayed.

Table 26-1 *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 26-2 *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)

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").

parameter	*default*	*range*	*base (radix)*
LEVel	ffff	0000__ffff	hex
INDex	0	0__16	dec

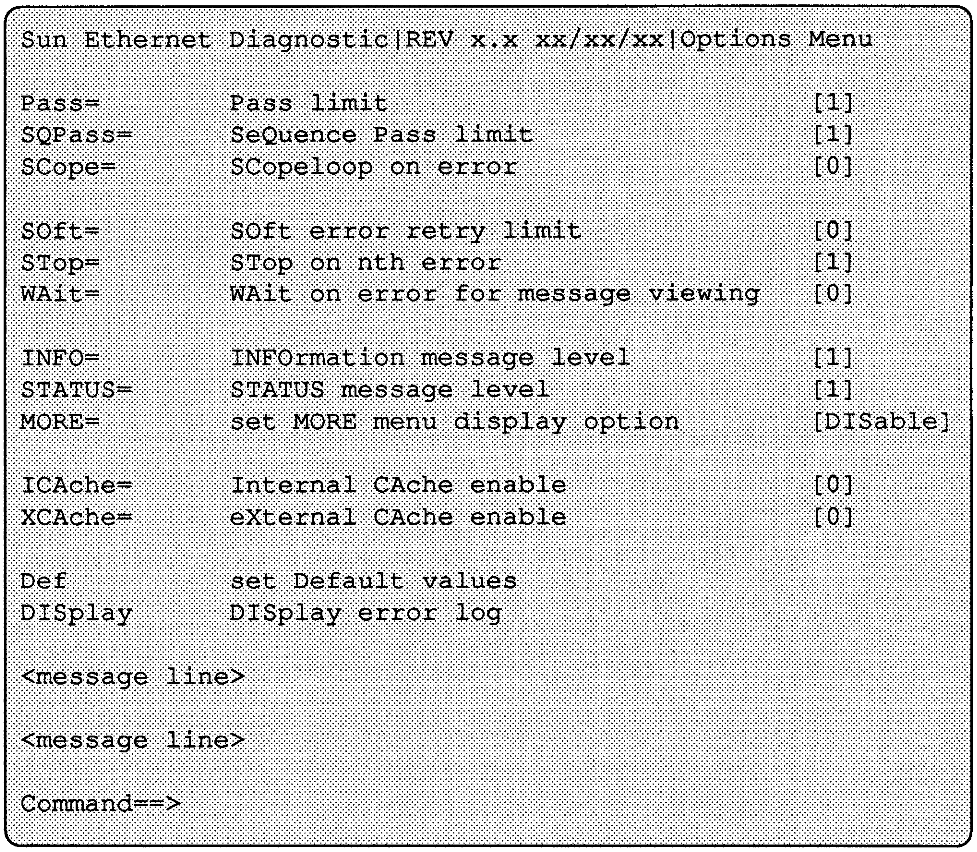
`DDS` *lev= ind=*

The *Display Data Structure Addresses* selection displays specified data structure addresses to assist debugging done with logic analyzers. This command indicates what locations the Ethernet chip set is attempting to access.

Parameter definitions and defaults are the same as the Dump command above.

The Options

The options are non-test, non-exerciser functions for control and display of program environment variables and flags and the error log.



```
Sun Ethernet Diagnostic|REV x.x xx/xx/xx|Options Menu

Pass=          Pass limit                           [1]
SQPass=        SeQuence Pass limit                  [1]
SCope=         SCopeloop on error                   [0]

SOft=          SOft error retry limit               [0]
STop=          STop on nth error                    [1]
WAit=          WAit on error for message viewing    [0]

INFO=          INFOrmation message level            [1]
STATUS=        STATUS message level                 [1]
MORE=          set MORE menu display option         [DISable]

ICAche=        Internal CAche enable                [0]
XCAche=        eXternal CAche enable                [0]

Def            set Default values
DISplay        DISplay error log

<message line>

<message line>

Command==>
```

Pass=

This limit allows a test to be run until the number of iterations exceeds the pass count limit unless an "on error limit" is exceeded first. The default number of passes is one. Enter the decimal value for the number of passes, like this:

command **p=10**

SCope=

This option sets the global scopeloop on error flag to the value entered after the equals sign. 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 will display 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. This is intended to be used as a SYNC trigger for scope/analyzer debugging. No messages are displayed while looping in order to keep the loop as tight as possible.

0 = off, 1 = on. The default setting is sc=0.

SOft

This option sets the soft error retry count limit to the value entered.

Applicable tests will retry if an error occurs until the number of retrys exceeds the soft error retry count limit. Enter

`so=`*some_value*

`STop=`

This option sets the stop on nth error limit to the value entered. This limit causes testing to be stopped when the number of errors exceeds the stop on nth error limit. A value of zero implies that the test will not stop, but continue, when an error occurs. A value of one will cause the test to stop on the first error. Enter

`st=`*some_value*

`Wait`

This option sets the wait on error flag, which causes testing to stop temporarily for 30 to 60 seconds when an error occurs so that messages on the screen may be read before testing continues or before the menu is repainted.

```
wa=0
```

means that the test will not wait on error.

```
wa=1
```

causes the test to stop as described above.

`INFO=`

This option control information messages. Settings are as follows:

0 = off, 1 = terse, 2 = verbose, 3 = more verbose

Default is “1”, or terse messages. To set the message level, enter a command such as **`info=2`**.

`STATUS=`

This option controls status messages. Settings are as follows:

0 = off, 1 = terse, 2 = verbose, 3 = more verbose

`MORE=`

This option controls whether or not the screen scrolls one page at a time, as it does when you use the SunOS `more` command.

To enable this option, type **`more=en`**. To disable the option, type **`more=dis`**.

`Internal CAche`

This option controls internal cache. To disable the internal cache, enter **`ica=0`**.

To enable internal instruction cache only, enter **`ica=1`**.

To enable internal data cache only, enter **`ica=2`**.

To enable both data and instruction cache, enter **`ica=3`**.

`eXternal CAche`

The **XCA** option controls external cache(s). Enter **`xca=0`** to disable the external cache.

To enable central processor cache only, enter **xca=1**.

To enable I/O cache only, enter **xca=2**.

To enable both central and I/O cache, enter **xca=3**.

Def

The **d** option sets default values for all variables and flags. You may enter just **d** on the command line, or **def**.

DISplay

The **dis** option displays the list of errors recorded in the errorlog. To display errors in the normal format, enter .**dis**fmt=1

To display errors in the indented calling sequence display mode, enter .**dis**fmt=2

To flush the error log, enter .**dis**fl=1

If you do not want the error log flushed, or cleaned out, enter .**dis**fl=0

This command also accepts the parameters shown at the beginning of this chapter under "Local Environmental Parameters".

27

Second Ethernet Board Diagnostic

27

Second Ethernet Board Diagnostic

27.1. Second Ethernet Board Overview

The second Ethernet Board provides an Ethernet interface by utilizing the Intel 82586 LAN (Local Area Network) controller, 82501 SIA (Serial Interface Adapter) and 82502 Transceiver. The board also provides 256K DRAM that the EDLC uses for buffering transmit/receive packets, and Page Map RAM s that consists of four 1Kx4 static RAM. The CPU communicates with the board over the system bus. The CPU also shares some of the 256K Ethernet memory with the EDLC. The CPU places commands to the EDLC in the shared memory area. The CPU attains the EDLC's attention by toggling Channel Attention (CA) pin. To get access to board's memory area, the CPU must put out a virtual address which when translated to a physical address that would have its lower 20 bits between 0x40000-0x80000. The EDLC may also interrupt the host CPU by putting out an interrupt signal at priority 3, vector number 0x75.

The Second Ethernet Board is accessed through virtual addressing and uses its own page map. The host CPU can access the board by putting out an address whose lower 20 bits have the pattern 0x40000-0x80000 for memory area or 0x88000-0x88800 for the register area. Here is a brief description of the board's register area:

offset 0x0-0x800	Page map 16-bit pointers.
offset 0x840	16-bit Status/Control Register.
offset 0x844	Parity Error Address Register.

Before starting any of the tests, ensure that the coaxial cable has carrier sense voltage levels that meet standard requirements. If you are doing a board-level test then it is recommended that the "All" option be selected with a Halt-on-Error condition.

27.2. Hardware Requirements

1. Multibus-based Sun-2 system or VMEbus-based Sun-3 (or later) system.
2. Multibus or VMEbus Ethernet board, depending on the host system.
3. Sun monitor or an RS-232 terminal.
4. Boot device that usually consists of remote disk connected to the test system over an Ethernet link.

27.3. Loading And Starting

Upon selecting the second Ethernet diagnostic from the Exec's Diagnostics menu, a menu shows several possible tests. Before starting any of the tests, ensure that the coaxial cable has carrier sense voltage levels that meet standard requirements. If you are doing a board-level test, it is recommended that the "All" option be selected with a Halt-on-Error condition. You may also select the number of passes that each sub-test should make.

27.4. The Main Menu

The Main Menu of the Second Ethernet board Diagnostic is shown below.

```
Second Ethernet Board Diagnostic  REV x.x  MM/DD/YY  Main Menu

Set                     Set the test parameters
All                     Run all the tests
Default                 Run only default tests
Quick                   Run only Quick tests
Register                Register Test
Memory                  Ethernet Memory Test
Pagemap                 Pagemap Test
Internal                Internal Loopback
Encoder                 Encoder Loopback
External                External Loopback
DIagnose                Execute Diagnose, TDR commands
INTR                    Test the interrupt circuitry

Command==>
```

The following text describes the Main Menu options.

S The *Set* option invokes another sub-menu that will be discussed in the next section. This option provides test parameter set-up choices.

A The *All* test sequentially executes the Register, Memory, Pagemap, Internal, Encoder, External, Diagnose and Interrupt tests. Upon completion it reports the results test run and the number of the passes made for each test. In case of a test failure, you should look up the error log maintained by the Exec to determine the cause of the failure.

D The *Default* option sequentially executes the Register, Memory, Pagemap, Internal, External, Diagnose and Interrupt test. Upon completion, it reports the results of the test run and the number of the passes made at each test. In case of a test failure, you should look up the error log maintained by the Exec to determine the cause of the failure.

Q The *Quick* option sequentially executes the Register, Memory, Pagemap, Internal and Diagnose test. Upon completion, it reports back the result of the test run and the number of the passes made at each test.

R *pass=*

The *Register* option performs a board reset, and a write/read test on the status register, then reports the test results. It also reads the memory expansion jumper to determine whether there is any expansion memory related to the board.

M *pass=*

The *Memory* option performs a memory diagnostic on the Ethernet memory space. Note that the memory diagnostic accesses the board's memory through the Port B (system bus port). During the execution of this test, the diagnostic will write/read several unique patterns within Port B memory. If there is any discrepancy between the written and read patterns, the diagnostic will report an error along with the address at which it found the discrepancy.

P *pass=*

The *Pagemap* option performs write/read tests on the Page Map RAM. The test is done on the entire 1Kx16 Page Map area. During the execution of this test, the diagnostic will write/read several unique patterns inside the Page Map RAM. If there is any discrepancy between the written and read patterns, the diagnostic will report an error.

I *pass=*

The *Internal* option configures the 82586 so that it is logically disconnected from the SIA and the transceiver. Then the 82586 transmits and receives the packets in full duplex. The diagnostic determines whether or not the transmit/receive operation of the 82586 chip works correctly and if received packets are indeed the same as the one transmitted. Note that this test does not test the SIA and the transceiver chips.

E *pass=*

The *Encoder* option configures the 82586 in the same way as described for the Internal option, except that it uses special data encoding mode. In particular, this diagnostic uses Manchester encoding techniques on the transmit side and the NRZ encoding technique on the receive side.

External *pass=*

The *External* option performs a more thorough testing of the 82586 transmit/receive operation because, in this configuration, the SIA and the transceiver are connected to the 82586. This option requires that a coaxial cable with an industry standard carrier threshold be connected to the transceiver through a port provided at the back of the board. Note that this option implicitly performs diagnostics of the SIA and the transceiver pair.

DI The *Diagnose* option causes the execution of TDR and the 82586 DIAGNOSE commands. The DIAGNOSE command triggers an internal self-test of Collision counters, Exponential Backoff Timeout registers and the Slot time register. The TDR command performs a time-domain reflectometry test on the coaxial cable. This test should enable you to locate any shorts or opens in the cable.

INTR

The *Interrupt* option tests the ability of the board to interrupt the host CPU. The system bus backplane IACKjumper must be OUT. The diagnostic then creates an exception condition which causes the board to send out an interrupt signal to the CPU. It will do this twice: once with the Interrupt Enabled condition (in which case the CPU should see the interrupt) and once with the Interrupt Disabled (in which case the CPU should not see the interrupt).

The test assumes that the board interrupts the host CPU at priority 3 with vector number 0x75.

27.5. Set Option Sub-Menu

```
Second Ethernet Board Diag.  Rev x.x  MM/DD/YY Set Option Menu

Pass=                 Number of passes to be made for each test
HOE                   Halt on Error
GOE                   Go on Error

Command ==>
```

pass=

Typing **pass=5**, for example, sets up an environmental parameter so that any test chosen from the main menu run five times. Note that, until set, the pass variable will have a value that is the same as the Exec's pass variable (usually 1).

HOE

The *Halt on Error* option makes the diagnostic halt on first error that it detects. This is the default.

GOE

The *Go on Error* option makes the diagnostic continue even on detection of an error. It will make a complete run on any test, allowing the diagnostic to log all possible errors. Note that the HOE is the test default.

27.6. Error Messages

The Second Ethernet board diagnostics error messages are displayed on the monitor as well as logged. Following is a brief interpretation of these messages, organized by test name.

After giving the command to load the diagnostic, a check is made to ensure that the Second Ethernet board is indeed present. If the diagnostic does not find the board, then following message is displayed:

Second Ethernet Board is not present.

The diagnostic self-aborts at this point.

Register Test

This test first issues a board-level reset and then writes a pattern into the status register area followed by a read. If there is a discrepancy in the write/read patterns then the following error message will be logged:

Status register data errors: Exp: 0x   Obs: 0x.

The test also dumps out various informatory messages such as the size of system bus port, status of memory expansion switch, and so on.

Memory Test

This test writes several patterns in the system bus port (Port B) memory and then reads them back. If there is a discrepancy, one or more of the following messages would be logged:

```
Parity Error detected at addr: 0x.
```

This message means that while making memory accesses, the parity detection circuitry caught errors.

```
Memory Data errors: Addr -  ; Exp - ; Obs -.
```

This message means that there is a mismatch between the read/write data patterns.

Pagemap Test

This test writes several patterns in the Page Map RAM and then reads them back. If there is a discrepancy, the following message will be logged:

```
Pagemap Data Error: Addr - ; Exp - ; Obs -
```

This message means that the test detected mismatch between the write/read patterns.

Internal Loopback Test

In this test, the EDLC is configured to operate in the internal loopback mode. If the chip fails to configure itself, the following message is displayed:

```
Internal Loopback - Ethernet chip's configuration failed.
```

This is considered to be a catastrophic error and the diagnostic self-aborts at this point. Otherwise, the test continues further.

The remaining messages are listed under "Loopback Error Messages".

Encoder Test

In this test, the EDLC chip is configured so that it performs loopback so that it transmits data with Manchester encoding and receives data with NRZ encoding. If the chip fails to configure itself, the following message is displayed:

```
Encoder Loopback - Ethernet chip's configuration
failed.
```

This is considered to be a catastrophic error and the diagnostic self-aborts at this point.

The remaining messages are listed under "Loopback Error Messages".

External Loopback Test

In this test, the EDLC chip is reconfigured so that it performs external loopback mode of transmission/reception. If the chip fails to configure itself, the following message is displayed:

```
External Loopback - Ethernet chip's configuration
failed.
```

This is considered to be a catastrophic error and the diagnostic self-aborts at this point.

The remaining messages are listed under "Loopback Error Messages".

27.7. DIAGNOSE Test

This test checks the ability of the EDLC chip to execute DIAGNOSE and the TDR Action commands. First, the diagnostic issues a reset command. If the command could not be properly executed, then the following error message will be logged.

```
DIAGNOSE - Ethernet chip could not be reset.
```

Next the DIAGNOSE command is issued to the EDLC. If the diagnostic has a problem executing this command, then following error message will be displayed:

```
Cannot execute Ethernet chip's DIAGNOSE command.
```

Now the diagnostic will use the TDR feature of the EDLC chip to check the faults in the coaxial cable. If the diagnostic could not execute this command properly, the following error message will be logged:

```
TDR command could not be executed.
```

If the diagnostic finishes executing the DIAGNOSE command, the status of the cable and the transceiver is displayed. Please note that this particular test requires that a coaxial cable with a carrier signal be attached to the transceiver through the port provided in the back of the board.

Loopback Error Messages

Listed below are the error messages related to the Loopback tests.

```
Command Unit START command not accepted; timeout
occurred.
```

This error message is displayed if, after issuing the transmit command, the CUC field in the command word of the SCB block is not cleared by the EDLC chip.

```
Timeout waiting for Command Unit and Receive Unit Idle
status.
```

This error message is displayed if the EDLC' s CU or the RU fails to enter the idle state.

```
Timeout waiting for RU's No Resources Bit to be set.
```

This error message is displayed if the EDLC' s RU fails to enter the "No Resources" mode. The diagnostic expects the RU to enter this state just immediately after the issuance of START command.

```
Transmit completion bit not set.
```

If the diagnostic detects that the Completion bit in the Command Block is not set within a certain time period after the issuance of the "Transmit Action" command.

```
Transmit Command Block's Busy Bit still set.
```

If the diagnostic detects that the Busy bit in the Command Block is still set, indicating that the Transmit Command is still not finished, the error message shown above is logged.

```
Tx unsuccessful due to loss of CTS signal.
```

This error message would be displayed if transmission did not complete successfully. In particular, the EDLC experienced a loss of Clear-to-Send signal.

```
Tx unsuccessful due to loss of Carrier Sense.
```

The cause of this message is that the EDLC sensed a loss of Carrier signal.

```
Tx. unsuccessful due to DMA underrun.
```

This message is displayed if the EDLC reported a DMA underrun. For example, the data was not available to the EDLC at the time that DMA was ready for transmission.

```
Tx. stopped due to excessive collisions.
```

This message is displayed if the EDLC detected collisions that are greater in number than the maximum retries allowed (16).

```
Reception completion bit not set.
```

This message is displayed if the diagnostic detected that the Completion bit in the Receive Frame Descriptor was not set.

```
Receive Frame Descriptor's Busy bit still set.
```

If the diagnostic detects that, after a certain time period, the Busy bit in the RFD is still set, the messages shown above is displayed.

```
Received frame has CRC errors.
```

This message means that the RU detected CRC errors in the received packet.

```
Received frame has Alignment Errors.
```

If the received frame is misaligned, the RU sets the Alignment Error bit, causing the message shown above to appear.

```
RU has no resources to receive this frame.
```

When the incoming packets have to be discarded while the RU is in "Actively Receiving" mode due to lack of free receive buffers, the message shown above is displayed.

```
Receive unit indicates DMA overrun errors.
```

This message is displayed if the incoming packets were discarded because the DMA was unable to obtain the memory bus in time to transfer the already-received packets.

```
RU did not set EOF flag for the Rcvd frame.
```

Each buffer descriptor has an EOF flag indicating the fact that this was the last buffer used for a particular frame. Since the incoming frames are of length less than buffer size, each RBD's EOF flag must be set. Otherwise, the diagnostic reports an error.

```
Buffer Byte count not validated by Rcv. Unit.
```

The RU indicates the amount of the buffer size that is actually occupied by the packet. However, this count is not valid until the RU sets the "F" flag in the

RBD. If this is the case, the message shown above is displayed.

```
Mismatch between Tx. and Rcvd. data - Tx: %x   Rcv: %x
```

Finally, the diagnostic checks to make sure that the received data is indeed the same as that transmitted. If there is a mismatch, the message shown above is displayed.

27.8. Glossary

CU
: The Command Unit portion of the EDLC. Refer to the EDLC description, below.

EDLC
: The Ethernet Data Link Controller (EDLC) is a programmable Local Area Network (LAN) controller that performs the data transfer between the user memory and the network, performs error-detection and other link management functions. It mainly consists of a Command Unit (CU) and the Receive Unit (RU). The CPU sends the commands to the EDLC chip through the shared memory and the Command Unit executes them. The CU notifies the CPU of completion by setting certain bits in the shared memory.

Ethernet Memory Space
: Multibus/VMEbus memory space from 0x40000 through 0x80000. Some of these memory is shared with the host CPU.

Ethernet Register Space
: Multibus/VMEbus memory space from 0x88000 through 0x88800.

RU
: The Receive Unit handles all activities related to the frame reception.

SIA
: The Serial Interface Adapter provides the encoding/decoding capability for the EDLC and also drives the transceiver.

Second Ethernet Board
: An Intel 82586 Ethernet Data Link Controller-based board that is commonly used in gateways. This board serves as a primary or secondary Ethernet interface (depending upon base address switch settings) in Multibus-based Sun-2 systems, and as a secondary Ethernet interface in VMEbus-based Sun-3 and later systems.

TDR
: TDR or Time Domain Reflectometer is used to detect and locate cable faults caused by short or open circuits on the coaxial cable.

28

FDDI Diagnostic

28

FDDI Diagnostic

28.1. General Description

The Fiber Distributed Data Interface (FDDI) board is a high-speed network interface for Sun computers and workstations. FDDI provides the link-level interface to support current and future Sun network services on dual-attachment networks conforming to the FDDI standard.

The FDDI Diagnostic is designed to ensure that the FDDI board works correctly. To do so, it tests three functional subsystems, each of which includes several devices:

□ VME Host Interface logic

The VME Host Interface subsystem is composed of the VME Switch, Configuration, and Interrupt Vector registers. This section also includes the associated DMA controller logic, which consists of the DMA Control and Status, DMA Transfer Counter, Address, and Data Switch registers.

□ Node Processor Glue logic

The Node Processor Glue subsystem is composed of the NP control and interface subsystem (System Enable Register, Diagnostics LED Register, host command/status interrupt logic, and the M68901 MFP device), shared RAM (256 Kbytes DRAM), the serial port controller, IDPROM, EEPROM, EPROM, and the timer (AMD 9513a).

□ FDDI chip set and support subsystem

The FDDI chip set and support subsystem includes the Buffer RAM (256 Kbytes SRAM/), AMD Supernet Chip Set (RBC, DPC, FORMAC, ENDECs), Fiber Optics Transceivers (FOX), and the edge connectors.

28.2. What This Chapter Contains

Following an overview of the diagnostic, the Main Menu and submenus are discussed. Menu selections and optional arguments are listed, along with brief descriptions. The end of the chapter contains a table showing the recommended testing sequence, a list and interpretation of test messages, and a glossary.

28.3. Overview of the Diagnostic

The FDDI Diagnostic runs under the Exec and conforms to the interface standards of the Exec. All, Default, and Quick Test sequences are provided, as well as individual tests for testing specific components of the three functional subsystems. Parameters of the tests are given default values upon execution. Online help is provided.

The diagnostic also generates and logs error messages for later retrieval.

General Tests

Several of the tests in the diagnostic invoke other common tests, such as RAM pattern testing, write tests, and read tests. These general tests are described in this section and referenced in the descriptions of the tests that invoke them.

RAM Address Test

The *RAM Address Test* is performed in two steps. First the contents of the Address Register is written to RAM, from specified start to end. Next the Address Register is reset to the indicated start of RAM, and the data are read back and compared to the contents of the Address Register.

RAM Checker Test

The *RAM Checker Test* is also performed in two steps. First a fixed *pattern* is written to location *n*, and the *inverted pattern* is written to location *n+1* throughout the entire specified range. The contents of the RAM locations are then read back and compared to the patterns. The data read from location *n* are compared to *pattern*, then data read from location *n+1* are compared to *inverted pattern.*

RAM Constant Test

The *RAM Constant Test* first writes a specified constant pattern to a specified range in RAM. The contents of RAM are then read back and compared to the constant pattern.

RAM Diagonal Test

The *RAM Diagonal Test* writes an entire specified range in RAM with a pattern shifted one bit left for each location to the limit set by the data access size. For example, if the data access size is eight bits, the pattern is shifted eight times. When the shift limit is reached, the pattern is reloaded. The next step in the test reads back the data, beginning at the specified starting location, and compares to the write pattern.

RAM Triangle Test

The *RAM Triangle Test* writes *pattern* to location *n* and then *pattern* shifted left one bit plus one to locations *n+1* thereafter. The data are then read back, beginning at the specified starting location, and compared.

RAM Unique Test

The *RAM Unique Test* writes *pattern* to location *n* and then *pattern* plus the data access size (1, 2, or 4) to locations *n+1* thereafter. The data are then read back, beginning at the specified starting location, and compared.

RAM MATS Test

The *RAM MATS Test* writes *pattern* to location *n*, *Inverted pattern* to location *n+1*, and the *palindrome* of the original *pattern* to location *n+2*. (For example, the palindrome of pattern 54321 is 5432112345.) The data are then read back, beginning at the specified starting location, and compared.

RAM NTA Test

The comprehensive *RAM NTA Test* includes six sequences, each like a separate pattern test. The first sequence reads and compares data then replaces the value of the current location with a new value. The next sequence reads and compares the data with what was written in the previous sequence and replaces it with a new value. This process is repeated for each location. Long word (32-bit) data access is used. The following sequences comprise the RAM NTA Test:

- Sequence 1

  Data are read back and compared to 0x00000000. Next 0x01010101 is written back. Data are again read back and compared to 0x01010101. Finally 0xFFFFFFFF is written back.

- Sequence 2

  Data are read back and compared to 0xFFFFFFFF. Next 0xF1F1F1F1 is written back. Data are again read back and compared to 0xF1F1F1F1. Finally 0x33333333 is written back.

- Sequence 3

  Data are read back and compared to 0x33333333. Next 0xF0F0F0F0 is written back. Data are again read back and compared to 0xF0F0F0F0. Finally 0x0F0F0F0F is written back.

- Sequence 4

  Data are read back and compared to 0x0F0F0F0F. Next 0x55555555 is written back. Data are again read back and compared to 0x55555555. Finally 0xAAAAAAAA is written back.

- Sequence 5

  Data are read back and compared to 0xAAAAAAAA. Next 0x05050505 is written and immediately replaced with 0x88888888. Data are again read back and compared to 0x88888888. Finally 0x11111111 is written back.

- Sequence 6

  Data are read back and compared to 0x11111111. Next 0x00000000 is written and data are again read back and compared to 0x00000000.

RAM Random Test

The *RAM Random Test* fills RAM with random numbers then reads them back and compares the data to the expected values.

Read Test

The *Read Test* is designed to check the integrity of address/control/data lines to a particular register. The test reads only one location of the specified size, masks its value with the specified mask, and compares it to the expected value.

Write Test

The *Write Test* also checks the integrity of address/control/data lines to a particular register. The test writes one location of the specified size with the specified mask value.

Write/Read Test

The *Write/Read Test* also checks the integrity of address/control/data lines to a particular register. The test first writes the specified address with the specified masked value, then reads the data back, masks it, and compares it to the expected value.

March Test

The *March Test* is also designed to check the integrity of address/control/data lines to a particular register. The test first writes then reads and compares a data pattern to the expected value. It then shifts the pattern left one bit for each bit in the target register and repeats the process. Each value is masked upon writing and reading.

Checksum Test

For the EPROM and EEPROM, the *Checksum Test* reads and sums the value of each location from the specified start address to the end address. It then compares the calculated checksum to the stored checksum in the specified location. For the IDPROM, the checksum is calculated by a logical *exclusive or* of all locations but the final location, which contains the checksum value.

Tools and Utilities

In addition to the diagnostic tests, a submenu provides access to special tools and utilities. The utilities allow you to perform a variety of functions which are not tests in themselves but which aid the testing process. You can, for example, display the contents of locations in memory, display the memory map of a device under test, and edit the contents of specific memory locations.

28.4. Hardware Requirements

The following hardware is required to run the FDDI Diagnostic:

- A Sun-3/400 with cardcage
- A fully-populated FDDI board
- A monitor
- A keyboard
- A boot device (local disk, local tape, or remote disk over Ethernet)
- A loopback connector for the serial port
- A fiber loopback connector for the two fiber ports
- A "dumb" terminal attached to the serial port of the FDDI board (recommended for board test)

You may obtain loopback connectors by contacting Sun Customer Support at 1-800-872-4786. the OMAR part number of the fiber loopback connector is FFC. You can either order the serial port loopback connector from Customer Support or make your own. The serial port pinouts are shown in Appendix B.

28.5. User Interface

The user interface of the FDDI Diagnostic adheres to the menu standards of the Exec. A Main Menu and submenus are provided. Each option may be selected from a menu by typing the letter or letters displayed in upper case in the column on the left side of the menu.

Additional parameters may be specified on the command line. The FDDI Diagnostic supports the common parameters and environment variables (PAss=, for example) that may be used with other tests that run under the Exec. A summary of these parameters and variables is given in the "Command Parameters" and "Environment Variables" sections later in this chapter. For a complete discussion of the parameters, see Chapter 2, "Using the SunDiagnostic Executive."

All, Default, and Quick Test sequences are provided on the submenus. For most efficient testing, the tests should be run in an order that verifies the functionality of core devices first then proceeds to test the devices that are dependent upon the devices already confirmed as functional. The All Test option automatically executes the tests in the proper sequence. When you run individual tests for specific debugging, it is recommended that you run the tests in order according to the numbers that are displayed to the right of the option names on the menus. For example, tests in the menu numbered 1.1 should execute before tests in the menu numbered 1.2, and so on. The recommended order of execution for all tests in the diagnostic is given in the section entitled "Recommended Testing Sequence," later in this chapter.

To display online Help for any menu option, enter the following on the command line:

```
? option_name
```

ß.6. Starting the Diagnostic

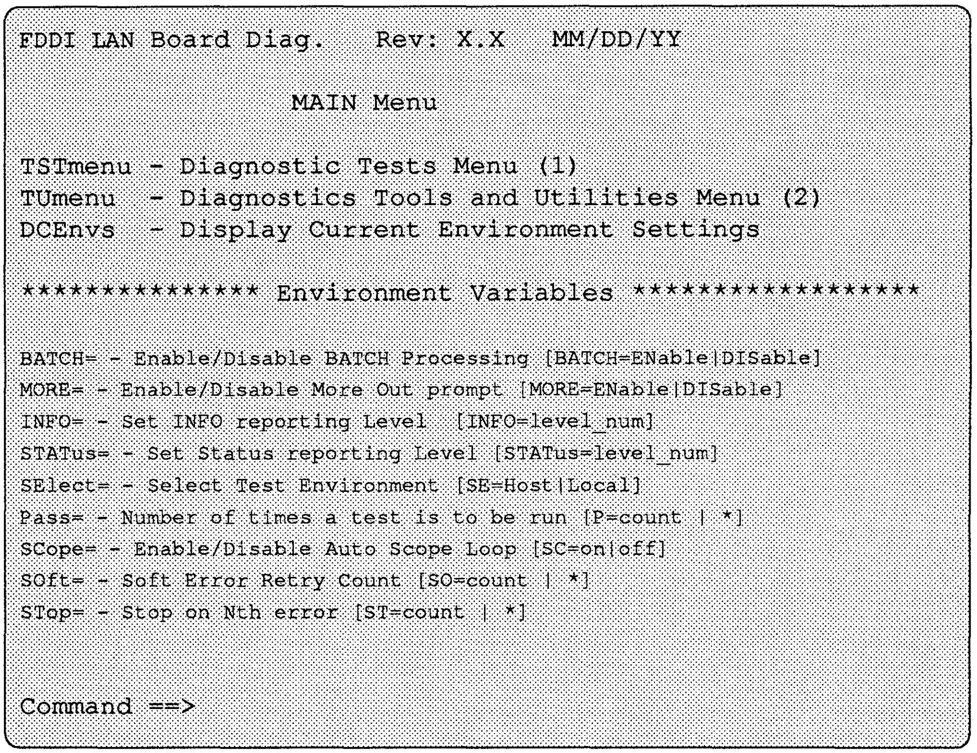
For information on starting the Exec, refer to Chapter 2, "Using the SunDiagnostic Executive." After you have started the Exec, choose the FDDI Diagnostic from the Diagnostics Menu.

28.7. The Diagnostic Menus

This section of the chapter provides a modular description of the FDDI Diagnostic, beginning with the Main Menu and working down through the options available on each of the submenus. A list of status, informational, and error messages for each test is given in the section entitled "Messages."

Main Menu

The Main Menu, which displays when you start the FDDI Diagnostic, provides access to the submenus of the individual tests:



```
FDDI LAN Board Diag.   Rev: X.X   MM/DD/YY

                    MAIN Menu

TSTmenu  - Diagnostic Tests Menu (1)
TUmenu   - Diagnostics Tools and Utilities Menu (2)
DCEnvs   - Display Current Environment Settings

*************** Environment Variables *****************

BATCH= - Enable/Disable BATCH Processing [BATCH=ENable|DISable]
MORE= - Enable/Disable More Out prompt [MORE=ENable|DISable]
INFO= - Set INFO reporting Level  [INFO=level_num]
STATus= - Set Status reporting Level [STATus=level_num]
SElect= - Select Test Environment [SE=Host|Local]
Pass= - Number of times a test is to be run [P=count | *]
SCope= - Enable/Disable Auto Scope Loop [SC=on|off]
SOft= - Soft Error Retry Count [SO=count | *]
STop= - Stop on Nth error [ST=count | *]


Command ==>
```

Command Parameters

The menu options in the FDDI Diagnostic take optional command parameters. You can specify the parameters by entering them on the command line when you run individual tests. If no parameters are specified, the default values are used.

The common command parameters are shown in the table that follows. Any other parameters that are specific to individual tests are described together with the tests to which they apply.

Parameter	*Description*	*Value*	*Default*
`PAss`	Determines the number of times a test or sequence of tests is run.	Non-negative integer (0 or * repeats indefinitely)	1
`OPtion`	Action performed by test.	R (Read) W (Write) WR (Write-then-Read) M (March)	March
`PATtern`	Data pattern used in testing.	Non-negative hex integer	0x55555555
`MAsk`	Data mask required by some tests.	Non-negative hex integer	0xFFFFFFFF

Parameter	*Description*	*Value*	*Default*
SAddress	Start address.	Non-negative hex integer	0 if not memory device. Start address of device if memory device.
EAddress	End address.	Non-negative hex integer	0 if not memory device. End address of device if memory device.
Dmode	Data or access mode.	B (Byte) W (Word) L (Long word)	Byte

Environmental Variables

The diagnostic testing environment is defined by the values of environmental variables. You can specify values for environmental variables by entering them on the command line, in the same way that you enter optional command parameters. When you set the value of an environmental variable on the command line of a particular diagnostic, the value is in effect for that session of the diagnostic. The Exec global environmental variables are unchanged.

At startup of the diagnostic, the values of the environmental variables are set to the values of the Exec environmental variables. For more information on environmental variables, refer to Chapter 2, "Using the SunDiagnostic Executive." The environmental variables (also shown on the Main Menu and Tools and Utilities Menu) are summarized in the table that follows.

Variable	*Description*	*Value*
BATCH	Enables or disables batch processing.	ENable/DISable
MORE	Enables or disables the More output prompt.	ENable/DISable
INFO	Sets the level of verbosity of informational messages.	Level number
STATus	Sets the level of verbosity of status messages.	Level number

Variable	*Description*	*Value*
`DUT`	Selects a previously mapped device to be the Device Under Test.	Name of mapped device
`SElect`	Selects the testing environment.	Host/Local
`Pass`	Determines the number of times a test or sequence of tests is run.	Non-negative integer (0 or * repeats indefinitely)
`SCope`	Enables or disables the automatic scope loop.	on/off
`SOft`	Number of times a test is retried upon detection of an error.	Non-negative integer (* retries indefinitely)
`STop`	Stops testing when Nth error is encountered.	Non-negative integer (* continues testing)

You may find some additional information to be helpful when using two of the environmental variables, `BATCH` and `DUT`:

□ *BATCH* — When this variable is enabled, you can enter a series of valid options from different menus. You must specify the options in order according to the numbers assigned on the menus to move from menu to menu. Tests in a series are executed successively without your intervention.

□ *DUT* — After you have specified the Device Under Test with this variable on the Tools and Utilities Menu, all subsequent testing is performed on that device. The device must have already been mapped, and the device name must be the same as that specified when mapping. Specifying "?" as the device name displays the current Device Under Test.

Diagnostic Tests Menu

TST

When you choose **TST** from the Main Menu, the FDDI Diagnostic Tests Menu is displayed:

```
FDDI LAN Board Diag.    Rev: X.X    MM/DD/YY

               (1) FDDI Diag. Tests Menu

A          - Perform All Test Sequence
D          - Perform the Default Test Sequence
Q          - Perform the Quick Test Sequence
VITmenu    - VME Interface Tests Menu (1.1)
NPGTmenu   - Node Processor Glue Tests Menu (1.2)
FDDImenu   - FDDI Tests Menu (1.3)
SCLoop     - << Scope Loop on a given Location >>
DRElog     - Display RAM Error Log
CRElog     - Clear RAM Error Log (** .sa and .ux versions only **)
DSStat     - Display/Save Test Statistics
CTStat     - Clear Test Statistics
DCEs       - Display Current Environment settings


Command ==>
```

This menu provides access to three submenus that you can use for in-depth testing of the functional subsystems of the FDDI board. It also allows you to execute a tight scope loop on a location under test, as well as display test statistics and current settings of environmental variables.

Perform All Test Sequence

A *pa= op= pat= ma= sa= ea= d=*

The *Perform All Test Sequence* option on the Main Menu executes the All Test option within each of the submenus. Each test is executed once, and the entire sequence is performed the number of times specified by `pa=`. This test sequence is long-running.

Perform the Default Test Sequence

D *pa= op= pat= ma= sa= ea= d=*

The *Perform the Default Test Sequence* option on the Main Menu executes the Default Test option within each of the submenus. Each test is executed once, and the entire sequence is performed the number of times specified by `pa=`. This test sequence is long-running.

Perform the Quick Test Sequence

Q *pa= op= pat= ma= sa= ea= d=*

The *Perform the Quick Test Sequence* option on the Main Menu executes the Quick Test option within each of the submenus. Each test is executed once, and the entire sequence is performed the number of times specified by `pa=`. This test sequence is relatively brief.

VME Interface Tests Menu

VIT

When you choose **VIT** from the FDDI Diagnostic Tests Menu, the VME Interface Tests Menu is displayed. This menu provides access to tests that verify the VME interface subsystem logic and functionality. For a complete description of the options on this menu, see the section entitled "VME Interface Tests Menu," later in this chapter.

Node Processor Glue Tests Menu

NPGT

When you choose **NPGT** from the FDDI Diagnostic Tests Menu, the Node Processor Glue Tests Menu is displayed. This menu provides access to tests that verify the Node Processor Glue subsystem. For a complete description of the options on this menu, see the section entitled "Node Processor Glue Tests Menu," later in this chapter.

FDDI Tests Menu

FDDI

When you choose *FDDI* from the FDDI Diagnostic Tests Menu, the FDDI Tests Menu is displayed. This menu provides access to tests that verify the logic and functionality of the FDDI board itself. For a complete description of the options on this menu, see the section entitled "FDDI Tests Menu," later in this chapter.

Scope Loop

SCL *sa= pat= op= d=*

The *Scope Loop on a Given Location* option on the FDDI Diagnostic Tests Menu allows you to perform a tight scope loop on the memory address specified by the hexadecimal value of sa=. For this option, the default value of op= is Read.

Display RAM Error Log

DRE

The *Display RAM Error Log* option on the FDDI Diagnostic Tests Menu allows you to view the contents of the Exec RAM error log.

Clear RAM Error Log

CRE

The *ClearRAM Error Log* option on the FDDI Diagnostic Tests Menu allows you to clear the contents of the Exec RAM error log if you are running the diagnostic as a standalone program. You *cannot* use this option when running the diagnostic under the Exec.

Display/Save Test Statistics

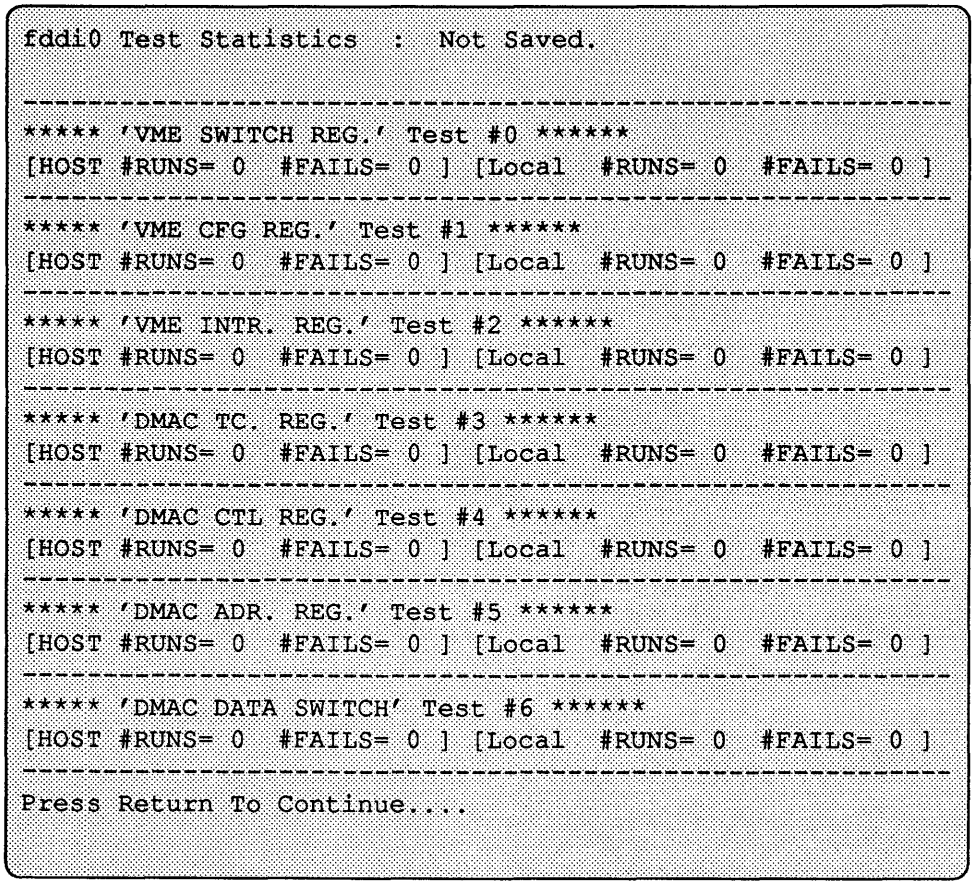
DSS *sa= ea= op=*

The *Display/Save Test Statistics* option on the FDDI Diagnostic Tests Menu allows you to save and display test statistics gathered since the start of the diagnostic. The statistics are saved to the Exec error log. You may choose the range of statistics to save and display by choosing starting and ending tests. The values of parameters used with this command have specific meanings; they are summarized in the following table:

Parameter	*Description*	*Default*
sa	Starting test number.	0

Parameter	*Description*	*Default*
ea	Ending test number.	Last test in series
op	Action option.	Write

A sample result of this option is shown in the following display:



```
fddi0 Test Statistics  :  Not Saved.


---------------------------------------------------------
***** 'VME SWITCH REG.' Test #0 ******
[HOST #RUNS= 0  #FAILS= 0 ] [Local  #RUNS= 0  #FAILS= 0 ]
---------------------------------------------------------
***** 'VME CFG REG.' Test #1 ******
[HOST #RUNS= 0  #FAILS= 0 ] [Local  #RUNS= 0  #FAILS= 0 ]
---------------------------------------------------------
***** 'VME INTR. REG.' Test #2 ******
[HOST #RUNS= 0  #FAILS= 0 ] [Local  #RUNS= 0  #FAILS= 0 ]
---------------------------------------------------------
***** 'DMAC TC. REG.' Test #3 ******
[HOST #RUNS= 0  #FAILS= 0 ] [Local  #RUNS= 0  #FAILS= 0 ]
---------------------------------------------------------
***** 'DMAC CTL REG.' Test #4 ******
[HOST #RUNS= 0  #FAILS= 0 ] [Local  #RUNS= 0  #FAILS= 0 ]
---------------------------------------------------------
***** 'DMAC ADR. REG.' Test #5 ******
[HOST #RUNS= 0  #FAILS= 0 ] [Local  #RUNS= 0  #FAILS= 0 ]
---------------------------------------------------------
***** 'DMAC DATA SWITCH' Test #6 ******
[HOST #RUNS= 0  #FAILS= 0 ] [Local  #RUNS= 0  #FAILS= 0 ]
---------------------------------------------------------
Press Return To Continue....
```

Clear Test Statistics

CTS

The *Clear Test Statistics* option on the FDDI Diagnostic Tests Menu clears all statistics generated by all tests.

CAUTION **This option results in the loss of all test statistics that have not been saved.**

Display Current Environment Settings

DCE

The *Display Current Environment Settings* option on the FDDI Diagnostic Tests Menu displays the current values of all environmental variables. A sample result of this option is shown in the following display:

```
Command ==>dce
---------------------------------------------------------------
FDDI Diagnostics Environment Setting:

Device Under Test :'fddi0' Virtual Base Address:0x0003e000
Device VME address: 0x01600000
BATCH Processing  : 'DISable'
Test Environment  : 'HOST'
More Out Prompt   : 'DISable'
Scope Loop        : 'off'
INFO   Level      : 1
STATus Level      : 1
Test Passcount        : 1
Stop On Nth Error     : *
Soft Error retry Count: 0
---------------------------------------------------------------

Please Press the 'Return' Key to Continue.
```

Diagnostics Tools and Utilities Menu

TU

When you choose **TU** from the Main Menu, the FDDI Diagnostic Tools and Utilities Menu is displayed:

```
FDDI LAN Board Diag.   Rev: X.X   MM/DD/YY

          (2) FDDI Diag. Utilities Menu

DCE         - Display Current Environment settings
EXC         - Execute Code In NP RAM [SA=entry_point]
MAP         - Map additional FDDI Boards in VME
UMAP        - Unmap FDDI Board from VME
DMM         - Display DUT Memory Map
DME         - DUT Memory Examine Utility
RSTnp       - Tell NP to Reset
GNMI        - Generate an NMI on NP
DCStat      - Display Controller Status
*DNLD       - Download Object Code onto NP RAM
*DLE        - Download and Execute Object Code on NP RAM

*************** Environment Variables *****************

DUT=     - VME mapped FDDI Board To be tested [DUT=FDDI0|board_nam]
SElect=  - Select Test Environment [SE=Host|Local]


Command ==>
```

This menu provides access to functions which are not tests but which aid testing and debugging. The two options marked with * on the menu — *Download*

Object Code onto NP RAM and *Download and Execute Object Code on NP RAM* — are available in the standalone version of the diagnostic only. These two options cannot be run under the Exec. Parameters of options on the Diagnostic Tools and Utilities Menu have special meanings. They are described in this section together with their corresponding options.

Display Current Environment Settings

DCE

The *Display Current Environment Settings* option on the FDDI Diagnostic Tools and Utilities Menu displays the current values of all environmental variables. For a sample display, see the discussion of this option under the FDDI Diagnostic Tools and Utilities Menu earlier in this chapter.

Execute Code in NP RAM

EXC *sa=*

The *Execute Code in NP RAM* option on the FDDI Diagnostic Tools and Utilities Menu allows you to execute a previously downloaded code in RAM or ROM on the Node Processor. The value of `sa=` specifies a valid entry-point address for the code to be executed on the Node Processor. The entry-point address must be an absolute address relative to 0. (Refer to the FDDI board memory map by using the Display DUT Memory Map option on the Tools and Utilities Menu, described later in this chapter.)

Map Additional FDDI Boards in VME

MAP *dev= ph= sz= vme= iv=*

The *Map Additional FDDI Boards in VME* option on the FDDI Diagnostic Tools and Utilities Menu allows you to map additional FDDI boards for testing. Before diagnostics can be performed on an FDDI board, you must map the board. During mapping, you can specify the board name, physical address, size (memory space), and board interrupt vector.

The parameters specific to this command are summarized in the table that follows. The `dev=` and `ph=` parameters are mandatory, the rest are optional.

Parameter	*Description*	*Default*
`dev`	Device name, as many as 14 ASCII characters. Specifying "?" causes a list of all currently mapped devices to be displayed.	None
`ph`	Hex physical or VME address of the board.	0
`sz`	Hex size of the board, in bytes.	200000
`vme`	VME access type, same as the VME device type specified in `/dev`, the special device directory.	vme32d32

Parameter	*Description*	*Default*
`iv`	Non-zero hex integer specifying the VME interrupt vector number of the board.	40

Unmap FDDI Board from VME

UMAP *dev=*

The *Unmap FDDI Board from VME* option on the FDDI Diagnostic Tools and Utilities Menu performs the opposite function of the MAP command—it removes a mapped device from the memory map. Any device removed from the map with this command must be mapped again before it can be tested by the diagnostic. The `dev=` parameter, which specifies the device name, is mandatory.

Display DUT Memory Map

DMM

The *Display DUT Memory Map* option on the FDDI Diagnostic Tools and Utilities Menu displays the memory map of the Device Under Test. The memory map shows information about the devices on board that are mapped and accessible. The following sample shows a DUT memory map display:

```
Section Name : xxxxxxx

VME Access (Yes), NP Access (Yes)

Section Type = n   (0=>R, 1=>W, 2=>R/W)

Data Transfer size = n Byte(s)

Start Offset = 0xnn, End Offset = 0xnn, Block Size = n Bytes
```

DUT Memory Examine Utility

DME

The *DUT Memory Examine Utility* option on the FDDI Diagnostic Tools and Utilities Menu invokes the DUT Memory Examine Utility (also known as the peek-'n-poke utility). This utility allows you to view the contents of a memory location, edit a memory location, and perform a block dump or block fill of a contiguous block of memory.

The utility also provides a reverse Polish notation calculator that performs addition, subtraction, multiplication, division, unary operations, negation, exclusive/inclusive OR, AND, and bitwise shift operations on integers. The results can be stored and used as a part of the utility. The command and value stack are fixed at 100 elements each. Integers may be entered as decimal, octal, or hexadecimal, but all output is hexadecimal.

Online Help provides a list of valid commands and syntax. To request Help, type the **`help`** command.

Sample output from this option follows:

```
Command ==>dme
O.K. help
<pattern> <number_of_elements> <address> [fill|bfill|wfill|lfill]
  Fills memory starting at <address> up to and including
  <address> + <number_of_elements * [byte|word|long]> with
  <pattern>

<number_of_elements> <address> [dump|bdump|wdump|ldump]
  Dumps contents starting at <address> up to and including
  <address> + <number_of_elements * [byte|word|long]> to screen

<address> [?|b?|w?|l?] Reads contents of <address> and places
 Read value on Top of Stack.  To get value type '.' or '='

<value> <address> [!|b!|w!|l!] Writes <Value> into <address>

Stack Operations :

 $- flush Value Stack,  @ - flush command stack
 '.' or '='  Displays Top of Value stack
 <n> dsp  - displays n elements from the top of value stack
 <value1> <value2> ['+' (add) | '-' (sub) | '/' (div) | '*' (mult)
 | '^' (Xor) | '|' (Ior) | '<<' (Shl) | '>>' (Shr) ]
    <value2> operates on  <value1>  Result goes on top of Stack

 <ESC>, 'exit', 'quit', '^X' -- Exit.
O.K.  <ESC>
```

Tell NP to Reset

RST

The *Tell NP to Reset* option on the FDDI Diagnostic Tools and Utilities Menu causes the Node Processor to perform a hardware reset.

Generate an NMI on NP

GNMI

The *Generate an NMI on NP* option on the FDDI Diagnostic Tools and Utilities Menu causes a non-maskable interrupt (NMI) to be generated by the Node Processor. This action is also known as a software reset.

Display Controller Status

DCS

The *Display Controller Status* option on the FDDI Diagnostic Tools and Utilities Menu allows you to examine the current status of the Node Processor controller board by interpreting the diagnostic LED register on the Node Processor. The following is a sample display of the diagnostic LED register:

```
NP DIAG STATUS : [~0x3e] ==> [OK] [Test state (#0x3e)]
Command ==>
```

In this sample, the field containing `[OK]` indicates the current error state. If an error state existed, this field would contain `[FAILED]`. The field containing `[Test state (#0x3e)]` can indicate either the test number or the software state of the controller. In this case, the controller is in the 0x3e state, which indicates an extended diagnostic prompt for input.

Download Object Code onto NP RAM

DNLD

The *Download Object Code onto NP RAM* option on the FDDI Diagnostic Tools and Utilities Menu allows you to download firmware through the host VME interface onto a fixed location on the Node Processor shared RAM. This option is only available under SunOS; it cannot be used when running under the Exec.

If you are running the diagnostic under SunOS, you will be prompted for the name of the file to download. Any file to be downloaded must be in the `a.out` executable file format (MC68000 family object code) and linked to start at location 0xc4xxx. The default directory is always assumed to be the current directory.

Download and Execute Object Code on NP RAM

DLE

The *Download and Execute Object Code on NP RAM* option on the FDDI Diagnostic Tools and Utilities Menu is available under SunOS only; it cannot be used when running under the Exec. This option performs two other options on the menu in the following order:

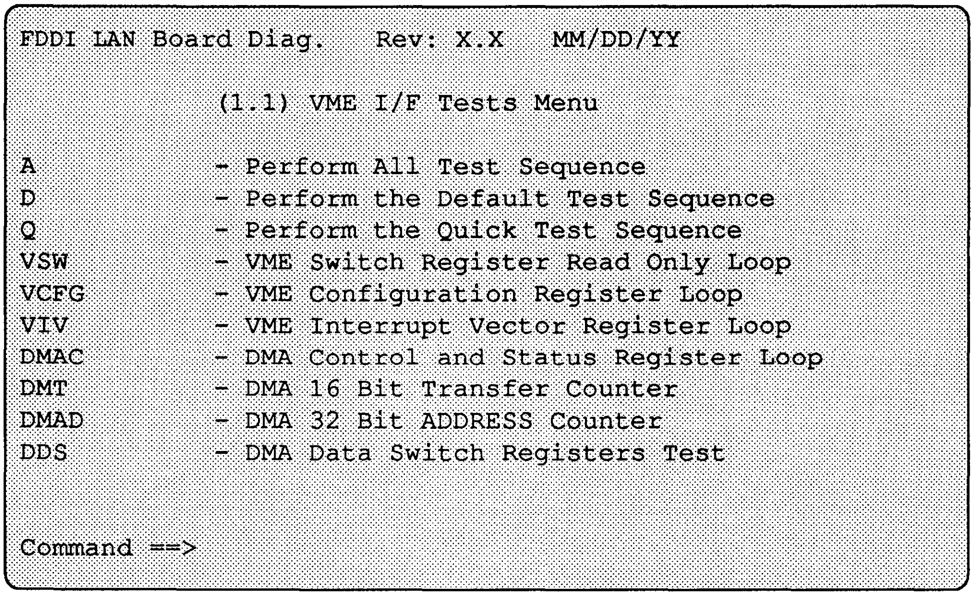
1. Download Object Code onto NP RAM
2. Execute Code in NP RAM

This option does not require you to specify an entry-point address, as does the Execute Code in NP RAM option. This option requires that the entry-point address must always be 0xc4000, specified in the `a.out` file. The execution address is identical to the entry-point address.

VME Interface Tests Menu

VIT

When you choose **VIT** from the FDDI Diagnostic Tests Menu, the VME Interface Tests Menu is displayed:



```
FDDI LAN Board Diag.   Rev: X.X   MM/DD/YY

               (1.1) VME I/F Tests Menu

A              - Perform All Test Sequence
D              - Perform the Default Test Sequence
Q              - Perform the Quick Test Sequence
VSW            - VME Switch Register Read Only Loop
VCFG           - VME Configuration Register Loop
VIV            - VME Interrupt Vector Register Loop
DMAC           - DMA Control and Status Register Loop
DMT            - DMA 16 Bit Transfer Counter
DMAD           - DMA 32 Bit ADDRESS Counter
DDS            - DMA Data Switch Registers Test


Command ==>
```

This menu provides access to tests that allow you to verify the VME interface, as well as basic DMA logic and functionality.

A *pa= op= pat= ma= sa= ea= d=*

The *Perform All Test Sequence* option on the VME Interface Tests Menu executes all of the tests on the menu in the order listed.

D *pa= op= pat= ma= sa= ea= d=*

The *Perform the Default Test Sequence* option on the VME Interface Tests Menu executes all of the tests on the menu in the order listed.

Q *pa= op= pat= ma= sa= ea= d=*

The *Perform the Quick Test Sequence* option on the VME Interface Tests Menu executes all of the tests on the menu in the order listed.

VSW *pa= pat= ma=*

The *VME Switch Register Read Only Loop* option on the VME Interface Tests Menu performs the Read Test on the VME Switch Register. The test is useful for determining whether a Read Strobe and the proper signal sequence are being generated. By running this test and changing the value in the VME Switch Register, you can test the validity of the data line connection to the register.

The value of `pat=` should be a hexadecimal VME switch setting. If you do not specify a value for `pat=`, the test compares the value in the VME Switch Register to the default VME address of the board.

VCFG *pa= op= pat= ma=*

The *VME Configuration Register Loop* option on the VME Interface Tests Menu performs the March Test on the Configuration Register. It verifies the control/data/address lines used by the register, as well as the VME arbitration logic.

VIV *pa= op= pat= ma=*

The *VME Interrupt Vector Register Loop* option on the VME Interface Tests Menu performs the same verification on the Interrupt Vector Register that the VME Configuration Register Loop option, just described, performs on the Configuration Register.

DMAC *pa= op= pat= ma=*

The *DMA Control and Status Register Loop* option on the VME Interface Tests Menu performs the March Test on the DMA Control and Status Register. The test verifies the control/data/address lines associated with the register.

DMT *pa= op= pat= ma=*

The *DMA 16 Bit Transfer Counter* option on the VME Interface Tests Menu performs the same verification on the control/data/address lines associated with the DMA Transfer Counter that the DMA Control and Status Register Loop option, just described, performs on the lines associated with the DMA Control and Status Register.

DMAD *pa= op= pat= ma=*

The *DMA 32 Bit ADDRESS Counter* option on the VME Interface Tests Menu performs the same verification on the control/data/address lines associated with the DMA Address Counter that the DMA 16 Bit Transfer Counter option, just described, performs on the lines associated with the DMA Transfer Counter.

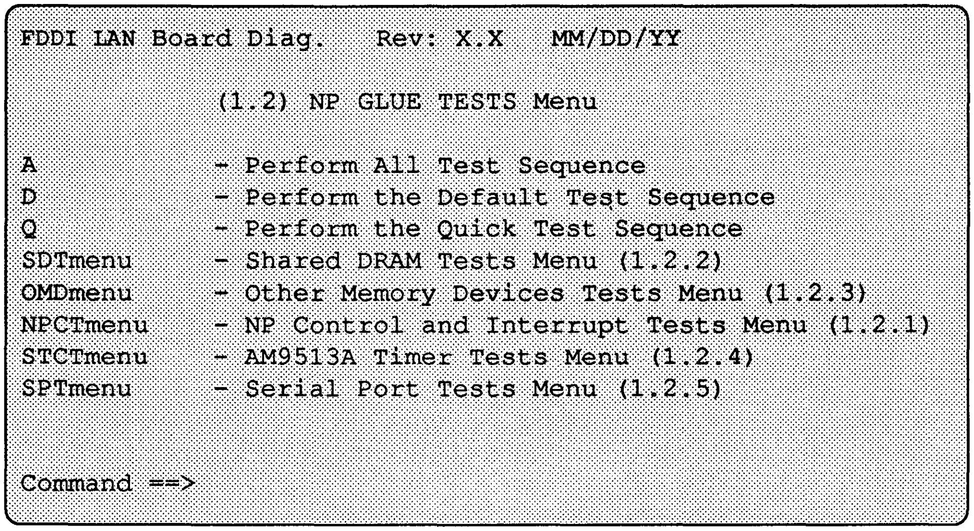
DDS *pa= op= pat= ma=*

The *DMA Data Switch Registers Test* option on the VME Interface Tests Menu performs the March Test on each of the four DMA Data Switch Registers. The test verifies the control/data/address lines associated with each of the registers. Because the Data Switch Registers provide the access to the DMA data pipe, this test, together with other DMA tests, can help you to identify a DMA problem along any segment of the DMA pipe.

Node Processor Glue Tests Menu

NPGT

When you choose **NPGT** from the FDDI Diagnostic Tests Menu, the Node Processor Glue Tests Menu is displayed:



```
FDDI LAN Board Diag.   Rev: X.X   MM/DD/YY

              (1.2) NP GLUE TESTS Menu

A             - Perform All Test Sequence
D             - Perform the Default Test Sequence
Q             - Perform the Quick Test Sequence
SDTmenu       - Shared DRAM Tests Menu (1.2.2)
OMDmenu       - Other Memory Devices Tests Menu (1.2.3)
NPCTmenu      - NP Control and Interrupt Tests Menu (1.2.1)
STCTmenu      - AM9513A Timer Tests Menu (1.2.4)
SPTmenu       - Serial Port Tests Menu (1.2.5)


Command ==>
```

This menu allows you to narrow the test field by providing tests for each of the functional subdevices of the Node Processor Glue. These subdevices include the following:

- Shared DRAM
- Serial port controller
- Diagnostics LED Register
- Other memory devices such as IDPROM, EEPROM, and EPROM
- Node Processor control and interrupt logic
- AM9513A timer

A *pa= op= pat= ma= sa= ea= d=*

The *Perform All Test Sequence* option on the Node Processor Glue Tests Menu executes the All Test Sequence in each of the submenus in the numerical order shown on this menu.

D *pa= op= pat= ma= sa= ea= d=*

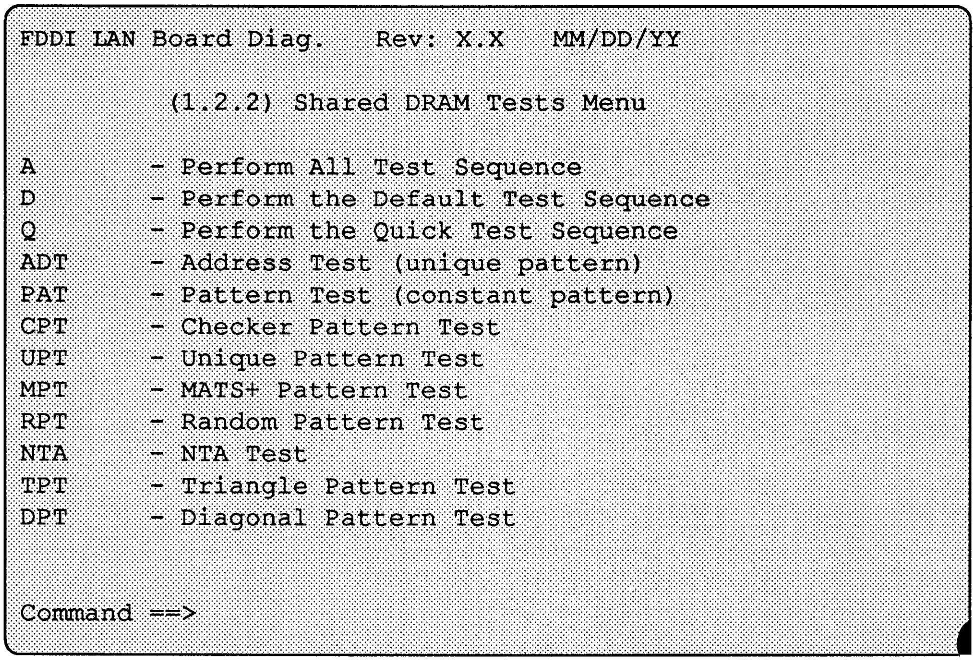
The *Perform the Default Test Sequence* option on the Node Processor Glue Tests Menu executes the Default Test Sequence in each of the submenus in the numerical order shown on this menu.

Q *pa= op= pat= ma= sa= ea= d=*

The *Perform the Quick Test Sequence* option on the Node Processor Glue Tests Menu executes the Quick Test Sequence in each of the submenus in the numerical order shown on this menu.

SDT

When you choose **SDT** from the Node Processor Glue Tests Menu, the Shared DRAM Tests Menu is displayed:



```
FDDI LAN Board Diag.   Rev: X.X   MM/DD/YY

         (1.2.2) Shared DRAM Tests Menu

A        - Perform All Test Sequence
D        - Perform the Default Test Sequence
Q        - Perform the Quick Test Sequence
ADT      - Address Test (unique pattern)
PAT      - Pattern Test (constant pattern)
CPT      - Checker Pattern Test
UPT      - Unique Pattern Test
MPT      - MATS+ Pattern Test
RPT      - Random Pattern Test
NTA      - NTA Test
TPT      - Triangle Pattern Test
DPT      - Diagonal Pattern Test


Command ==>
```

The options on this menu allow you to test the logic and functionality of the Shared DRAM. The default values of `sa=` and `ea=` are the start and end of DRAM, respectively.

A *pa= pat= ma= sa= ea= d=*

The *Perform All Test Sequence* option on the Shared DRAM Tests Menu executes all of the tests on the menu in the order listed.

D *pa= pat= ma= sa= ea= d=*

The *Perform the Default Test Sequence* option on the Shared DRAM Tests Menu executes all of the tests on the menu in the order listed.

Q *pa= pat= ma= sa= ea= d=*

The *Perform the Quick Test Sequence* option on the Shared DRAM Tests Menu executes the following tests in order:

1. Address Test
2. Checker Pattern Test
3. Unique Pattern Test
4. Random Pattern Test
5. NTA Test

The other options on this menu execute the specified standard tests, as described in the section entitled "General Tests," earlier in this chapter. You may specify the same parameters as those listed with the All, Default, and Quick Test Sequences.

OMD

When you choose **OMD** from the Node Processor Glue Tests Menu, the Other Memory Devices Tests Menu is displayed:

```
FDDI LAN Board Diag.   Rev: X.X   MM/DD/YY

        (1.2.3) Other Mem. Dev. Test Menu

A            - Perform All Test Sequence
D            - Perform the Default Test Sequence
Q            - Perform the Quick Test Sequence
EEPA         - EEPROM Read Access Loop
EEPC         - EEPROM Checksum/Data Test
EEPM         - EEPROM March Test
EPA          - EPROM Read Access Loop {SE=Local Only}
EPC          - EPROM Checksum/Data Test {SE=Local Only}
IDPA         - IDPROM Read Access Loop
IDPC         - IDPROM Checksum/Data Test


Command ==>
```

The options on this menu allow you to test the logic and functionality of the EEPROM, EPROM, and IDPROM. Some of these memory devices are accessible only through the Node Processor and cannot be accessed by the Host. The tests which verify these devices (EEPROM Read Access Loop and EPROM Checksum/Data Test) must be run in local environment mode only. You should specify `se=1` on the Main Menu or on the Tools and Utilities Menu before you run these tests.

A *pa=*

The *Perform All Test Sequence* option on the Other Memory Devices Tests Menu executes all of the tests on the menu, except one, in order. The EEPROM March Test is not performed in this sequence because the number of write accesses to the EEPROM is limited.

D *pa=*

The *Perform the Default Test Sequence* option on the Other Memory Devices Tests Menu executes all of the tests on the menu, except one, in order. The EEPROM March Test is not executed because the number of write operations that can be performed to the EEPROM is limited. You may run this test separately.

Q *pa=*
The *Perform the Quick Test Sequence* option on the Other Memory Devices Tests Menu executes the following tests in order:

1. EEPROM Read Access Loop
2. EEPROM March Test
3. EPROM Read Access Loop
4. IDPROM Read Access Loop
5. IDPROM Checksum/Data Test

EEPA *pa=*
The *EEPROM Read Access Loop* option on the Other Memory Devices Tests Menu allows you to verify the control/address/data lines of the EEPROM by performing a dump of all data in the EEPROM to the screen. For this option to execute correctly, the value of `info=` must be set to 2 or higher.

EEPC *pa=*
The *EEPROM Checksum/Data Test* option on the Other Memory Devices Tests Menu performs the standard Checksum Test on the EEPROM.

EEPM *pa=*
The *EEPROM March Test* option on the Other Memory Devices Tests Menu performs the March Test on every location in the EEPROM. Because data in the EEPROM is lost when write operations are performed during the test, you are prompted for confirmation that you want to run the test before the test begins.

NOTE *This test is slow-running because of the time required for each write access to the device. It is recommended that you run this test infrequently.*

EPA *pa=*
The *EPROM Read Access Loop* option on the Other Memory Devices Tests Menu allows you to verify the control/address/data lines of the EPROM by performing a dump of all data in the EPROM to the screen.

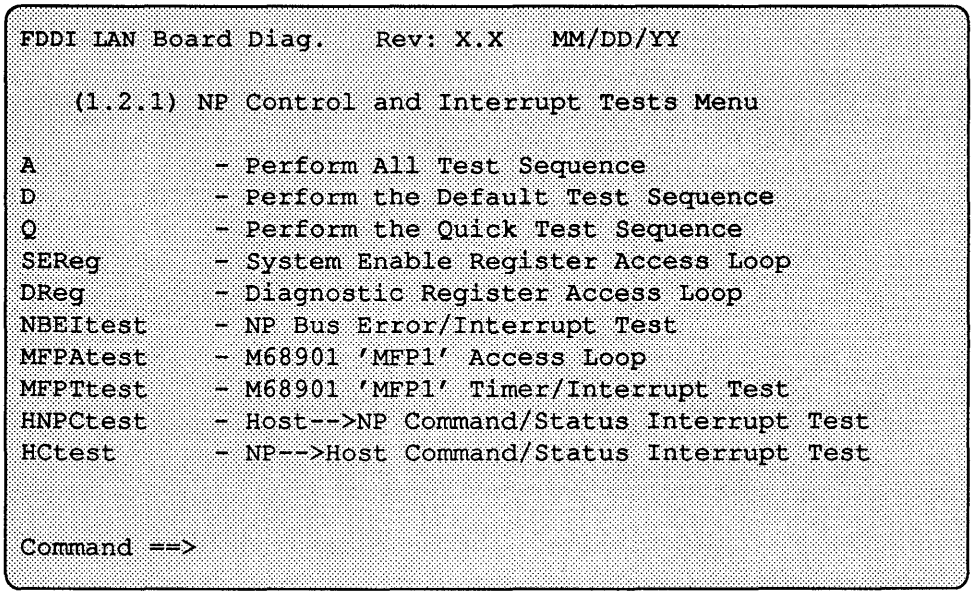
EPC *pa=*
The *EPROM Checksum/Data Test* option on the Other Memory Devices Tests Menu performs the standard Checksum Test on the EPROM.

IDPA *pa=*
The *IDPROM Read Access Loop* option on the Other Memory Devices Tests Menu allows you to verify the control/address/data lines of the IDPROM by performing a dump of all data in the IDPROM to the screen.

IDPC *pa=*
The *IDPROM Checksum/Data Test* option on the Other Memory Devices Tests Menu performs the standard Checksum Test on the IDPROM.

NPCT

When you choose **NPCT** from the Node Processor Glue Tests Menu, the NP Control and Interrupt Tests Menu is displayed:



```
FDDI LAN Board Diag.   Rev: X.X   MM/DD/YY

   (1.2.1) NP Control and Interrupt Tests Menu

A           - Perform All Test Sequence
D           - Perform the Default Test Sequence
Q           - Perform the Quick Test Sequence
SEReg       - System Enable Register Access Loop
DReg        - Diagnostic Register Access Loop
NBEItest    - NP Bus Error/Interrupt Test
MFPAtest    - M68901 'MFP1' Access Loop
MFPTtest    - M68901 'MFP1' Timer/Interrupt Test
HNPCtest    - Host-->NP Command/Status Interrupt Test
HCtest      - NP-->Host Command/Status Interrupt Test


Command ==>
```

The options on this menu allow you to test the logic and functionality of the Node Processor's control and interrupt system. This system contains the following hardware components:

- System Enable Register
- Diagnostics LED Register
- M68901 (MFP) device
- Registers that generate Interrupt, Hardware Reset, and NMI signals

A *pa= op= pat= ma=*

The *Perform All Test Sequence* option on the NP Control and Interrupt Tests Menu executes all of the tests on the menu in the order listed.

D *pa= op= pat= ma=*

The *Perform the Default Test Sequence* option on the NP Control and Interrupt Tests Menu executes all of the tests on the menu in the order listed.

Q *pa= op= pat= ma=*

The *Perform the Quick Test Sequence* option on the NP Control and Interrupt Tests Menu executes all of the tests on the menu in the order listed.

SER *pa= op= pat=*

The *System Enable Register Access Loop* option on the NP Control and Interrupt Tests Menu tests the control/data/address lines associated with the System Enable Register. The default mask for this test is 0xFFFFFFFE.

DR *pa= op= pat= ma=*

The *Diagnostic Register Access Loop* option on the NP Control and Interrupt Tests Menu tests the control/data/address lines associated with the Diagnostics LED Register.

NBEI *pa=*

The *NP Bus Error/Interrupt Test* option on the NP Control and Interrupt Tests Menu tests the bus error generation logic of the Node Processor. The test first clears the region reserved on the controller board for reporting Node Processor bus errors. It then writes to EPROM location 0 to generate a bus error on the controller. Finally the location used for reporting bus errors is checked for verification that a bus error occurred.

MFPA *pa= op= pat= ma=*

The *M68901 'MFP1' Access Loop* option on the NP Control and Interrupt Tests Menu tests the control/data/address lines associated with the MFP (Multi-Function Peripheral) M68901 device. It performs the test specified as the value of op= (default March Test) on the STC Interrupt Enable A Register.

MFPT *pa=*

The *M68901 'MFP1' Timer/Interrupt Test* option on the NP Control and Interrupt Tests Menu verifies functions of the MFP device that may be used by applications. First the MFP device is initialized to a known state. The test then sets timer A to generate an interrupt after one clock tick. Next the timer is enabled and, after the special region used for reporting MFP interrupts is cleared, the timer is started. Finally the value in that region is read and compared against the expected interrupt bit.

HNPC *pa=*

The *Host to NP Command/Status Interrupt Test* option on the NP Control and Interrupt Tests Menu tests the functionality of the Magic location and associated logic in generating a host command interrupt on the controller board. The test first initializes the MFP, enables the appropriate interrupt bit, and clears the region used for reporting MFP interrupts. The host then writes to the Magic location to generate the interrupt. The region associated with the MFP interrupt is read and checked against the expected interrupt information. If the value does not compare, the MFP Interrupt Pending Register is read and its value is compared against the expected interrupt information. If the values do not match, an error message is displayed.

If the interrupt was not generated, and the interrupt pending bit was set in the MFP, a problem may exist in the interrupt generation circuitry between the MFP and the M68020 processor. If, however, the interrupt pending bit was not set, a problem may exist either in the associated PAL or in another device between the Magic location circuitry and the MFP.

NPHC *pa=*

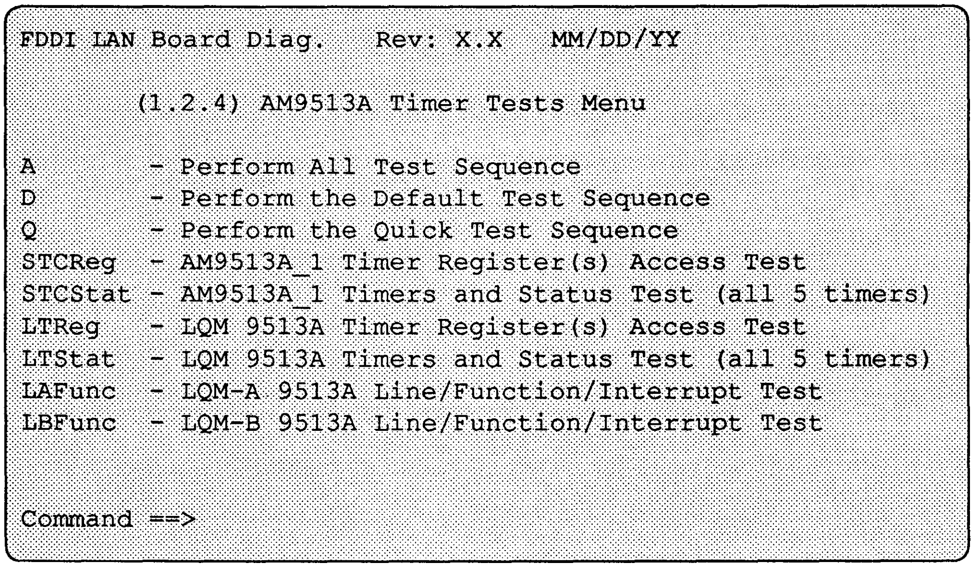
The *NP to Host Command/Status Interrupt Test* option on the NP Control and Interrupt Tests Menu tests the ability of the Node Processor to generate an interrupt on the host. The test performs the following steps:

1. Resets the interrupt pending bit in the controller's VME Configuration Register and disables the controller's ability to generate interrupts.
2. Installs the appropriate interrupt handler on the host.
3. Downloads and executes a special firmware module which writes a value to the Magic location associated with the generation of an interrupt on the host.
4. Initializes the VME Vector Register on the controller to a preselected vector, sets the arbitration and host interrupt levels to 3, and asserts the interrupt enable bit.

An interrupt should be generated on the host. If an interrupt is not detected, the test checks the VME Configuration Register for the interrupt pending bit. The value in the Magic location is checked to ensure that the generation of the interrupt was intentional.

STCT

When you choose **STCT** from the Node Processor Glue Tests Menu, the AM9513A Timer Tests Menu is displayed:



```
FDDI LAN Board Diag.    Rev: X.X    MM/DD/YY

        (1.2.4) AM9513A Timer Tests Menu

A        - Perform All Test Sequence
D        - Perform the Default Test Sequence
Q        - Perform the Quick Test Sequence
STCReg   - AM9513A_1 Timer Register(s) Access Test
STCStat  - AM9513A_1 Timers and Status Test (all 5 timers)
LTReg    - LQM 9513A Timer Register(s) Access Test
LTStat   - LQM 9513A Timers and Status Test (all 5 timers)
LAFunc   - LQM-A 9513A Line/Function/Interrupt Test
LBFunc   - LQM-B 9513A Line/Function/Interrupt Test



Command ==>
```

The options on this menu allow you to test the AM9513A timer, its associated logic, and functionality.

A *pa= op= pat= ma=*

The *Perform All Test Sequence* option on the AM9513A Timer Tests Menu executes all of the tests on the menu in the order listed.

D *pa= op= pat= ma=*

The *Perform the Default Test Sequence* option on the AM9513A Timer Tests Menu executes all of the tests on the menu in the order listed.

Q *pa= op= pat= ma=*

The *Perform the Quick Test Sequence* option on the AM9513A Timer Tests Menu executes the following tests on the menu:

1. AM9513A_1 Timer Register(s) Access Test
2. LQM 9513A Timer Register(s) Access Test

STCR *pa= op= pat= ma=*

The *AM9513A_1 Timer Register(s) Access Test* option on the AM9513A Timer Tests Menu tests the control/data/address lines associated with the STC (AM9513A) device. It performs the test specified as the value of op= (default March Test) on the STC Interrupt Enable A Register.

STCS *pa=*

The *AM9513A_1 Timers and Status Test* option on the AM9513A Timer Tests Menu verifies functions of the STC device that may be used by applications. The test performs the following steps:

1. Initializes the STC device to a known state.
2. Clears the TC Out toggle bit for all five counter groups.
3. Sets the Mode Register, Load Register, and Hold Register for all five counter groups.
4. Sets the Alarm Registers for timers 1 and 2.
5. Tests the TC Out toggle bits using the Set/Clear instructions.
6. Places the counter in timer mode and single-steps until the value in the counter exceeds the alarm value.
7. Saves the value in timer 1.
8. Reads the value in the Hold Register and compares it to the alarm value. If the values do not compare, an error message is displayed.
9. Reinitializes the STC and loads, arms, and verifies the status of all counters.

This test repeats for all five counters.

LTR *pa= op= pat= ma=*

The *LQM 9513A Timer Register(s) Access Test* option on the AM9513A Timer Tests tests the control/data/address lines associated with the LQM (Link Quality Monitor) AM9513A device. It performs the test specified as the value of op= (default March Test) on the LQM Interrupt Enable A Register.

LTS *pa=*

The *LQM 9513A Timers and Status Test* option on the AM9513A Timer Tests Menu performs the same test on the LQM that the AM9513A_1 Timers and Status Test, described earlier, performs on the STC.

LAF *pa= op= pat= ma=*

The *LQM-A 9513A Line/Function/Interrupt Test* option on the AM9513A Timer Tests Menu verifies that both of the following are true:

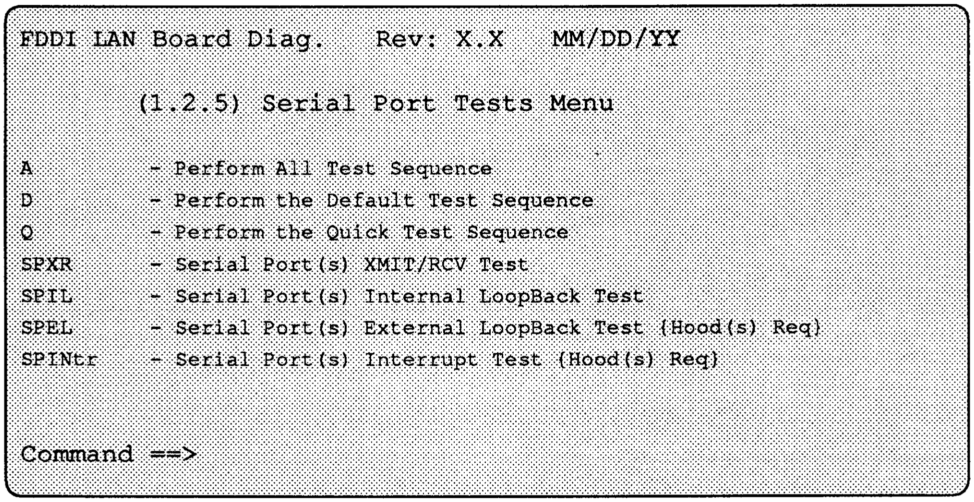
- The Enable LQM-A line from the FDDI Control Register is connected to the LQM STC 9513 device.
- The counters in the LQM STC work properly and are capable of generating a terminal count. The terminal count should generate a signal to the MFP 68901, which generates an interrupt to the Node Processor.

LBF *pa= op= pat= ma=*

The *LQM-B 9513A Line/Function/Interrupt Test* option on the AM9513A Timer Tests Menu performs the same test on LQM-B that the LQM-A 9513A Line/Function/Interrupt Test, just described, performs on LQM-A.

SPT

When you choose **SPT** from the Node Processor Glue Tests Menu, the Serial Port Tests Menu is displayed:



```
FDDI LAN Board Diag.   Rev: X.X   MM/DD/YY

         (1.2.5) Serial Port Tests Menu

A          - Perform All Test Sequence
D          - Perform the Default Test Sequence
Q          - Perform the Quick Test Sequence
SPXR       - Serial Port(s) XMIT/RCV Test
SPIL       - Serial Port(s) Internal LoopBack Test
SPEL       - Serial Port(s) External LoopBack Test (Hood(s) Req)
SPINtr     - Serial Port(s) Interrupt Test (Hood(s) Req)


Command ==>
```

The options on this menu allow you to verify the functioning of the serial port interface and its associated logic. For each test in this menu, the Node Processor is made idle so that it loops indefinitely on ROM instructions. At the end of each test, the Node Processor is reset to active state.

A *pa=*

The *Perform All Test Sequence* option on the Serial Port Tests Menu executes all of the tests on the menu in the order listed.

D *pa=*

The *Perform the Default Test Sequence* option on the Serial Port Tests Menu executes all of the tests on the menu, except the Serial Port(s) XMIT/RCV Test, in the order listed.

Q *pa=*

The *Perform the Quick Test Sequence* option on the Serial Port Tests Menu executes the following tests on the menu:

1. Serial Port(s) Internal LoopBack Test
2. Serial Port(s) External LoopBack Test

SPXR *pa=*

The *Serial Port(s) XMIT/RCV Test* option on the Serial Port Tests Menu performs a read loop on the serial port of the controller. It begins by displaying an A on the screen of the terminal attached to the serial port of the controller board. You can type other characters to be displayed, for visual verification of the serial port interface. To end the test, press ESC.

SPIL *pa=*

The *Serial Port(s) Internal LoopBack Test* option on the Serial Port Tests Menu initializes the serial port controller so that it transmits the values 0–255 at 300, 2400, and 9600 baud in internal loopback mode.

SPEL *pa=*

The *Serial Port(s) External LoopBack Test* option on the Serial Port Tests Menu initializes the serial port controller so that it transmits the values 0–255 at 300, 2400, and 9600 baud in pass-through mode.

NOTE *You must install an external loopback connector before executing this test.*

SPIN *pa=*

The *Serial Port(s) Interrupt Test* option on the Serial Port Tests Menu verifies that the Transmit Buffer Empty and Receive Buffer Full conditions function properly and that they can generate the proper interrupts. The interrupt line itself is also checked.

NOTE *You must install an external loopback connector before executing this test.*

The test performs the following steps:

1. Idles the Node Processor (controller) and initializes the serial port to 9600 baud.
2. Clears the region in controller memory used for reporting MFP interrupts.
3. Enables the serial port XMIT interrupt in the MFP and writes out the character X. This should cause an XMIT interrupt.
4. Verifies the interrupt condition and the setting of the proper bit.
5. Reinitializes the serial port controller and clears the special region reserved for receiving interrupt information.
6. Enables the serial port RCV interrupt in the MFP and writes out the character X. This should cause an RCV interrupt.
7. Verifies the interrupt condition and the setting of the proper bit.

FDDI Tests Menu

FDDI

When you choose *FDDI* from the FDDI Diagnostic Tests Menu, the FDDI Tests Menu is displayed:

```
FDDI LAN Board Diag.    Rev: X.X    MM/DD/YY

                (1.3) FDDI Tests Menu

A               - Perform All Test Sequence
D               - Perform the Default Test Sequence
Q               - Perform the Quick Test Sequence
FBTmenu         - AMD FDDI Basic Test Menu (1.3.1)
BSTmenu         - Buffer SRAM Tests Menu (1.3.2)
FFTmenu         - AMD FDDI Functional Test Menu (1.3.3)
FUTmenu         - FDDI Utilities Menu (1.3.4)


Command ==>
```

The options on this menu form the core of the FDDI Diagnostic. They allow you to narrow the test field by providing tests for specific devices in the FDDI subsystem.

Several of the submenu of the FDDI Tests Menu list the `ETrx` variable as the last option on the menu. The `ETrx` variable menu allows you to select a testing mode for all tests relating to ENDECs. You may select any of the following modes:

- WRAP A — Data is both transmitted from and received by ENDEC A.
- WRAP B — Data is both transmitted from and received by ENDEC B.
- THRU A — ENDEC A is the receiving ENDEC, and data is transmitted over ENDEC B.
- THRU B — ENDEC B is the receiving ENDEC, and data is transmitted over ENDEC A.
- Cycle — The test cycles through all available modes, repeating the test for each mode.

Once you make a selection, any test from any menu in the diagnostic that is associated with ENDECs is performed only on the ENDEC you specified. This makes it possible for you to isolate an ENDEC for testing. To display the current ENDEC mode, type **?** as the value for this variable. The other possible values for this variable are shown in the following table:

Value	*Meaning*
A	WRAP A
B	WRAP B
T	THRU A
O	THRU B

Value	*Meaning*
C	Cycle Mode (default)

A *pa= op= pat= ma=*

The *Perform All Test Sequence* option on the FDDI Tests Menu executes the All Test Sequence in each of the submenus in the numerical order shown on this menu.

D *pa= op= pat= ma=*

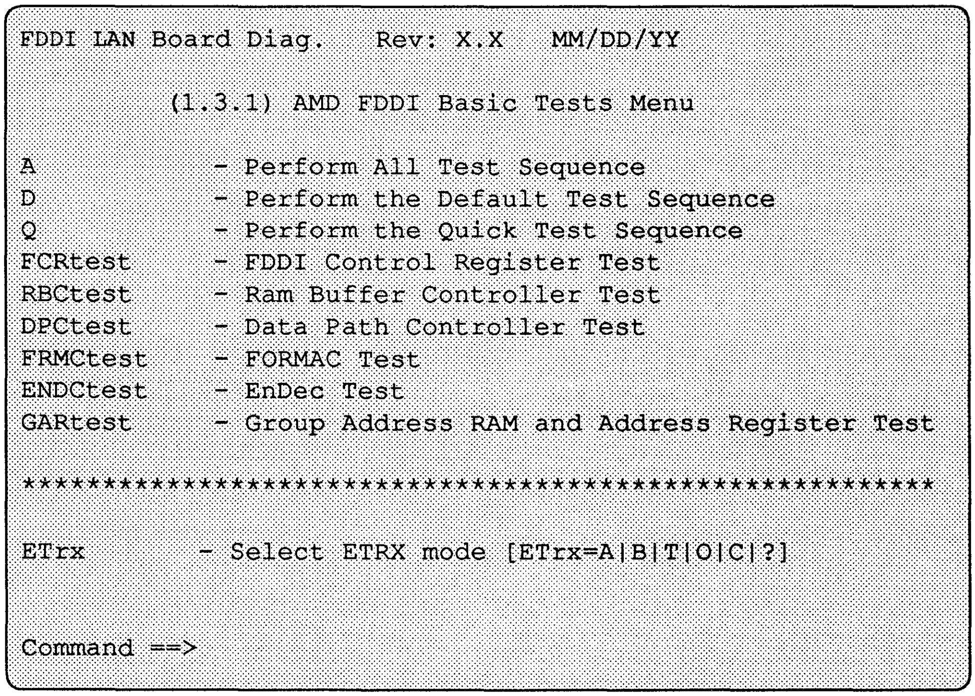
The *Perform the Default Test Sequence* option on the FDDI Tests Menu executes the Default Test Sequence in each of the submenus in the numerical order shown on this menu.

Q *pa= op= pat= ma=*

The *Perform the Quick Test Sequence* option on the FDDI Tests Menu executes the Quick Test Sequence in each of the submenus in the numerical order shown on this menu.

FBT

When you choose **FBT** from the FDDITests Menu, the AMD FDDI Basic Tests Menu is displayed:



```
FDDI LAN Board Diag.   Rev: X.X   MM/DD/YY

         (1.3.1) AMD FDDI Basic Tests Menu

A            - Perform All Test Sequence
D            - Perform the Default Test Sequence
Q            - Perform the Quick Test Sequence
FCRtest      - FDDI Control Register Test
RBCtest      - Ram Buffer Controller Test
DPCtest      - Data Path Controller Test
FRMCtest     - FORMAC Test
ENDCtest     - EnDec Test
GARtest      - Group Address RAM and Address Register Test

*********************************************************

ETrx        - Select ETRX mode [ETrx=A|B|T|O|C|?]


Command ==>
```

The options on this menu allow you to verify the AMD FDDI basic subsystem.

The `ETrx` variable at the bottom of this menu allows you to select a testing mode for all tests relating to ENDECs. For a complete description of this variable and its options, see the section entitled "FDDI Tests Menu," earlier in this chapter.

A *pa= op= pat= ma=*

The *Perform All Test Sequence* option on the AMD FDDI Basic Tests Menu executes all of the tests on the menu in the order listed.

D *pa= op= pat= ma=*

The *Perform the Default Test Sequence* option on the AMD FDDI Basic Tests Menu executes all of the tests on the menu in the order listed.

Q *pa= op= pat= ma=*

The *Perform the Quick Test Sequence* option on the AMD FDDI Basic Tests Menu executes the following tests:

1. RAM Buffer Controller Test
2. Data Path Controller Test

FCR *pa= op= pat=*

The *FDDI Control Register Test* option on the AMD FDDI Basic Tests Menu tests the control/data/address lines associated with the FDDI Control Register. This register controls the reset state and parity for the FDDI devices.

RBC *pa=*

The *RAM Buffer Controller Test* option on the AMD FDDI Basic Tests Menu tests the functionality of the RAM Buffer Controller, as well as the control/data/address lines associated with the controller. The test performs these steps:

1. Performs a hard reset on the RBC through the FDDI Control Register.
2. Verifies that the reset executed correctly.
3. Initializes the RBC by running the RBC initialization routine. This routine performs a hardware and software reset and initializes the RBC to a known state.
4. Verifies the data written to the RBC by the initialization routine.
5. Performs a March Test on the RBC MAR (Memory Address Register) to verify the data lines.

DPC *pa=*

The *Data Path Controller Test* option on the AMD FDDI Basic Tests Menu tests the functionality of the Data Path Controller, as well as the control/data/address lines associated with the controller. The test performs the following steps:

1. Performs a hard reset on the DPC through the FDDI Control Register.
2. Verifies that the reset executed correctly.
3. Initializes the DPC by running the DPC initialization routine. This routine performs a hardware and software reset and initializes the DPC to a known state.

4. Verifies the data written to the DPC by the initialization routine.
5. Performs a March Test on the DPC Length Counter Register to verify the data lines.

FRMC *pa=*

The *FORMAC Test* option on the AMD FDDI Basic Tests Menu tests the functionality of the FORMAC (Fiber Optic Ring Media Access Controller), as well as the associated control/data/address lines. The test performs these steps:

1. Performs a hard reset on the FORMAC through the FDDI Control Register.
2. Verifies that the reset executed correctly.
3. Initializes the FORMAC by running the FORMAC initialization routine. This routine performs a hardware and software reset and initializes the FORMAC to a known state.
4. Verifies the data written to the FORMAC by the initialization routine.
5. Performs a March Test on the FORMAC Tmax (maximum network token rotation time) Register to verify the data lines.

ENDC *pa=*

The *EnDec Test* option on the AMD FDDI Basic Tests Menu tests the functionality of the ENDEC, as well as the associated control/data/address lines. The ENDEC tested depends upon the value of the `ETrx` variable. The test performs these steps:

1. Performs a hard reset on the ENDEC through the FDDI Control Register.
2. Verifies that the reset executed correctly.
3. Initializes the ENDEC by running the ENDEC initialization routine. This routine performs a hardware and software reset and initializes the ENDEC to a known state.
4. Verifies the data written to the ENDEC by the initialization routine.
5. Performs a March Test on the ENDEC Tmax Register to verify the data lines.

GAR *pa=*

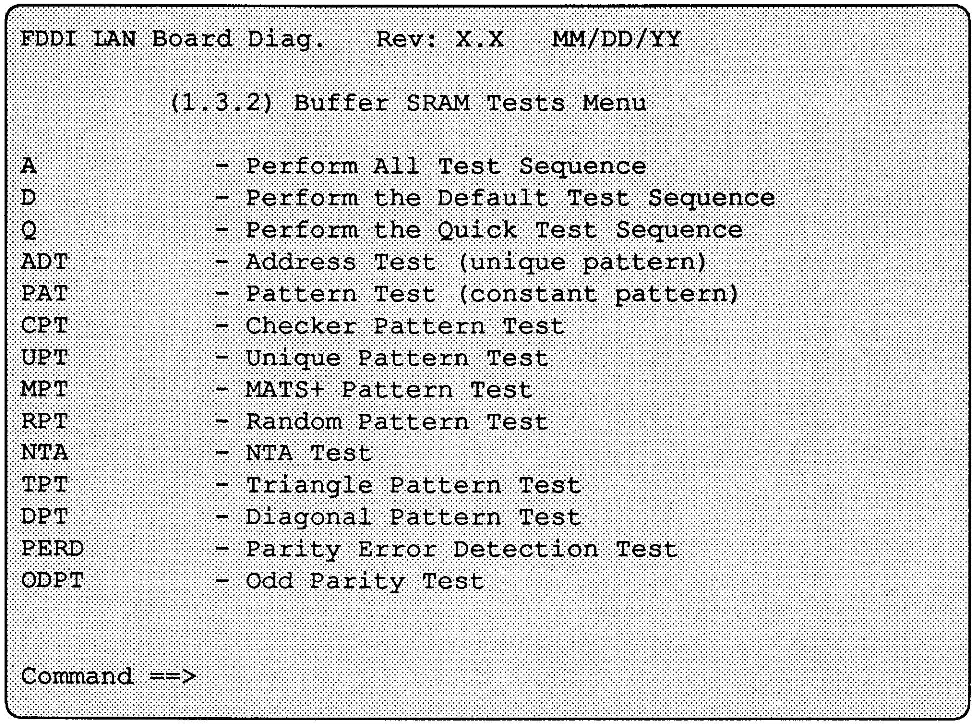
The *Group Address RAM and Address Register Test* option on the AMD FDDI Basic Tests Menu is a two-part test. The first part tests the Address Pointer Register, and the second part tests the Group Address RAM.

To test the Address Pointer Register, a write-only register, the test performs a write access to it, to verify that the register is accessible. Because the register is write-only, any further testing must be performed with a scope.

The second part of the test initializes the entire Group Address RAM to zero and verifies the initialization. It then fills the GAR with ones. Finally a pattern of alternating zeros and ones is written to the entire GAR and verified.

BST

When you choose **BST** from the FDDITests Menu, the Buffer SRAM Tests Menu is displayed:



```
FDDI LAN Board Diag.   Rev: X.X   MM/DD/YY

         (1.3.2) Buffer SRAM Tests Menu

A            - Perform All Test Sequence
D            - Perform the Default Test Sequence
Q            - Perform the Quick Test Sequence
ADT          - Address Test (unique pattern)
PAT          - Pattern Test (constant pattern)
CPT          - Checker Pattern Test
UPT          - Unique Pattern Test
MPT          - MATS+ Pattern Test
RPT          - Random Pattern Test
NTA          - NTA Test
TPT          - Triangle Pattern Test
DPT          - Diagonal Pattern Test
PERD         - Parity Error Detection Test
ODPT         - Odd Parity Test


Command ==>
```

The options on this menu allow you to verify the functionality of the Buffer SRAM. The default values of sa= and ea= are the start and end of Buffer RAM, respectively. The default value of d= is long word.

A *pa= pat= ma= sa= ea= d=*

The *Perform All Test Sequence* option on the Buffer SRAM Tests Menu executes all of the tests on the menu in the order listed.

D *pa= pat= ma= sa= ea= d=*

The *Perform the Default Test Sequence* option on the Buffer SRAM Tests Menu executes all of the tests on the menu in the order listed.

Q *pa= pat= ma= sa= ea= d=*

The *Perform the Quick Test Sequence* option on the Buffer SRAM Tests Menu executes the following tests in order:

1. Unique Pattern Test
2. NTA Test
3. Parity Error Detection Test
4. Odd Parity Test

The following options on this menu execute the specified standard tests, as described in the section entitled "General Tests," earlier in this chapter:

- □ Address Test (unique pattern)
- □ Pattern Test (constant pattern)
- □ Checker Pattern Test
- □ Unique Pattern Test
- □ MATS+ Pattern Test
- □ Random Pattern Test
- □ NTA Test
- □ Triangle Pattern Test
- □ Diagonal Pattern Test

The tests are performed with even parity and with parity checking disabled. You may specify the same parameters as those listed with the All, Default, and Quick Test Sequences.

PERD *pa=*

The *Parity Error Detection Test* option on the Buffer SRAM Tests Menu verifies the functionality of the parity error checking mechanism in the DPC. The test forces a parity error by performing these steps:

1. Initializes the RBC and DPC and enabling the parity error detection logic in the DPC.
2. Clears the region used for receiving interrupt information in the SRAM.
3. Sets the ODD/EVEN parity bit in the FDDI Control Register to EVEN and writes two long of words of data to the Buffer RAM.
4. Sets the ODD/EVEN parity bit in the FDDI Control Register to ODD and reads back the first long word. This action should cause the SNHPAR (Status NP/Host Parity Error) bit to be set in the RBC.
5. Checks the RBC and the interrupt reporting region for the proper values.

ODPT *pa= pat= ma= sa= ea= d=*

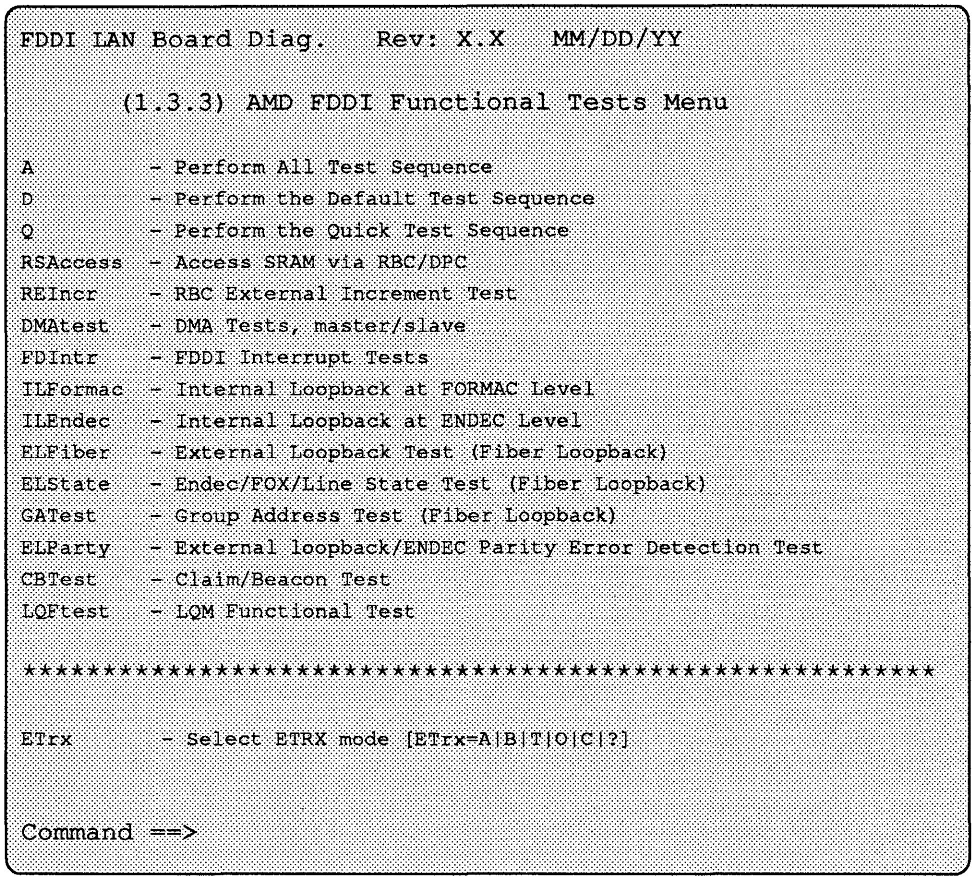
The *Odd Parity Test* option on the Buffer SRAM Tests Menu performs the following three tests in order with the DPC initialized to detect and report parity errors and the *ODD* parity bit set in the FDDI Control Register:

1. Address Test (unique pattern)
2. Unique Pattern Test
3. NTA Test

The default values of `sa=` and `ea=` are the start and end of Buffer RAM, respectively. The default value of `d=` is long word.

FFT

When you choose **FFT** from the FDDITests Menu, the AMD FDDI Functional Tests Menu is displayed:



```
FDDI LAN Board Diag.   Rev: X.X   MM/DD/YY

       (1.3.3) AMD FDDI Functional Tests Menu

A         - Perform All Test Sequence
D         - Perform the Default Test Sequence
Q         - Perform the Quick Test Sequence
RSAccess  - Access SRAM via RBC/DPC
REIncr    - RBC External Increment Test
DMAtest   - DMA Tests, master/slave
FDIntr    - FDDI Interrupt Tests
ILFormac  - Internal Loopback at FORMAC Level
ILEndec   - Internal Loopback at ENDEC Level
ELFiber   - External Loopback Test (Fiber Loopback)
ELState   - Endec/FOX/Line State Test (Fiber Loopback)
GATest    - Group Address Test (Fiber Loopback)
ELParty   - External loopback/ENDEC Parity Error Detection Test
CBTest    - Claim/Beacon Test
LQFtest   - LQM Functional Test

*******************************************************


ETrx       - Select ETRX mode [ETrx=A|B|T|O|C|?]


Command ==>
```

The options on this menu provide functional tests for comprehensive verification of the AMD FDDI subsystem.

The external loopback tests on this menu require a fiber loopback connector. You can configure the loopback connection in two different ways, depending upon the type of test you want to run:

- Connect the ENDEC XMIT line to its RECEIVE line. This configuration is useful for selective testing of individual ENDEC s.
- Connect theXMIT line of ENDEC A to the RECEIVE line of ENDEC B, and vice versa. This cross-connect configuration is required for the CROSS ENDEC external XMIT loop test. For details on this test, see the description of the External Loopback Test option in this section.

The `ETrx` variable at the bottom of this menu allows you to select a testing mode for all tests relating to ENDECs. For a complete description of this variable and its options, see the section entitled "FDDI Tests Menu," earlier in this chapter.

A *pa=*

The *Perform All Test Sequence* option on the AMD FDDI Functional Tests Menu executes all of the tests on the menu, in the order listed.

D *pa=*

The *Perform the Default Test Sequence* option on the AMD FDDI Functional Tests Menu executes all of the tests on the menu, in the order listed.

Q *pa=*

The *Perform the Quick Test Sequence* option on the AMD FDDI Functional Tests Menu executes the following tests in order:

1. Access SRAM via RBC/DPC
2. DMA Tests
3. FDDI Interrupt Tests
4. Internal Loopback at ENDEC Level
5. External Loopback Test
6. Endec/FOX/Line State
7. Group Address Test
8. Claim/Beacon Test
9. LQM Functional Test

RSA *pa= pat= ma=*

The *Access SRAM via RBC/DPC* option on the AMD FDDI Functional Tests Menu tests access to the SRAM through the RBC SRAM interface registers. The test performs the following steps:

1. Initializes the RBC, DPC, and the special region in Shared RAM used for interrupt reporting by the controller.
2. Writes a long word containing a specific pattern to the Buffer RAM through the RBC MAR (Memory Address Register), the DPC DRU (Data Register Upper), the DPC DRL (Data Register Lower), with a write data instruction starting at location 0 in the Buffer RAM.
3. Repeats the write instruction, marching the pattern across 32 bits.
4. Resets the devices and clears the DPC, MDRU, and MDRL registers to zero.
5. Reads the data from the Buffer RAM through the same path and compares the pattern read to the expected data.

REI *pa=*

The *RBC External Increment Test* option on the AMD FDDI Functional Tests Menu verifies the function of the RBC External Increment Register. Following initialization of the RBC and the region in SRAM used to receive interrupt information, the test reads the RBC RPR (Read Pointer for Receive buffer). Next it writes to the RBC Increment Pointer Register, reads the incremented value in the RPR, and compares the pattern read to the expected data. The process is repeated for the RBC external WPX (Write Pointer for Xmit buffer) Increment, using a read operation on the RBC External Increment Pointer Register and the RBC read WPX pointer instruction.

DMA *pa=*

The *DMA Tests* option on the AMD FDDI Functional Tests Menu is divided into two parts. The DMA Slave test checks all functions of the DMA system except automatic DMA transfers. The DMA Master test checks the functionality of the DMA system when performing an automatic transfer. The DMA system functions only on the Buffer RAM.

The DMA Slave test performs these steps:

1. Initializes the support devices of the DMA system, including the RBC, MFP, and the special region in Shared RAM used for receiving interrupt information.

2. Invokes a function that initializes the DMA system for a manual transfer from host to Node Processor.

3. Writes a pattern to the DMA data pipe and confirms that the proper bits were set and interrupt generated.

4. Initiates a slave transfer from NP to host with 0 rotation.

5. Checks to verify that the data is correct and that the data rotation mechanism in the DMAC (Direct Memory Access Controller) is operating correctly.

6. Repeats the sequence for host-to-NP and NP-to-host transfers with rotations of 1, 2, and 3 bytes.

The DMA Master test performs these steps:

1. Maps a section of user DVMA space on the host.

2. Initializes all DMA support devices and initializes DMAC counters to Burst Mode and a Backoff value of 0, to allow the DMAC to operate without hindrance.

3. Initializes the DVMA space and sets the DMAC to transfer data from host to controller.

4. Sets the VME arbitration and interrupt levels in the VME Configuration Register.

5. Arms the DMAC.

6. Verifies the DMAC, as well as the data in the target region of the Buffer RAM.
7. Initializes the user DVMA space on the host and initiates a DMA transfer from the,controller's Buffer RAM to the host.
8. Verifies the DMAC operation and checks the data for consistency.
9. Deallocates the user DVMA space.

FDI *pa=*

The *FDDI Interrupt Tests* option on the AMD FDDI Functional Tests Menu verifies the ability of the FDDI devices to generate the proper interrupt and status information.

The test first initializes the RBC and performs a hard reset on the DPC and FORMAC. It then forces a Buffer Full or Empty Error and verifies the RBC NMI (Non-maskable Interrupt) status and proper interrupt generation. Next the DPC NMI, DPC MINTR (Maskable Interrupt) and the FORMAC MINTR are verified by initializing all FDDI devices (RBC, DPC, FORMAC, and two of the ENDECs), and setting the beacon packet descriptor to 0. Packets are moved into the Buffer RAM, and the FORMAC is brought online and forced to an idle state. By examining the DPC and FORMAC registers, the status and interrupt generating capability of the FORMAC are verified, as well as the capability of the DPC to generate the proper level of interrupt.

ILF *pa=*

The *Internal Loopback at FORMAC Level* option on the AMD FDDI Functional Tests Menu verifies the ability of the FDDI devices to function together to transmit a packet correctly.

Following initialization of the FDDI devices, canned packets are built and placed in the appropriate areas in Buffer RAM, and the FORMAC is set to perform an internal loopback. The FORMAC is then forced to transmit a packet without waiting for a token. The test checks for display of the status missed packet and the setting of the proper bits in the Status Register.

ILE *pa=*

The *Internal Loopback at ENDEC Level* option on the AMD FDDI Functional Tests Menu performs a similar function to the previous test, extending the test path to the ENDECs. The ENDECs specified by the setting of the `ETrx` variable are initialized to operate in loopback mode, instead of the FORMAC, as in the previous test. The same test steps are performed.

ELF *pa=*

The *External Loopback Test* option on the AMD FDDI Functional Tests Menu extends the previous test to the FIBER level. All FDDI devices are tested, as well as the FIBER connection.

NOTE *This test requires a fiber loopback connector.*

Depending upon the setting of the `ETrx` variable, different tests are performed:

- □ If you specify **et=a**, the ENDEC A External XMIT Loopback Test is performed.
- □ If you specify **et=b**, the ENDEC B External XMIT Loopback Test is performed.
- □ If you specify **et=c**, in addition to the ENDEC A and ENDEC B External XMIT Loopback Tests, the ENDEC A XMIT ENDEC B Receive and ENDEC B XMIT ENDEC A Receive Tests are also performed. The latter two tests require a cross loopback connector (XMIT on A connected to RCV on B and vice versa).

ELS *pa=*

The *Endec/FOX/Line State Test* option on the AMD FDDI Functional Tests Menu verifies the functionality of the Line State FIFO and transition detection logic.

NOTE *This test requires a fiber loopback connector.*

Following device initialization and setup, an ENDEC is selected, based upon the setting of the `ETrx` variable. The ENDEC is set to force an IDLE stream, then a stream of QUIETs, and another IDLE stream. The line state FIFO is checked to see that it contains proper data and that the transition was recognized. Interrupt information is examined to verify that the transition caused the proper interrupt. IDLE/HALT and IDLE/ACTIVE transitions are also tested. If the proper transitions are not detected, an error message is displayed.

GAT *pa=*

The *Group Address Test* option on the AMD FDDI Functional Tests Menu performs a general test of the group address recognition logic.

NOTE *This test requires a fiber loopback connector.*

The test executes the following steps:

1. Initializes the FDDI devices.
2. Changes the destination address of the Sync packet to 0x80, to cause the group address bit to be set.
3. Places data packets in Buffer RAM.
4. Initializes the Group Address RAM to zero, selects the FORMAC and appropriate ENDEC, and transmits the Sync packet.
5. Checks the status address detect and missed packet bits. If the bits are set, the test fails and an error message is displayed.
6. Reinitializes the FDDI devices and fills Group Address RAM with ones.
7. Transmits the Sync packet again and checks for proper status of the status address detect and missed packet bits.

ELP *pa=*

The *External Loopback/ENDEC Parity Error Detection Test* option on the AMD FDDI Functional Tests Menu tests the ability of the ENDECs to recognize parity errors and generate the correct signals.

NOTE *This test requires a fiber loopback connector.*

The test performs the following steps:

1. Initializes the FDDI devices and sets the ODD/EVEN parity bit in the FDDI Control Register to EVEN. (Packet parity must be ODD.)
2. Turns off parity detection in the DPC, writes the packets to Buffer RAM, initializes the MFP, and clears the region in SRAM used for processing of interrupts.
3. Enables the ENDEC parity error interrupt bit in the MFP and transmits the Sync packet. This should cause the ENDEC to detect a parity error, and a parity error interrupt should be generated.
4. If a parity error interrupt is not generated, checks the Interrupt Pending Register for the proper bit, to help to isolate the cause of the problem.

CBT *pa=*

The *Claim/Beacon Test* option on the AMD FDDI Functional Tests Menu uses an internal loopback at the ENDEC level to test the combined functionality of the FDDI devices in the Claim/Beacon process. The test performs the following steps:

1. Initializes the FDDI devices.
2. Sets the Claim packet information field to 0xFFFFFFFF and writes packets to Buffer RAM.
3. Selects the ENDEC and places the FORMAC online. The FORMAC should enter the Claim state and then the Beacon state. If it does not, an error message is displayed.

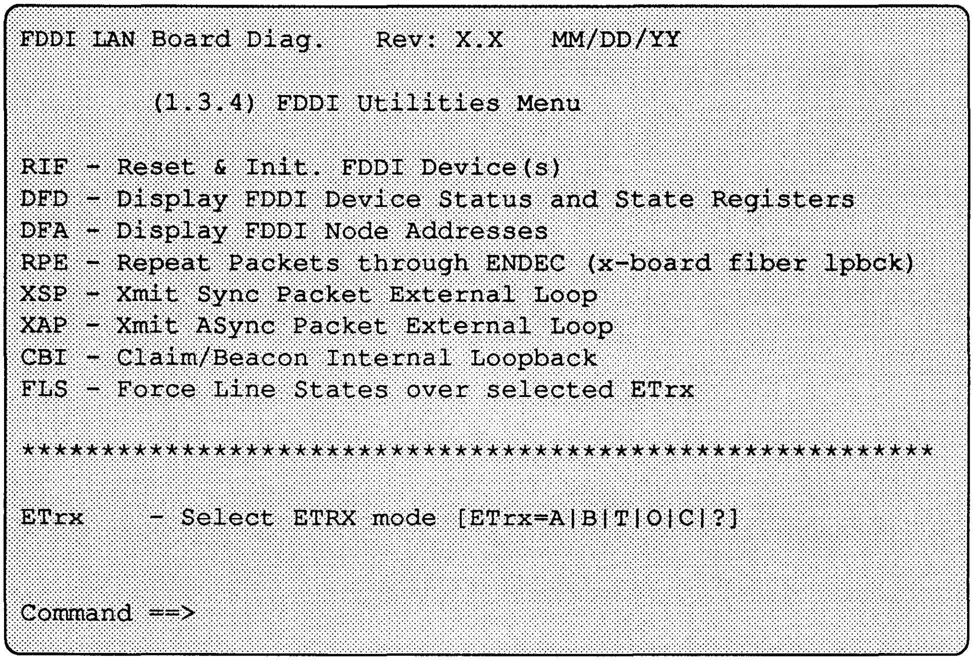
LQF *pa=*

The *LQM Functional Test* option on the AMD FDDI Functional Tests Menu verifies the error generation logic of LQM-A and LQM-B.

The test places the FDDI devices in the Claim/Beacon state. Next QUIET line states are repeatedly forced on the transmitting ENDEC. The receiving ENDEC should encounter illegal symbols which should be detected by the LQM error detection logic, resulting in an interrupt to the Node Processor. Although the LQM circuit itself is not accessible, this test verifies that it is at least partially functional.

FUT

When you choose **FUT** from the FDDI Tests Menu, the FDDI Utilities Menu is displayed:



```
FDDI LAN Board Diag.    Rev: X.X    MM/DD/YY

        (1.3.4) FDDI Utilities Menu

RIF - Reset & Init. FDDI Device(s)
DFD - Display FDDI Device Status and State Registers
DFA - Display FDDI Node Addresses
RPE - Repeat Packets through ENDEC (x-board fiber lpbck)
XSP - Xmit Sync Packet External Loop
XAP - Xmit ASync Packet External Loop
CBI - Claim/Beacon Internal Loopback
FLS - Force Line States over selected ETrx

*********************************************************

ETrx     - Select ETRX mode [ETrx=A|B|T|O|C|?]


Command ==>
```

The options on this menu provide utilities that you can use to perform common functions associated with the FDDI devices and their operation.

The `ETrx` variable at the bottom of this menu allows you to select a testing mode for all tests relating to ENDECs. For a complete description of this variable and its options, see the section entitled "FDDI Tests Menu," earlier in this chapter.

RIF *dev=*

The *Reset and Initialize FDDI Device(s)* option on the FDDI Utilities Menu allows you to reset and initialize a specific FDDI device or all FDDI devices, both hardware and software. The device to be reset and initialized is specified by the setting of the `dev=` parameter. The possible values of this parameter include the following:

```
RBC
DPC
FORMAC
ETRXA
ETRXB
ALL
```

The default is `ALL`.

DFD

The *Display FDDI Device Status and State Registers* option on the FDDI Utilities Menu displays the contents of all FDDI device registers and packet information used in testing.

DFA

The *Display FDDI Node Addresses* option on the FDDI Utilities Menu displays the current FDDI address being used by the FDDI Diagnostic tests. A sample display resulting from this option follows:

```
----------------------------------------------------------
             FDDI Node Addresses
----------------------------------------------------------
SAID = 0x59e9   LAID = 0x0000110359e9   SAGP = 0x59e9
----------------------------------------------------------
Please Press the 'Return' Key to Continue.
```

LAID represents the Long Address ID, SAID represents the Short Address ID, and SAGP represents the Short Group Address.

RPE

The *Repeat Packets Through ENDEC* option on the FDDI Utilities Menu allows you to connect an additional FDDI board, through fiber, to the board under test. You can then use the additional board instead of a fiber loopback connector. The additional board must be functional and must have passed the External Loopback Test on the AMD FDDI Functional Tests Menu.

XSP

The *XMIT Sync Packet External Loop* option on the FDDI Utilities Menu performs a sequence of the following functions:

1. Sets up the Sync packet with the loop bit enabled.
2. Initializes all FDDI devices and transfers packets into the Buffer RAM.
3. Transmits the Sync packet over the fiber.
4. Displays the contents of the FDDI registers.

XAP

The *XMIT ASync Packet External Loop* option on the FDDI Utilities Menu performs a sequence of the following functions:

1. Sets up the ASync packet with the loop bit enabled.
2. Initializes all FDDI devices and transfers packets into the Buffer RAM.
3. Transmits the ASync packet over the fiber.
4. Displays the contents of the FDDI registers.

`CBI`

The *Claim/Beacon Internal Loopback* option on the FDDI Utilities Menu allows you to change information in the Claim and Beacon packets while a test is executing the Claim/Beacon stages in the FORMAC. It also allows you to perform only steps associated with initialization of devices and entering Claim/Beacon mode.

`FLS`

The *Force Line States Over Selected ETrx* option on the FDDI Utilities Menu allows you to force line states through the ENDEC specified by the setting of the `ETrx` variable. The specified ENDEC is connected to the fiber, and the line state that you select is forced until you select a different line state or terminate the option by pressing [ESC]. When you terminate the option, the FDDI devices are reset.

28.8. Recommended Testing Sequence

The FDDI Diagnostic can be used most effectively as a "crawl out" sequence of tests. This means that testing should begin with a group of core devices and, as the functionality of those devices is verified, move on to other devices that are dependent on the core set. The All Test Sequence options in the diagnostic menus perform the tests in this order. The following list provides an overview of the test dependencies, so that you can follow the proper sequence when running individual tests. For easy reference, tests are numbered on the menus also, so that by following the numerical order, you can run tests in the most efficient sequence. For example, tests in menu 1.1 should be invoked before tests in menu 1.2, and so on.

Recommended Testing Sequence
VME Interface Tests Menu
VME Switch Register Read Only Loop
VME Configuration Register Loop
VME Interrupt Vector Register Loop
DMA Control and Status Register Loop
DMA 16 Bit Transfer Counter
DMA 32 Bit ADDRESS Counter
DMA Data Switch Registers Test
NP Glue Tests Menu
Shared DRAM Tests Menu
Tests in any order
Other Memory Devices Tests Menu
EEPROM Read Access Loop

Recommended Testing Sequence
EEPROM Checksum/Data Test
EEPROM March Test
EPROM Read Access Loop {SE=Local Only}
EPROM Checksum/Data Test {SE=Local Only}
IDPROM Read Access Loop
IDPROM Checksum/Data Test
NP Control & Interrupt Tests Menu
System Enable Register Access Loop
Diagnostic Register Access Loop
NP Bus Error/Interrupt Test
M68901 'MFP1' Access Loop
M68901 'MFP1' Timer/Interrupt Test
Host-->NP Command/Status Interrupt Test
NP-->Host Command/Status Interrupt Test
AM9513A Timer Test Menu
AM9513A Timer Register(s) Access Test
AM9513A Timers and Status Test (all 5 timers)
LQM 9513A Timer Register(s) Access Test
LQM 9513A Timers and Status Test (all 5 timers)
LQM-A 9513A Line/Function/Interrupt Test
LQM-B 9513A Line/Function/Interrupt Test
Serial Port Tests Menu
Serial Port(s) XMIT/RCV Test
Serial Port(s) Internal LoopBack Test
Serial Port(s) EXternal LoopBack Test {Hood(s) Req}
Serial Port(s) Interrupt Test {Hood(s) Req}
FDDI Tests Menu
AMD FDDI Basic Tests Menu
FDDI Control Register Tests
RAM Buffer Controller Tests
Data Path Controller Tests
FORMAC Tests

Recommended Testing Sequence
EnDec Tests
Group Address RAM and Address Register Test
Buffer SRAM Tests Menu
First nine tests in any order
Parity Error Detection Test
Odd Parity Test
AMD FDDI Functional Tests Menu
Access SRAM via RBC/DPC
RBC External Increment Test
DMA Tests, master/slave
FDDI Interrupt Tests
Internal Loopback at FORMAC Level
Internal Loopback at ENDEC Level
External Loopback Test (Fiber Loopback)
Endec/FOX/Line State Test (Fiber Loopback)
Group Address Test (Fiber Loopback)
External Loopback/ENDEC Parity Error Detection Test
Claim/Beacon Test

28.9. Messages

This section lists the messages which may be displayed by the FDDI Diagnostic, together with their interpretations. Informational messages provide general information about the test or utility being executed. Status messages report whether a test passes or fails. Error messages identify the errors encountered during testing.

```
**** EDF Cycle State Machine Out of Phase, PROGRAM ERROR, s=nn ****
```

This message is displayed in case of an internal software error.

```
*** 'xxxx' FAILED,  Cannot Map Board ****
```

The FDDI board could not be mapped (VME mapping problem). The `xxxx` represents the function name in which the failure occurred.

```
**** Cannot UnMap 'xxxx', Call To 'xxxx' FAILED ****
```

The board could not be unmapped.

```
**** 'xxxx' has not been mapped in. ***
```

An attempt was made to set Device Under Test (DUT) without mapping the named device first. Use the Map option to map the board first.

```
**** Could not clear trap handler using 'xxxx' for : 'xxxx' ***
```

The call to clear a specific trap handler failed (software problem).

```
**** 'xxxx' FAILED,  Cannot Map USER DVMA SPACE ****
```

An attempt to map in user DVMA space failed because of a problem beyond the control of the software (software or library problem).

```
**** Call to 'xxxx' FAILED, Cannot Unmap USER DVMA SPACE ****
```

An attempt to unmap user DVMA space failed because of a problem beyond the control of the software (software or library problem).

```
**** Cannot Execute Code In NP RAM ***
```

The host tried to execute code within the Node Processor RAM, and the attempt failed. Possible synchronization problem. Resetting the board and retrying the same option may be successful.

```
**** Can't Scopeloop in 'xxxx' Mode *****
```

Either the host or the Node Processor cannot access the specified portion of the address space on the Node Processor board. The mode may be either Host or Local.

```
**** Buserror at '0xnnnnn' During Download..ABORTED..***
```

A bus error was generated during an attempt to download code into NP RAM.

```
**** Cannot Download Firmware, FAILED ****
```

An attempt to download firmware to be executed in NP RAM failed.

```
*****  Fatal,  program FAILED access to its own data space ***
```

This is a software problem, possibly caused by a bad binary or library.

```
**** FAILURE : Cannot Access Firmware COM area on NP ****
```

An attempt to access the common communication area between host and NP in NP RAM failed. Possible causes may be incompatibility of the FDDI on-board selftest and the revision of the diagnostic, hardware failure, or improper VME switch settings.

```
*** Got SPURIOUS Interrupt During Test 'xxxx' ****
```

The board generated an unexpected SPURIOUS interrupt (possibly a serious hardware problem).

```
**** Cannot SET Environment Variable 'xxxx' ***
```

An attempt to set the environmental variable failed, possibly because of an internal software problem.

```
**** Unexpected Level 3 VME Vectored interrupt ****
```

The host received an unexpected Level 3 VME interrupt from the NP (possible hardware problem).

```
**** Time Out on Firmware Executing in NP RAM, BOARD FAILURE ****
```

This failure may have been caused by a synchronization problem or a hardware problem that forced the NP to hang. A software problem or a software race may have caused the failure as well. The downloaded firmware may have caused the NP to hang. Resetting the board and retrying may clear a synchronization problem.

```
**** PROBLEM : Please set the FDDI Board diag switch to DIAG position ***
```

The standalone diagnostics cannot execute properly if the NP board is not placed in the Extended Diagnostics Mode (DIAG mode). Set the DIAG switch on the NP to DIAG mode before continuing.

```
??? HW RESET RBC DID NOT WORK,  'xxxx' Did Not Reset ???
```

An attempt to reset the RBC through the FDDI Control Register failed. The xxxx represents the name of the internal register that did not reset.

```
??? External Increment 'xxxx' FAILED ???
```

An attempt to increment the RBC 'RPR' or 'WPX' registers through the External Increment registers failed.

```
??? Change in 'xxxx' status FAILED TO GENERATE a Level nn Interrupt ???
```

A specific interrupt on the NP at level nn was expected because of a change in the xxxxx device status. The problem may have been caused by a PAL or line connection problem between devices on board.

```
??? Change in 'xxxx' state FAILED TO SET proper bit in the 'xxxx'???
```

As a result of change in the state of a device, a bit should have been set in the xxxx device. The failure to set the bit may have been caused by a bad PAL or misconnected line.

```
??? Slave DMA FAILED: Byte Rot. Xfer TO NP=nn, Rcv From NP=nn ???
```

A Slave DMA transfer, transferring from NP and receiving from NP with the specified byte rotation of nn, failed.

```
??? nn Mstr DMA Host --> NP FAIL: Host= zzzzzz  NP= yyyyyy ???
```

The *n*th word in the Master DMA transfer from host to NP failed. The observed value on host and NP are displayed to show the inconsistency of data.

```
??? nn Mstr DMA NP --> Host FAIL: NP= zzzzzz  HOST= yyyyyy ???
```

The *n*th word in the Master DMA transfer from NP to host failed. The observed value on host and NP are displayed to show the inconsistency of data.

```
??? nn Mstr DMA HOST <--> NP FAIL: Exp= zzzzzz  Got= yyyyyy  ???
```

DMA transfer failed. This message states that the data was transferred from the host to NP and back to the host.

```
??? 'xxxx' 'yyyyy' NMI/status Test  FAILED ???
```

The Non-maskable Interrupt Test failed on device xxxx. The yyyy indicates the expected status which should have generated the interrupt.

```
??? 'xxxx' 'yyyyy' Maskable Intr./status Test  FAILED ???
```

The Maskable Interrupt Test failed on device xxxx. The yyyy indicates the expected status which should have generated the interrupt.

```
??? Internal LoopBack Through 'xxxx' FAILED.......... ???
```

An internal loopback though device xxxx failed.

```
??? External LoopBack Through 'xxxx' FAILED.......... ???
```

An internal loopback though device xxxx failed.

```
xxxx FIFO did not contain expected Sequence of Line States :
```

Either Line State A or Line State B did not contain the expected sequence of line states (possible PAL or ENDEC problem).

```
xxxx FIFO Didn't Flush (nn reads), Last Val=0x%nnn
```

An attempt was made to flush the line states FIFO, but the FIFO empty bit was not set after nn reads.

```
??? 'xxxx' 'yyyy' transition TEST FAILED ???
```

Line state xxxx to line state yyyy transition test failed. The transition from one line state to the next failed to generate the expected results (possible PAL, ENDEC, or interrupt logic problem).

```
??? FAILED, 'xxxx' entry was nn, address 'was/was not' recognized ???
```

This error message refers to the Group Address Test. When the xxxx location of the group address was set or cleared, the address was or was not recognized accordingly. When the entry is cleared, the group address should not be recognized; when it is set, the group address should be recognized by the FORMAC.

```
??? 'xxxx' Parity Error FAILED TO GENERATE a Level nn Interrupt ???
```

A parity error detected by the ENDEC should have generated a level n interrupt. The xxxx represents either ENDEC-A or ENDEC-B.

```
??? 'xxxx' Parity Error FAILED TO SET proper Bit the in 'yyyy' ???
```

A parity error detected by the ENDEC failed to set the appropriate bit in device yyyy. The xxxx represents either ENDEC-A or ENDEC-B.

```
??? 'xxxx' FAILED to go into 'yyyyyy' STATE,  mode = 'zzzz' ???
```

This general message indicates that device xxxx failed to enter the expected yyyyyy state in the zzzz mode. The four possible modes are WRAP A, WRAP B, THRU A and THRU B.

```
??? Major PROBLEM : Abort Testing,  Got unexpected 'xxxx' ..^Z or ^C to Abort ??
```

This interrupt is caused by the host CPU board or is the result of a software problem. It may not indicate any problem with the NP hardware.

```
#### ('xxxx') 'xxxx' BUSERR,  ADR= 0x%08x[0xnnn], Dat sz= nn bytes ####
```

An attempt to read or write from the NP board resulted in a bus error. A probable cause is incorrect setting of the VME switches on the NP board. Other causes include a synchronization problem between the NP and host software or bad VME interface logic on the NP board.

```
WARNING:  FORMAC Went into an Unexpected Beacon State.
```

This warning message may not be the result of a serious problem, but it does suggest a problem the FDDI chip set. The test will not fail as a result of this error.

```
Ring NOT UP. and NO Token Issued, Last FORMAC STAT=0xnnn, exp=0xnnn
```

The ring could not be established, and no token was issued by the FORMAC. The message shows the FORMAC status and the expected status. This may indicate a serious problem, and the error will cause the test to fail. The problem may be with the ENDECs, the fiber connection, the FORMAC itself, or PAL.

```
}}}} Line states samples indicate that 'xxxx' IS Unstable {{{{
```

This message indicates that ENDEC xxxx may be unstable.

```
???? Bad or Misconnected FIBER Detected, No Carrier on 'xxxx' ????
```

The message is the result of a bad ENDEC or a bad fiber connection. The xxxx represents the name of the receiving ENDEC.

```
??? 'xxxx' Initialization FAILED in 'yyyyy' Register verification ???
```

Register yyyy in device xxxx contained a different value than the one to which it had been initialized.

```
IP ??? Host CMD To NP FAILED ???
```

The host cannot access NP memory or the diagnostic is out of phase with NP selftest firmware. This error is usually caused by synchronization problems. Different versions of LLCP (Low Level Command Protocol) on host and NP could cause this problem. If no bus errors are generated when this message is displayed, the cause is probably a software problem.

```
NP/HOST Communication Area : BUS ERROR at '0xnnnnn'
```

The host's attempt to access the communication area address 0xnnnn on NP RAM resulted in a bus error.

```
PROBLEM : BAD News,  NP did Not Reset
```

The NP could not be reset by the host. The Magic NP RAM location which generates the reset signal is not functioning properly. The error could be caused by a synchronization problem or a PAL problem.

28.10. Glossary

Command Parameters	Parameters that alter the behavior of individual tests. Values for these parameters may be specified on the command line.
DMAC	Direct Memory Access Controller
DPC	Data Path Controller
DRL	Data Register Lower.
DRU	Data Register Upper.
ENDEC	Encoder/Decoder.
Environmental Variable	A variable that controls the environment of the diagnostic tests. Environmental variables can alter the behavior of the entire diagnostic.
FDDI	Fiber Distributed Data Interface (ASC X3T9).
FORMAC	Fiber Optic Ring Media Access Controller
FOX	Fiber Optics Transceivers.
Host	A mode of operation in the FDDI Diagnostic in which the tests are conducted by the host CPU, and all access to devices is done over the VME interface by the host CPU.
Local	A mode of operation in the FDDI Diagnostic in which the tests are conducted by the Node Processor (68020 processor on the FDDI board).
LLCP	Low Level Command Protocol
LQM	Link Quality Monitor
Magic	A particular group of addresses in the onboard RAM. Writing a value to one of these addresses generates a hardware signal that triggers a predefined event, such as generating an interrupt or resetting the interface.
MAR	Memory Address Register.

Node Processor

The FDDI intelligent controller board, containing an MC68020 microprocessor, which plugs into the VME bus.

NP

Node Processor.

RBC

RAM Buffer Controller.

RPR

Read Pointer for Receive buffer.

Tmax

Maximum token rotation time for the network.

VME

Internationally developed standard computer bus architecture specification. This specification defines an interfacing system used to connect data processing, data storage, and peripheral control devices in a closely coupled hardware configuration.

WPX

Write Pointer for Xmit buffer.

29

Sun-3/80 ESP SCSI Diagnostic

29

Sun-3/80 ESP SCSI Diagnostic

29.1. General Description

The ESP chip is an onboard SCSI (Small Computer System Interface) bus controller. All read and write operations performed on a disk or tape device connected to a Sun-3/80 are requested through the ESP chip. The ESP SCSI Diagnostic is designed to test the following devices attached to the Sun-3/80 SCSI bus:

- Internal and external hard disk drives
- External cartridge tape drives

The external disk and tape drives must be packaged in a Mass Storage Subsystem.

The ESP SCSI Diagnostic runs under the Exec and uses the same menu interface as the other Exec diagnostics.

29.2. What This Chapter Contains

Following an overview of the diagnostic and a list of required hardware and firmware, the SCSI Subsystem Diagnostics Main Menu and submenus are discussed. Menu selections and optional parameters are listed, along with brief descriptions. The end of the chapter contains a list of supported SCSI commands, a description of the diagnostic messages, and a glossary.

29.3. Overview of the Diagnostic

The ESP SCSI Diagnostic provides All, Default, and Quick test sequences, as well as individual tests. Parameters of the tests are given default values upon execution. Online help is provided. The diagnostic also generates and logs error messages for later retrieval.

The ESP SCSI Diagnostic tests data control and handling between two types of SCSI devices, as defined in *SCSI Standard Specifications*: an *Initiator* device and a *Target* device. An Initiator requests an operation to be performed by another SCSI device; a Target performs the operation requested by the Initiator. In this case, the onboard SCSI bus controller chip (ESP) is the Initiator, and the attached disk or tape drive is the Target.

SCSI Commands

Communication between Initiator and Target is implemented through sets of SCSI commands. Some of these commands are common; some are specific to individual devices. The data formats of certain commands differ, depending upon the vendor. SCSI commands involving data transfer are designed around a data structure consisting of a contiguous set of logical blocks of a fixed or explicitly defined data length. The SCSI device maps the physical characteristics of the internal storage medium into this logical block data structure. A single READ or WRITE command transfers one or more logical blocks of data.

The ESP SCSI Diagnostic is designed around the existence of four internal data buffers, each capable of holding four logical blocks. The data length of each logical block is 512 bytes. In addition, 14 large data buffers of 56 Kbytes each assist in large data transfers.

The diagnostic tests use common SCSI commands, and standard features of those commands, wherever possible. By minimizing the use of vendor-specific commands, it is possible to test a wide variety of disk and tape devices. Provision for identifying specific devices has been included, however, so that vendor-unique attributes can be accommodated. A list of the commands supported by this diagnostic for both internal embedded hard disk and external disk and tape drives is included under "SCSI Commands" later in this chapter.

Error Detection

Upon completion of a SCSI command, a Target returns a Status byte to the Initiator. The Status byte has the following form:

7	*6*	*5*	*4*	*3*	*2*	*1*	*0*	*Description*
R	V	V	0	0	0	0	V	GOOD
R	V	V	0	0	0	1	V	CHECK CONDITION
R	V	V	0	1	0	0	V	BUSY
R	V	V	1	0	0	0	V	INTERMEDIATE/GOOD
R	V	V	1	1	0	0	V	RESERVATION CONFLICT

V indicates a vendor-specific bit (usually 0), and R indicates a reserved bit (currently 0).

When an Initiator first attempts to connect with a Target, when certain conditions arise during the connection, and when the connection is terminated, a message system controls path management by reporting status. The ESP SCSI Diagnostic reports an error if it detects an error message code in this SCSI message system report. The message codes are shown in the following table:

Code	Type	Description	Direction	
0x00	M	Command Complete	In	
0x01	O	Extended Message		Out
0x02	O	Save Data Pointer	In	
0x03	O	Restore Pointers	In	
0x04	O	Disconnect	In	
0x05	O	Initiator-detected Error		Out
0x06	O	Abort		Out
0x07	O	Message Reject	In	Out
0x08	O	No Operation		Out
0x09	O	Message Parity Error		Out
0x0A	O	Linked Command Complete	In	
0x0B	O	Linked Command Complete (with Flag)	In	
0x0C	O	Bus Device Reset		Out
0x0D:0x7F	R	Reserved Codes		
0x80:0xFF	O	Identify	In	Out

M	Mandatory
O	Optional
In	Target-to-Initiator
Out	Initiator-to-Target

For a more complete description of these codes, see the *SCSI Standard Specification* and the manuals supplied with the individual Target devices.

29.4. Hardware Requirements

The following hardware is required to run the ESP SCSI Diagnostic:

- A Sun-3/80 CPU board with onboard ESP chip and minimum 4 Mbytes of RAM.
- A tested and operational internal embedded SCSI disk drive assigned as Drive 0. Other disk and tape drives may be attached to the SCSI bus as Drives 0–6. The onboard ESP chip is assigned to Drive 7.
- All hardware supporting the onboard ESP chip previously tested and operational.
- A monitor.
- A keyboard.
- A boot device (local disk, local tape, or remote disk over Ethernet).

29.5. User Interface

The user interface of the ESP SCSI Diagnostic adheres to the menu standards of the Exec. Each test may be selected from a menu by typing the letter or letters displayed in uppercase in the column on the left side of the menu.

A Main Menu and submenus are provided. The commands on the submenus provide for testing the general functions of internal and external disks and a tape drive. Some submenus include more than one screen. In these cases, you can choose the `More` option to move to the next menu and `Exit` to return to the prior menu. You can also move to the prior menu by pressing ESC then RETURN.

All and Quick test sequences are provided on the SCSI Subsystem Diagnostics Main Menu and submenus. A choice on the Main Menu allows you to reset the internal system to its default values and reconfigure the attached SCSI devices.

To display online Help for the current menu, press **?** when the menu is displayed.

29.6. Starting the Diagnostic

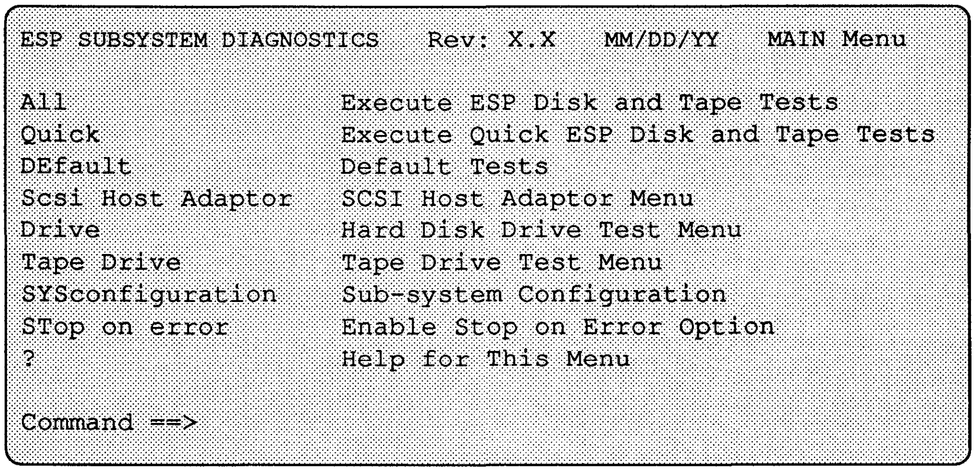
For information on starting the Exec, refer to Chapter 2, "Using the SunDiagnostic Executive." After you have started the Exec, choose the ESP SCSI Diagnostic from the Diagnostics Menu.

29.7. The Diagnostic Menus

This section of the chapter provides a modular description of the ESP SCSI Diagnostic, beginning with the Main Menu and working down through the options available on each of the submenus. A list of status and error messages is given in the section entitled *Messages*.

ESP Subsystem Diagnostics Main Menu

The ESP Subsystem Diagnostics Main Menu, which displays when you start the ESP SCSI Diagnostic, provides access to the submenus of the individual tests:



```
ESP SUBSYSTEM DIAGNOSTICS   Rev: X.X   MM/DD/YY   MAIN Menu

All                   Execute ESP Disk and Tape Tests
Quick                 Execute Quick ESP Disk and Tape Tests
DEfault               Default Tests
Scsi Host Adaptor     SCSI Host Adaptor Menu
Drive                 Hard Disk Drive Test Menu
Tape Drive            Tape Drive Test Menu
SYSconfiguration      Sub-system Configuration
STop on error         Enable Stop on Error Option
?                     Help for This Menu

Command ==>
```

Parameters

The parameters that control how the tests in the ESP SCSI Diagnostic tests execute can be set through the Parameter First Menu and Parameter Second Menu, for both disk and tape devices. You can display the current values of all parameters through the Parameter and Buffer Display Menu. Access to both of these menus is provided through the Hard Disk Drive Test Menu and Tape Drive Test Menu options on the Main Menu.

Execute ESP Disk and Tape Tests

A

The *Execute ESP Disk and Tape Tests* option on the ESP Subsystem Diagnostics Main Menu executes a sequence of all major disk and tape tests. The tests run until successful completion or until an error is encountered.

Execute Quick ESP Disk and Tape Tests

Q

The *Execute Quick ESP Disk and Tape Tests* option on the ESP Subsystem Diagnostics Main Menu runs a quick disk test, followed by a quick tape test, until successful completion or until an error is encountered. The quick disk test runs the Random Write/Read/Verify Test on all attached disk devices. The Quick Tape Test on all attached tape devices performs the following steps:

1. Initializes the Target tape drive.
2. Issues a REWIND command.
3. Writes four logical blocks of data using the default data pattern.
4. Writes four file marks.
5. Issues a REWIND.
6. Reads four logical blocks of data.
7. Compares the data in the respective write and read buffers.
8. Reads into a file mark and analyzes the reported status.
9. Spaces into a file mark and analyzes the reported status.

Default Tests

DE

The *Default Tests* option on the ESP Subsystem Diagnostics Main Menu is reserved for a future definition of a standard test for subsystem verification. This choice currently executes the Quick disk and tape tests.

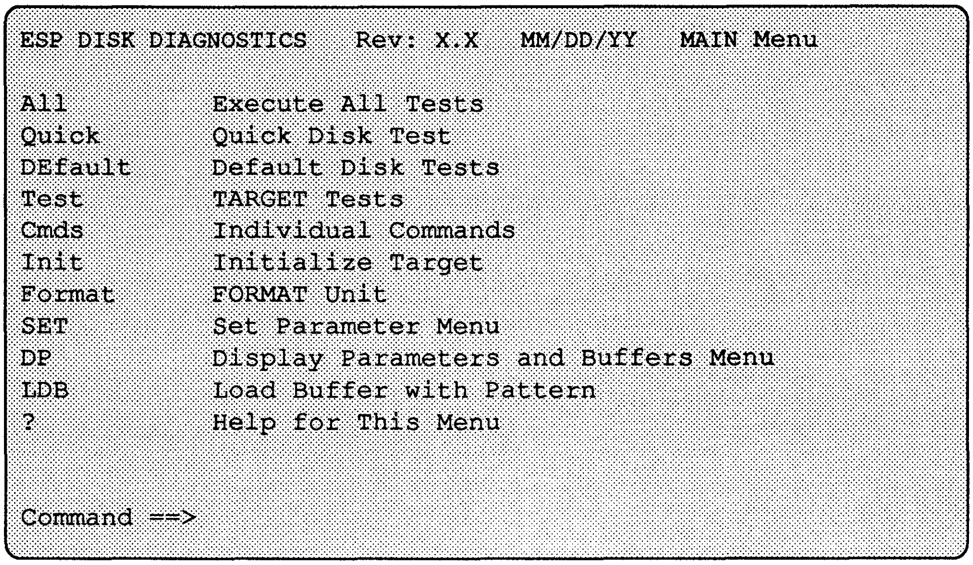
SCSI Host Adaptor Menu

This choice on the ESP Subsystem Diagnostics Main Menu is reserved for later use. It will be used for detailed ESP test mode verification.

Hard Disk Drive Test Menu

D

When you choose **D** from the ESP Subsystem Diagnostics Main Menu, the ESP Disk Diagnostics Main Menu is displayed:



```
ESP DISK DIAGNOSTICS   Rev: X.X   MM/DD/YY   MAIN Menu

All          Execute All Tests
Quick        Quick Disk Test
DEfault      Default Disk Tests
Test         TARGET Tests
Cmds         Individual Commands
Init         Initialize Target
Format       FORMAT Unit
SET          Set Parameter Menu
DP           Display Parameters and Buffers Menu
LDB          Load Buffer with Pattern
?            Help for This Menu


Command ==>
```

The options on the ESP Disk Diagnostics Main Menu provide access to additional submenus for comprehensive disk testing. You can select major tests and individual commands, as well as obtain status displays.

Execute All Tests

A

The *Execute All Tests* option on the ESP Disk Diagnostics Main Menu executes all of the disk tests, in the order listed on the test submenu.

Execute Quick Disk Test

Q

The *Execute Quick Disk Test* option on the ESP Disk Diagnostics Main Menu runs a quick disk test on all attached disk devices.

Default Disk Tests

DE

The *Default Disk Tests* option on the ESP Disk Diagnostics Main Menu executes the Disk base Test #1 (described later in this chapter under the ESP Disk target Diagnostics Second Menu) and the Quick Disk Test.

T

When you choose the *TARGET Tests* option on the ESP Disk Diagnostics Menu, the ESP Disk Target Diagnostics Test Menu is displayed. This menu allows you to select major disk drive tests. For a complete description of the options on this menu, see the section entitled "SCSI Disk Target Diagnostics Test Menu" later in this chapter.

C

When you choose the *Individual Commands* option on the SCSI Disk Diagnostics Menu, the Disk Individual Commands Main Menu is displayed. This menu allows you to issue commands to the default drive. For a complete description of the options on this menu, see the section entitled "Disk Individual Commands Main Menu" later in this chapter.

I

The *Initialize Target* option on the ESP Disk Diagnostics Menu reinitializes the default disk drive. If you want to change the default drive, choose **SET** to display the Parameter Menu, from which you can change the number of the Target drive.

F

The *FORMAT Unit* option on the ESP Disk Diagnostics Menu formats the default disk drive. If you want to change the default drive, choose **SET** to display the Parameter Menu, from which you can change the number of the Target drive.

SET

When you choose the *Set Parameter Menu* option on the SCSI Disk Diagnostics Menu, the Disk Parameter First Menu is displayed. The options on this menu allow you to specify settings that control how the ESP SCSI Diagnostic tests execute. An additional submenu allows you to display current parameter settings and the contents of internal buffers. For a complete discussion of the options on this menu, see the section entitled "Disk Parameter First Menu," later in this chapter.

DP

When you choose the *Display Parameters and Buffers Menu* option on the SCSI Disk Diagnostics Menu, the Parameter and Buffer Display Menu is displayed. The options on this menu allow you to display the program default parameters, as well as the contents of the four internal data buffers and the SCSI command input and output buffers. For a complete description of the options on this menu, see the section entitled "Parameter and Buffer Display Menu," later in this chapter.

LDB

The *Load Buffer with Pattern* option on the SCSI Disk Diagnostics Menu allows you to load a specific data pattern into the internal data buffer that you choose.

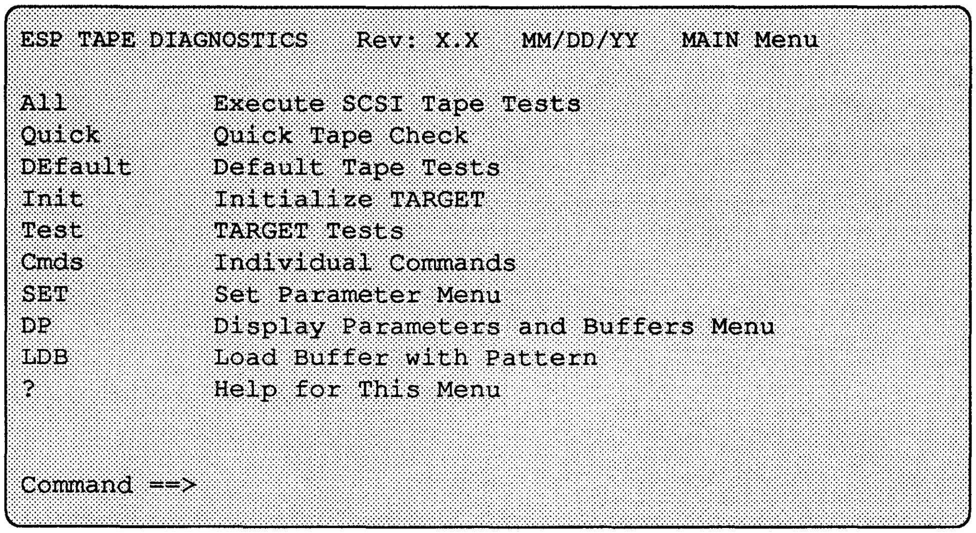
?

The *Help for This Menu* option on the SCSI Disk Diagnostics Menu displays a Help message that briefly describes the options on the current menu.

Tape Drive Test Menu

T

When you choose **T** from the Main Menu, the ESP Tape Diagnostics Main Menu is displayed:



```
ESP TAPE DIAGNOSTICS   Rev: X.X   MM/DD/YY   MAIN Menu

All          Execute SCSI Tape Tests
Quick        Quick Tape Check
DEfault      Default Tape Tests
Init         Initialize TARGET
Test         TARGET Tests
Cmds         Individual Commands
SET          Set Parameter Menu
DP           Display Parameters and Buffers Menu
LDB          Load Buffer with Pattern
?            Help for This Menu


Command ==>
```

The ESP Tape Diagnostics Main Menu provides access to the options available for tape device testing. All and Quick test sequences are provided, as well as additional submenus for comprehensive tape testing.

A

The *Execute SCSI Tape Tests* option on the ESP Tape Diagnostics Menu causes all of the major tape tests to execute sequentially on all attached tape devices.

Q

The *Quick Tape Check* option on the ESP Tape Diagnostics Menu causes the Quick Tape Test to execute on all attached tape devices. The steps in the Quick Tape Test are outlined in the section describing the ESP Subsystem Diagnostic Main Menu earlier in this chapter.

DE

The *Default Tape Tests* option on the ESP Tape Diagnostics Menu is reserved for later definition of a standard test for subsystem verification. This choice currently executes the Quick tape tests.

I

The *Initialize TARGET* option on the ESP Tape Diagnostics Menu reinitializes the default tape drive. If you want to change the default drive, choose **SET** to display the Tape Parameter First Menu, from which you can change the number of the Target drive.

T

The *TARGET Tests* option on the ESP Tape Diagnostics Menu is currently not implemented. This choice will provide for the selection of major tape drive tests.

C

When you choose the *Individual Commands* option on the ESP Tape Diagnostics Menu, the Tape Individual Commands Main Menu is displayed. This menu allows you to issue SCSI commands that support tape testing. For a complete description of the options on this menu, see the section entitled "Tape Individual Commands Main Menu" later in this chapter.

SET

When you choose the *Set Parameter Menu* option on the ESP Tape Diagnostics Menu, the Tape Parameter First Menu is displayed. The options on this menu allow you to specify settings that control how the ESP SCSI Diagnostic tests execute. An additional submenu allows you to display current parameter settings and the contents of internal buffers. For a complete discussion of the options on this menu, see the section entitled "Tape Parameter First Menu," later in this chapter.

DP

When you choose the *Display Parameters and Buffers Menu* option on the ESP Tape Diagnostics Menu, the Parameter and Buffer Display Menu is displayed. The options on this menu allow you to display the program default parameters, as well as the contents of the four internal data buffers and the SCSI command input and output buffers. For a complete description of the options on this menu, see the section entitled "Parameter and Buffer Display Menu," later in this chapter.

LDB

The *Load Buffer with Pattern* option on the ESP Tape Diagnostics Menu allows you to load a specific data pattern into the internal data buffer that you choose.

?

The *Help for This Menu* option on the ESP Tape Diagnostics Menu displays a Help message that briefly describes the options on the current menu.

Sub-system Configuration

SYS

The *Sub-system Configuration* option on the ESP Subsystem Diagnostics Main Menu reinitializes the internal system parameters and reconfigures all attached SCSI devices.

Stop on Error

ST

The *Enable Stop on Error* option on the ESP Subsystem Diagnostics Main Menu allows you to change the value of the local stop-on-error runtime parameter. The default setting causes all tests to stop when an error is encountered.

Help

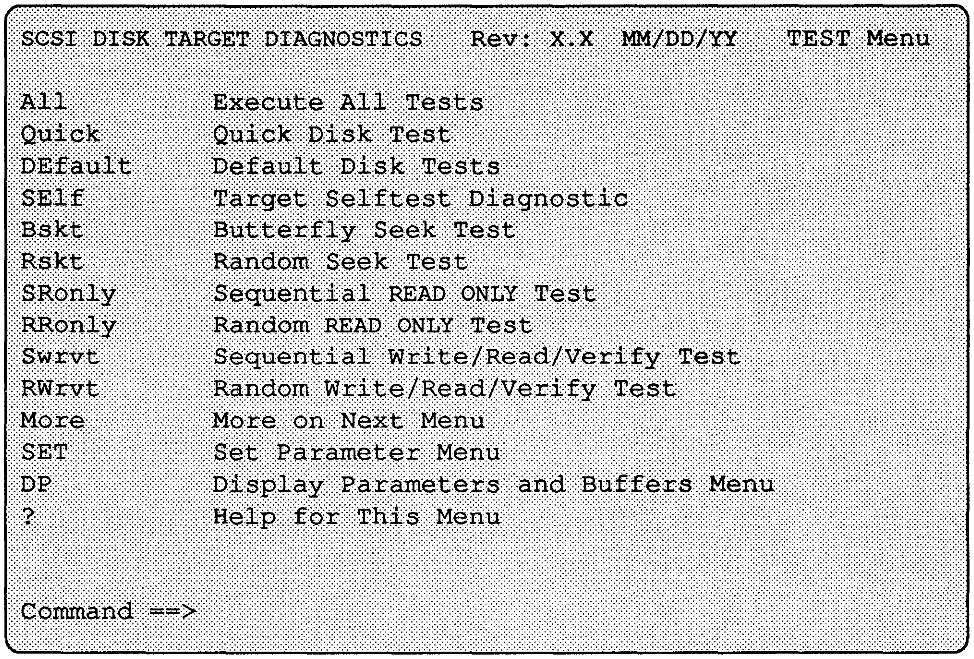
?

The *Help for Main Menu* option on the ESP Subsystem Diagnostics Main Menu displays a Help message that briefly describes the options on the Main Menu.

SCSI Disk Target Diagnostics Test Menu

T

When you choose **T** from the ESP Disk Diagnostics Menu, the SCSI Disk Target Diagnostics Test Menu is displayed:



```
SCSI DISK TARGET DIAGNOSTICS   Rev: X.X  MM/DD/YY   TEST Menu

All          Execute All Tests
Quick        Quick Disk Test
DEfault      Default Disk Tests
SElf         Target Selftest Diagnostic
Bskt         Butterfly Seek Test
Rskt         Random Seek Test
SRonly       Sequential READ ONLY Test
RRonly       Random READ ONLY Test
Swrvt        Sequential Write/Read/Verify Test
RWrvt        Random Write/Read/Verify Test
More         More on Next Menu
SET          Set Parameter Menu
DP           Display Parameters and Buffers Menu
?            Help for This Menu


Command ==>
```

The options on the SCSI Disk Target Diagnostics Test Menu provide access to the major disk tests.

A

The *Execute All Tests* option on the SCSI Disk Target Diagnostics Menu executes a sequence of all of the major disk tests on the menu on all attached disk devices.

Q

The *Quick Disk Test* option on the SCSI Disk Target Diagnostics Menu executes the Random Write/Read/Verify Test on all attached disk devices.

DE

The *Default Disk Tests* option on the SCSI Disk Target Diagnostics Menu executes the Random Write/Read/Verify Test and the Disk Base Test #1 on the ESP Disk Target Diagnostics Second Menu on all attached disk devices.

SE

The *Target Selftest Diagnostic* option on the SCSI Disk Target Diagnostics Menu issues the SEND DIAGNOSTIC command, which causes execution of the selftest on the default drive.

B

The *Butterfly Seek Test* option on the SCSI Disk Target Diagnostics Menu executes the Butterfly Seek Test on the default drive. This test uses two LBA (Logical Block Address) pointers, one initialized to 0, the other initialized to the maximum available LBA for the default drive. The test then seeks to a pointer and optionally sends a VERIFY command to the disk controller. Next it repeats these operations with the other pointer. If the operations complete successfully, the first pointer is incremented and the second decremented. The test completes when each pointer reaches the initial value of the other.

R

The *Random Seek Test* option on the SCSI Disk Target Diagnostics Menu executes the Random Seek Test on the default drive. This test generates a random LBA and issues a seek to that LBA. If the default verify flag is not 0, a VERIFY command is issued to the selected disk device. These operations are repeated the number of times specified by the random operation count (set on the Disk Parameter First Menu).

SR

The *Sequential Read-Only Test* option on the SCSI Disk Target Diagnostics Menu executes the Sequential Read-Only Test on the default drive. This test sequentially reads the LBAs from the disk, beginning at the default starting LBA and ending at the default ending LBA. The read operation is nondestructive. The test relies upon the ability of the disk controller to detect data errors.

RR

The *Random Read-Only Test* option on the SCSI Disk Target Diagnostics Menu executes the Random Read-Only Test on the default drive. This test generates a random LBA within a range bounded by the default starting LBA and default ending LBA. It then reads the LBA from the disk. The read operation is nondestructive. The test relies upon the ability of the disk controller to detect data errors. The process is repeated the number of times specified by the random operation count.

S

The *Sequential Write/Read/Verify Test* option on the SCSI Disk Target Diagnostics Menu executes the Sequential Write/Read/Verify Test on the default drive. This test sequentially writes a default data pattern on the disk at the current LBA, retrieves the disk-resident LBA into a separate internal data buffer, and then verifies the write and read operation data buffers. The test begins at the default starting LBA and ends at the default ending LBA.

`RW`

The *Random Write/Read/Verify Test* option on the SCSI Disk Target Diagnostics Menu executes the Random Write/Read/Verify Test on the default drive. This test generates a random LBA within a range bounded by the default starting LBA and default ending LBA. It then sequentially writes a default data pattern on the disk at the current LBA, retrieves the disk-resident LBA into a separate internal data buffer, and finally verifies the write and read operation data buffers. This process is repeated the number of times specified in the default random operation count.

`M`

The *More on Next Menu* option on the SCSI Disk Target Diagnostics Menu allows you to go to the second Disk Target Diagnostics Menu screen. For a complete discussion of the options on this menu, see the next section entitled "ESP Disk Target Diagnostics Second Menu."

`SET`

When you choose the *Set Parameter Menu* option on the SCSI Disk Target Diagnostics Menu, the Disk Parameter First Menu is displayed. The options on this menu allow you to specify settings that control how the ESP SCSI Diagnostic tests execute. An additional submenu allows you to display current parameter settings and the contents of internal buffers. For a complete discussion of the options on this menu, see the section entitled "Disk Parameter First Menu," later in this chapter.

`DP`

When you choose the *Display Parameters and Buffers Menu* option on the SCSI Disk Target Diagnostics Menu, the Parameter and Buffer Display Menu is displayed. The options on this menu allow you to display the program default parameters, as well as the contents of the four internal data buffers and the SCSI command input and output buffers. For a complete description of the options on this menu, see the section entitled "Parameter and Buffer Display Menu," later in this chapter.

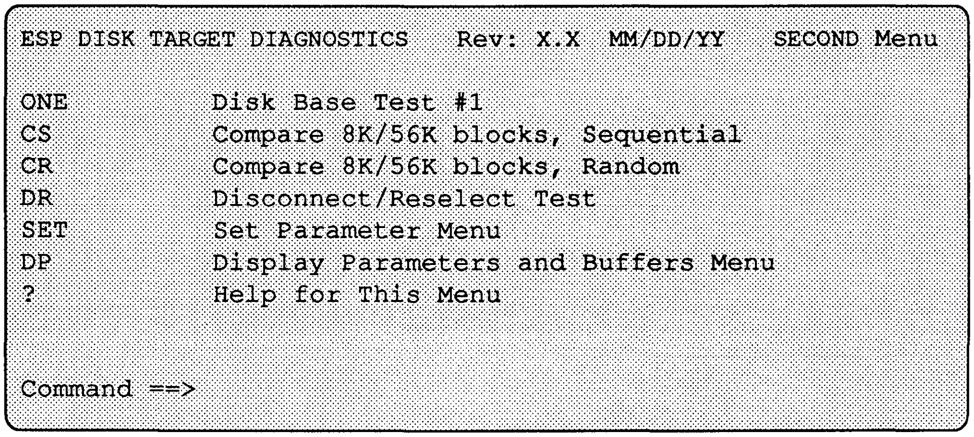
`?`

The *Help for This Menu* option on the SCSI Disk Target Diagnostics Menu displays a Help message that briefly describes the options on the current menu.

ESP Disk Target Diagnostics Second Menu

T

When you choose **T** from the SCSI Disk Target Diagnostics Test Menu, the ESP Disk Target Diagnostics Second Menu is displayed:



```
ESP DISK TARGET DIAGNOSTICS   Rev: X.X  MM/DD/YY   SECOND Menu

ONE          Disk Base Test #1
CS           Compare 8K/56K blocks, Sequential
CR           Compare 8K/56K blocks, Random
DR           Disconnect/Reselect Test
SET          Set Parameter Menu
DP           Display Parameters and Buffers Menu
?            Help for This Menu


Command ==>
```

The last three options on this menu perform the same functions as the corresponding options on the SCSI Disk Target Diagnostics Test Menu, just described. The first four options on this menu are described next.

ONE

The *Disk Base Test #1* option on the ESP Disk Target Diagnostics Second Menu reads and verifies every LBA on the disk by using the VERIFY command.

CS

The *Compare 8K/56K blocks, Sequential* option on the ESP Disk Target Diagnostics Second Menu reads in one large block to fill a data buffer. It then reads in the same data one logical block (512 Kbytes) at a time to fill a second buffer. The contents of the buffers are then compared. If the contents of the buffers are identical, the next sequential logical block is selected and the process is repeated.

CR

The *Compare 8K/56K blocks, Random* option on the ESP Disk Target Diagnostics Second Menu reads in one large block to fill a data buffer. It then reads in the same data one logical block (512 Kbytes) at a time to fill a second buffer. The contents of the buffers are then compared. If the contents of the buffers are identical, the next logical block is selected randomly and the process is repeated.

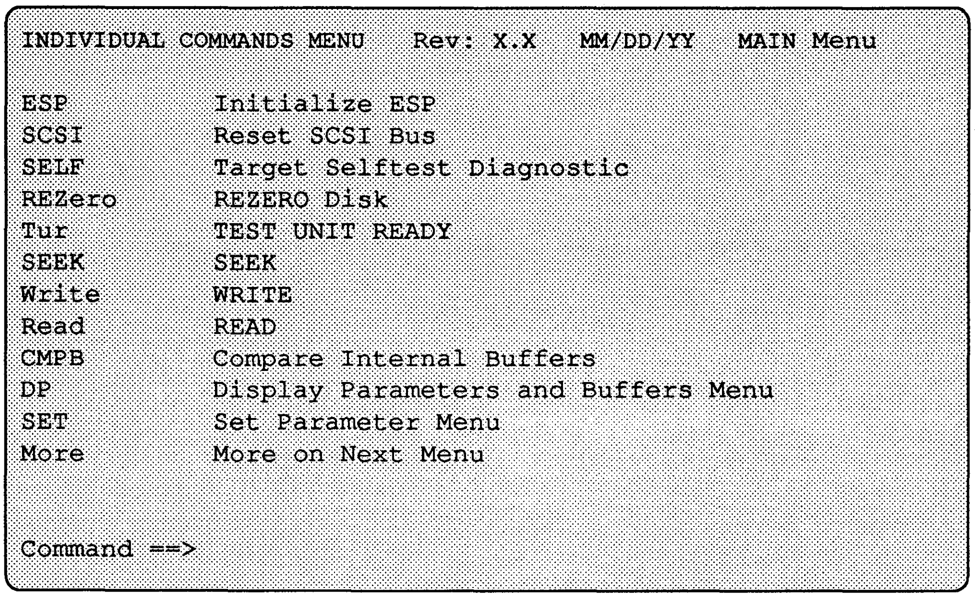
DR

The *Disconnect/Reselect Test* option on the ESP Disk Target Diagnostics Second Menu performs read data transfer operations on all attached disk devices until you halt the test by entering **#**.

Disk Individual Commands Main Menu

C

When you choose **C** from the ESP Disk Diagnostics Menu, the Disk Individual Commands Main Menu is displayed:



```
INDIVIDUAL COMMANDS MENU   Rev: X.X   MM/DD/YY   MAIN Menu

ESP          Initialize ESP
SCSI         Reset SCSI Bus
SELF         Target Selftest Diagnostic
REZero       REZERO Disk
Tur          TEST UNIT READY
SEEK         SEEK
Write        WRITE
Read         READ
CMPB         Compare Internal Buffers
DP           Display Parameters and Buffers Menu
SET          Set Parameter Menu
More         More on Next Menu


Command ==>
```

The options on the Disk Individual Commands Main Menu allow you to select and execute individual SCSI commands that support disk testing. Two screens comprise the Disk Individual Commands Menu. To go to the second screen, choose `More` on the Disk Individual Commands Main Menu.

ESP

The *Initialize ESP* option on the Disk Individual Commands Main Menu reinitializes the onboard SCSI controller chip (ESP).

SCSI

The *Reset SCSI Bus* option on the Disk Individual Commands Main Menu executes a SCSI bus reset on all devices attached to the SCSI bus.

NOTE *Any device that is reset will report a CHECK CONDITION in the status byte of the first command issued after the SCSI bus reset. The CHECK CONDITION should clear when you issue the next command. It is recommended that you issue a TEST UNIT READY command immediately after a SCSI bus reset.*

SELF

The *Target Selftest Diagnostic* option on the Disk Individual Commands Main Menu issues the SEND DIAGNOSTIC command, which causes execution of the selftest on the default drive.

REZ

The *REZERO Disk* option on the Disk Individual Commands Main Menu issues the REZERO command. For details on this command and its usage, see the product description manual for your particular SCSI device.

T

The *TEST UNIT READY* option on the Disk Individual Commands Main Menu issues the TEST UNIT READY command. For details on this command and its usage, see the product description manual for your particular SCSI device.

SEEK

The *SEEK* option on the Disk Individual Commands Main Menu issues the SEEK command. For details on this command and its usage, see the product description manual for your particular SCSI device.

W

The *WRITE* option on the Disk Individual Commands Main Menu issues the WRITE command. For details on this command and its usage, see the product description manual for your particular SCSI device.

R

The *READ* option on the Disk Individual Commands Main Menu issues the READ command. For details on this command and its usage, see the product description manual for your particular SCSI device.

CMPB

The *Compare Internal Buffers* option on the Disk Individual Commands Main Menu compares the contents of the internal default write and read buffers.

DP

When you choose the *Display Parameters and Buffers Menu* option on the Disk Individual Commands Main Menu, the Parameter and Buffer Display Menu is displayed. The options on this menu allow you to display the program default parameters, as well as the contents of the four internal data buffers and the SCSI command input and output buffers. For a complete description of the options on this menu, see the section entitled “Parameter and Buffer Display Menu,” later in this chapter.

SET

When you choose the *Set Parameter Menu* option on the Disk Individual Commands Main Menu, the Disk Parameter First Menu is displayed. The options on this menu allow you to specify settings that control how the ESP SCSI Diagnostic tests execute. An additional submenu allows you to display current parameter settings and the contents of internal buffers. For a complete discussion of the options on this menu, see the section entitled “Disk Parameter First Menu,” later in this chapter.

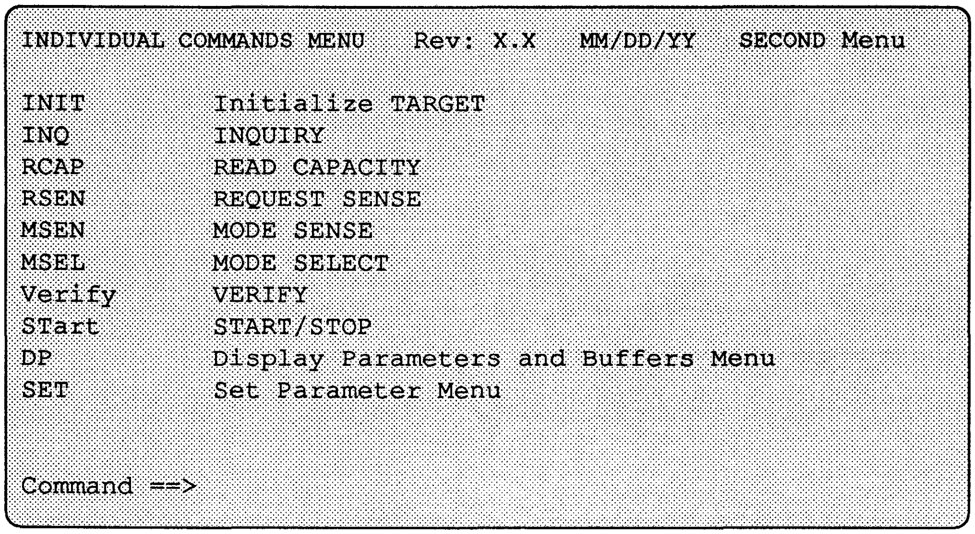
M

The *More on Next Menu* option on the Disk Individual Commands Main Menu allows you to go to the second Disk Individual Commands Menu screen. For a complete discussion of the options on this menu, see the next section entitled “Disk Individual Commands Second Menu.”

Disk Individual Commands Second Menu

M

When you choose **M** from the Disk Individual Commands Main Menu, the Disk Individual Commands Second Menu is displayed:



```
INDIVIDUAL COMMANDS MENU    Rev: X.X    MM/DD/YY    SECOND Menu

INIT          Initialize TARGET
INQ           INQUIRY
RCAP          READ CAPACITY
RSEN          REQUEST SENSE
MSEN          MODE SENSE
MSEL          MODE SELECT
Verify        VERIFY
STart         START/STOP
DP            Display Parameters and Buffers Menu
SET           Set Parameter Menu


Command ==>
```

INIT

The *Initialize Target* option on the Disk Individual Commands Second Menu reinitializes the default drive.

INQ

The *INQUIRY* option on the Disk Individual Commands Second Menu issues the INQUIRY command. For details on this command and its usage, see the product description manual for your particular SCSI device.

RCAP

The *READ CAPACITY* option on the Disk Individual Commands Second Menu issues the READ CAPACITY command. For details on this command and its usage, see the product description manual for your particular SCSI device.

RSEN

The *REQUEST SENSE* option on the Disk Individual Commands Second Menu issues the REQUEST SENSE command. For details on this command and its usage, see the product description manual for your particular SCSI device.

MSEN

The *MODE SENSE* option on the Disk Individual Commands Second Menu issues the MODE SENSE command. For details on this command and its usage, see the product description manual for your particular SCSI device.

MSEL

The *MODE SELECT* option on the Disk Individual Commands Second Menu issues the MODE SELECT command. For details on this command and its usage, see the product description manual for your particular SCSI device.

V

The *VERIFY* option on the Disk Individual Commands Second Menu issues the VERIFY command. For details on this command and its usage, see the product description manual for your particular SCSI device.

ST

The *START/STOP* option on the Disk Individual Commands Second Menu issues the START/STOP command. For details on this command and its usage, see the product description manual for your particular SCSI device.

DP

When you choose the *Display Parameters and Buffers Menu* option on the Disk Individual Commands Second Menu, the Parameter and Buffer Display Menu is displayed. The options on this menu allow you to display the program default parameters, as well as the contents of the four internal data buffers and the SCSI command input and output buffers. For a complete description of the options on this menu, see the section entitled "Parameter and Buffer Display Menu," later in this chapter.

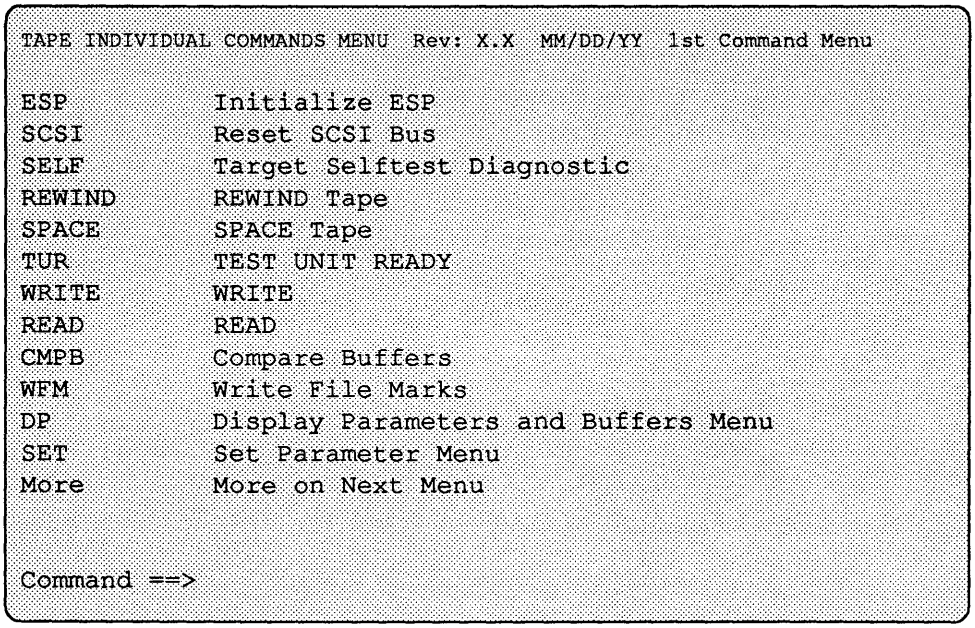
SET

When you choose the *Set Parameter Menu* option on the Disk Individual Commands Second Menu, the Disk Parameter First Menu is displayed. The options on this menu allow you to specify settings that control how the ESP SCSI Diagnostic tests execute. An additional submenu allows you to display current parameter settings and the contents of internal buffers. For a complete discussion of the options on this menu, see the section entitled "Disk Parameter First Menu," later in this chapter.

Tape Individual Commands First Menu

C

When you choose C from the SCSI Tape Diagnostics Menu, the Tape Individual Commands First Menu is displayed:



```
TAPE INDIVIDUAL COMMANDS MENU  Rev: X.X  MM/DD/YY  1st Command Menu

ESP          Initialize ESP
SCSI         Reset SCSI Bus
SELF         Target Selftest Diagnostic
REWIND       REWIND Tape
SPACE        SPACE Tape
TUR          TEST UNIT READY
WRITE        WRITE
READ         READ
CMPB         Compare Buffers
WFM          Write File Marks
DP           Display Parameters and Buffers Menu
SET          Set Parameter Menu
More         More on Next Menu


Command ==>
```

The options on the Tape Individual Commands First Menu allow you to select and execute individual SCSI commands that support tape testing. Two screens comprise the Disk Individual Commands Menu. To go to the second screen, choose `More` on the Tape Individual Commands First Menu.

Most of the options on the Tape Individual Commands First Menu perform the same functions for tape drive testing that the corresponding options on the Disk Individual Commands Main Menu, described previously, perform for disk drive testing. For more information on these options, refer to the section entitled "Disk Individual Commands Main Menu," earlier in this chapter.

The following options on the Tape Individual Commands First Menu are specific to tape testing only:

- REWIND Tape
- SPACE Tape
- Write File Marks

These options are described below.

REWIND

The *REWIND Tape* option on the Tape Individual Commands First Menu issues a REWIND command to the default tape drive. For details on this command and its usage, see the product description manual for your particular SCSI device.

SPACE

The *SPACE Tape* option on the Tape Individual Commands First Menu issues a SPACE command to the default tape drive. For details on this command and its usage, see the product description manual for your particular SCSI device.

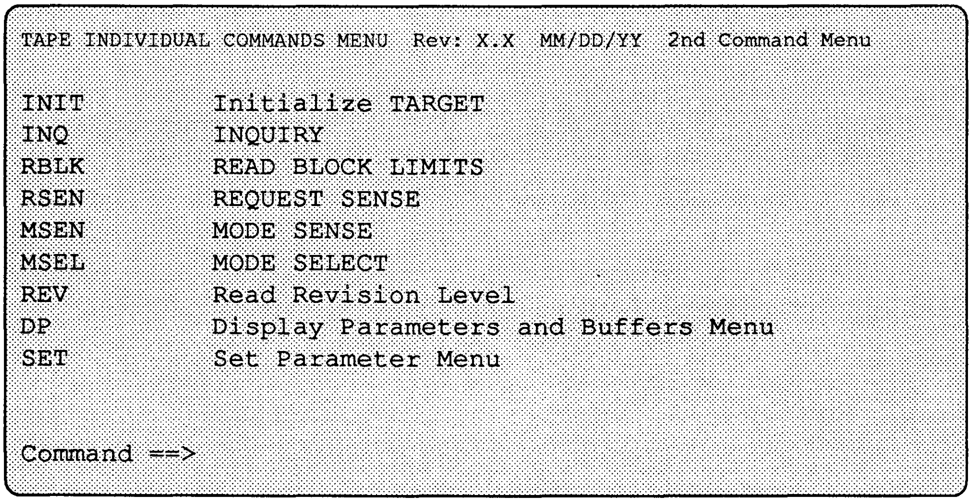
WFM

The *Write File Marks* option on the Tape Individual Commands First Menu issues a WRITE FILE MARK command. For details on this command and its usage, see the product description manual for your particular SCSI device.

Tape Individual Commands Second Menu

M

When you choose **M** from the Tape Individual Commands First Menu, the Tape Individual Commands Second Menu is displayed:



```
TAPE INDIVIDUAL COMMANDS MENU  Rev: X.X  MM/DD/YY  2nd Command Menu

INIT         Initialize TARGET
INQ          INQUIRY
RBLK         READ BLOCK LIMITS
RSEN         REQUEST SENSE
MSEN         MODE SENSE
MSEL         MODE SELECT
REV          Read Revision Level
DP           Display Parameters and Buffers Menu
SET          Set Parameter Menu


Command ==>
```

With two exceptions, the options on the Tape Individual Commands Second Menu perform the same functions for tape drive testing that the corresponding options on the Disk Individual Commands Second Menu, described previously, perform for disk drive testing. For more information on these options, refer to the section entitled "Disk Individual Commands Second Menu," earlier in this chapter.

The two options on the Tape Individual Commands Second Menu that are specific to tape testing are described below.

RBLK

The *READ BLOCK LIMITS* option on the Tape Individual Commands Second Menu issues the READ BLOCK LIMITS command. For details on this command and its usage, see the product description manual for your particular SCSI device.

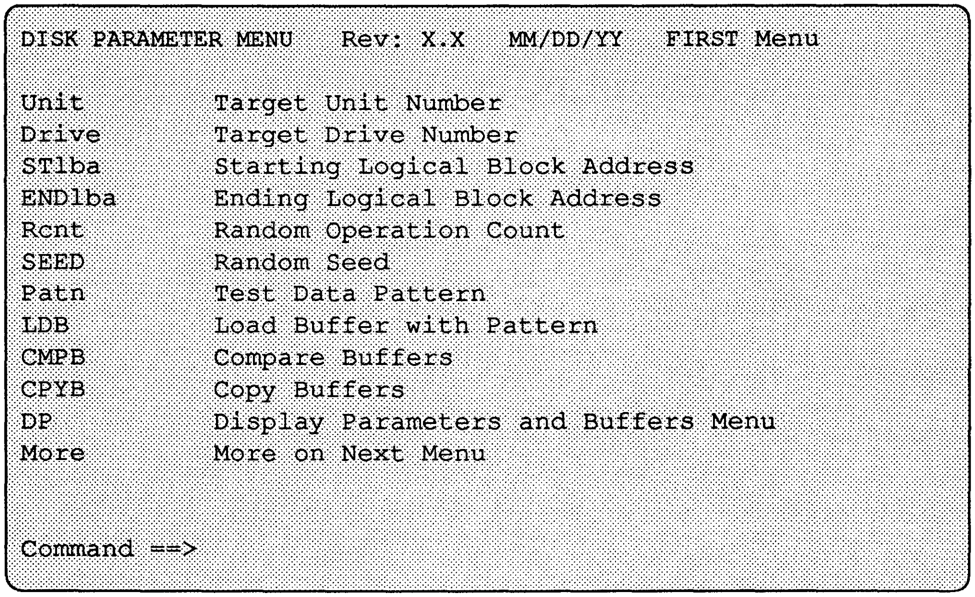
REV

The *Read Revision Level* option on the Tape Individual Commands Second Menu issues the READ REVISION LEVEL command. For details on this command and its usage, see the product description manual for your particular SCSI device.

Disk Parameter First Menu

SET

When you choose the *Set Parameter Menu* option from any disk test menu on which it appears, the Disk Parameter First Menu is displayed:



```
DISK PARAMETER MENU   Rev: X.X   MM/DD/YY   FIRST Menu

Unit        Target Unit Number
Drive       Target Drive Number
STlba       Starting Logical Block Address
ENDlba      Ending Logical Block Address
Rcnt        Random Operation Count
SEED        Random Seed
Patn        Test Data Pattern
LDB         Load Buffer with Pattern
CMPB        Compare Buffers
CPYB        Copy Buffers
DP          Display Parameters and Buffers Menu
More        More on Next Menu


Command ==>
```

The options on the Disk Parameter First Menu allow you to specify settings that control how the ESP SCSI Diagnostic tests execute, including the pass count for random tests, choice of target drive, address range, and data pattern to be used in pattern testing. An additional submenu allows you to display current parameter settings and the contents of internal buffers. Two screens comprise the Disk Parameter Menu. To go to the second screen, choose `More` on the Disk Parameter First Menu.

U

The *Target Unit Number* option on the Disk Parameter First Menu specifies the Logical Unit Number associated with a Command Descriptor Block when a SCSI command is sent. For currently supported devices, this value *must* be 0.

D

The *Target Drive Number* option on the Disk Parameter First Menu specifies the SCSI bus ID address associated with an attached device. The permissible range of values is 0–6.

ST

The *Starting Logical Block Address* option on the Disk Parameter First Menu specifies the default starting Logical Block Address. The minimum value for this parameter is 0, and during initialization, it is set to 0.

END

The *Ending Logical Block Address* option on the Disk Parameter First Menu specifies the default ending Logical Block Address. The minimum value for this parameter is 0; during initialization, it is set to 1000. The maximum value may be greater than the capacity of any device. To determine the actual maximum LBA of an attached device, use the READ CAPACITY command.

R

The *Random Operation Count* option on the Disk Parameter First Menu specifies the number of times certain random tests — such as the Random Read/Write/Verify Test — are performed.

SEED

The *Random Seed* option on the Disk Parameter First Menu allows you to specify the random seed used when generating random numbers. The initialization default is 68739.

P

The *Test Data Pattern* option on the Disk Parameter First Menu allows you to specify the data pattern to be used in pattern testing, as well as other associated parameters. The patterns are shown in the following table. The initialization default is 0.

Value	*Pattern*
0	Load buffer with user-supplied constant data byte.
1	Increment user-supplied data byte by one.
2	Decrement user-supplied data byte by one.
3	Increment user-supplied data byte by a user-supplied data byte.
4	Decrement user-supplied data byte by a user-supplied data byte.
5	Generate a random data pattern using the default random seed.

LDB

The *Load Buffer with Pattern* option on the Disk Parameter First Menu lets you select an internal buffer and load a data pattern into it.

CMPB

The *Compare Buffers* option on the Disk Parameter First Menu allows you to select two of the internal data buffers and compare their contents. The comparison halts on the first error encountered.

CPYB

The *Copy Buffers* option on the Disk Parameter First Menu allows you to copy the contents of one internal data buffer onto another.

DP

When you choose the *Display Parameters and Buffers Menu* option on the Disk Parameter First Menu, the Parameter and Buffer Display Menu is displayed. The options on this menu allow you to display the program default parameters, as well as the contents of the four internal data buffers and the SCSI command and output buffers. For a complete description of the options on this menu, see the section entitled "Parameter and Buffer Display Menu," later in this chapter.

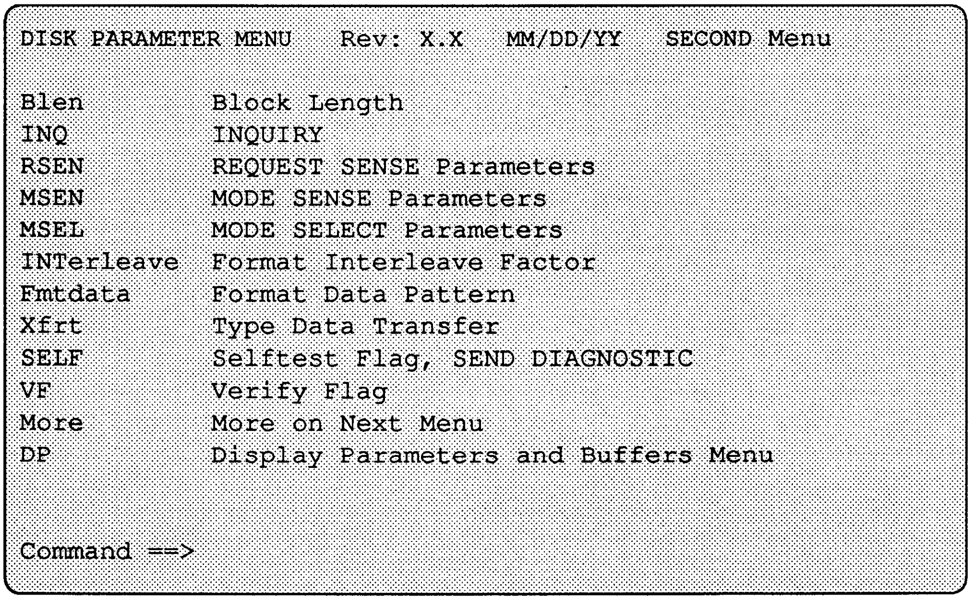
M

The *More on Next Menu* option on the Disk Parameter First Menu allows you to go to the second Disk Parameter Menu screen. For a complete discussion of the options on this menu, see the next section entitled "Disk Parameter Second Menu."

Disk Parameter Second Menu

M

When you choose the *More on Next Menu* option on the Disk Parameter First Menu, the Disk Parameter Second Menu is displayed:



```
DISK PARAMETER MENU   Rev: X.X   MM/DD/YY   SECOND Menu

Blen         Block Length
INQ          INQUIRY
RSEN         REQUEST SENSE Parameters
MSEN         MODE SENSE Parameters
MSEL         MODE SELECT Parameters
INTerleave   Format Interleave Factor
Fmtdata      Format Data Pattern
Xfrt         Type Data Transfer
SELF         Selftest Flag, SEND DIAGNOSTIC
VF           Verify Flag
More         More on Next Menu
DP           Display Parameters and Buffers Menu


Command ==>
```

B

The *Block Length* option on the Disk Parameter Second Menu allows you to alter the default block length.

CAUTION **Exercise extreme caution when using this parameter. Many devices support only specific block lengths. For example, the internal embedded-SCSI Quantum device supports block lengths of 512, 1024, and 2048 bytes only. The Emulex MT02 tape device supports a block length of 512 bytes only. Refer to the Product Description Manual for the block lengths supported by your particular device before using this parameter.**

INQ

The *INQUIRY* option on the Disk Parameter Second Menu allows you to change parameters associated with the INQUIRY command.

RSEN

The *REQUEST SENSE Parameters* option on the Disk Parameter Second Menu allows you to change parameters associated with the REQUEST SENSE command.

MSEN

The *MODE SENSE Parameters* option on the Disk Parameter Second Menu allows you to change parameters associated with the MODE SENSE command.

MSEL

The *MODE SELECT Parameters* option on the Disk Parameter Second Menu allows you to change parameters associated with the MODE SELECT command.

INT

The *Format Interleave Factor* option on the Disk Parameter Second Menu allows you to alter the interleave parameter used during formatting of a disk device.

F

The *Format Data Pattern* option on the Disk Parameter Second Menu allows you to change the single-byte format data pattern used during formatting of a disk device.

X

The *Type Data Transfer* option on the Disk Parameter Second Menu allows you to specify the default type of data transfer operation, either DMA or non-DMA.

SELF

The *Selftest Flag, SEND DIAGNOSTIC* option on the Disk Parameter Second Menu allows you to change the selftest parameter used during the execution of a SEND DIAGNOSTIC command.

VF

The *Verify Flag* option on the Disk Parameter Second Menu allows you to change the verify parameter used during the disk tests. If set, a test using this parameter requests the disk controller to verify the physical storage medium associated with the current LBA after the seek operation has completed. If not set, no verification is performed.

M

The *More on Next Menu* option on the Disk Parameter Second Menu allows you to go to the third Disk Parameter Menu screen. For a complete discussion of the options on this menu, see the next section entitled "Disk Parameter Third Menu."

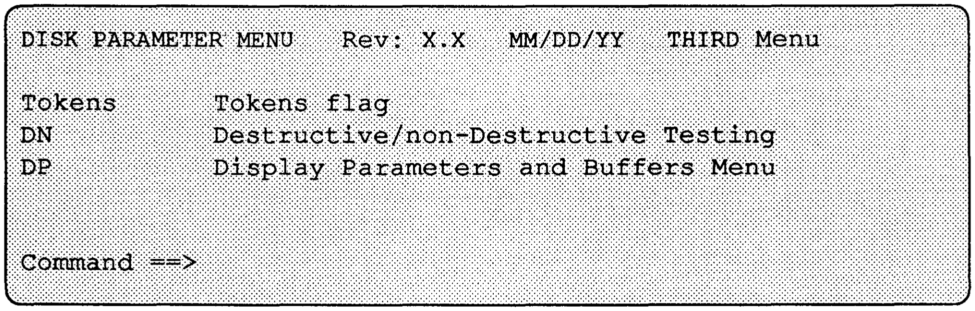
DP

When you choose the *Display Parameters and Buffers Menu* option on the Disk Parameter Second Menu, the Parameter and Buffer Display Menu is displayed. The options on this menu allow you to display the program default parameters, as well as the contents of the four internal data buffers and the SCSI command input and output buffers. For a complete description of the options on this menu, see the section entitled "Parameter and Buffer Display Menu," later in this chapter.

Disk Parameter Third Menu

M

When you choose the *More on Next Menu* option on the Disk Parameter Second Menu, the Disk Parameter Third Menu is displayed:



```
DISK PARAMETER MENU   Rev: X.X   MM/DD/YY   THIRD Menu

Tokens        Tokens flag
DN            Destructive/non-Destructive Testing
DP            Display Parameters and Buffers Menu


Command ==>
```

T

The *Tokens Flag* option on the Disk Parameter Third Menu allows you to enable or disable the display of message tokens. The display of tokens is disabled by default.

DN

"The *Destructive/Non-Destructive Testing* option on the Disk Parameter Third Menu allows you to specify whether or not the disk contents can be altered by testing. The alteration of the disk contents is disabled by default.

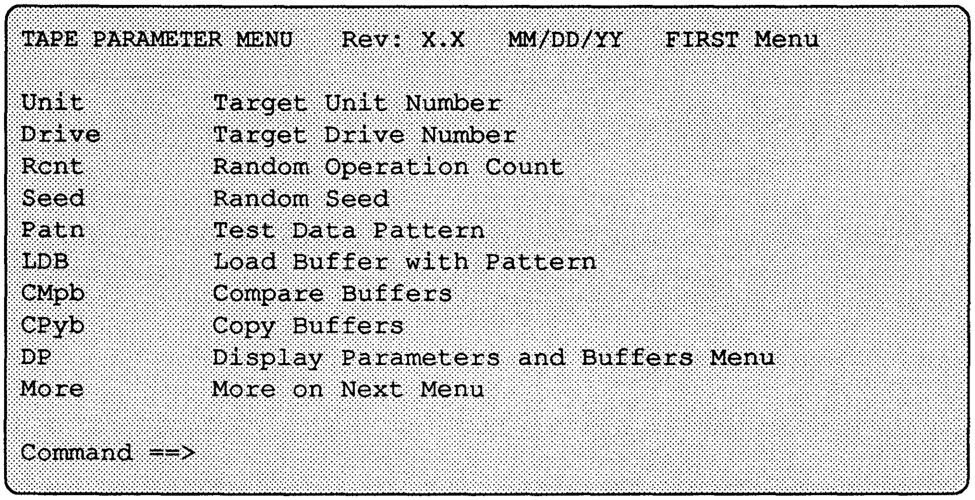
DP

When you choose the *Display Parameters and Buffers Menu* option on the Disk Parameter Third Menu, the Parameter and Buffer Display Menu is displayed. The options on this menu allow you to display the program default parameters, as well as the contents of the four internal data buffers and the SCSI command input and output buffers. For a complete description of the options on this menu, see the section entitled "Parameter and Buffer Display Menu," later in this chapter.

Tape Parameter First Menu

SET

When you choose the *Set Parameter Menu* option from any tape test menu on which it appears, the Tape Parameter First Menu is displayed:



```
TAPE PARAMETER MENU   Rev: X.X   MM/DD/YY   FIRST Menu

Unit        Target Unit Number
Drive       Target Drive Number
Rcnt        Random Operation Count
Seed        Random Seed
Patn        Test Data Pattern
LDB         Load Buffer with Pattern
CMpb        Compare Buffers
CPyb        Copy Buffers
DP          Display Parameters and Buffers Menu
More        More on Next Menu

Command ==>
```

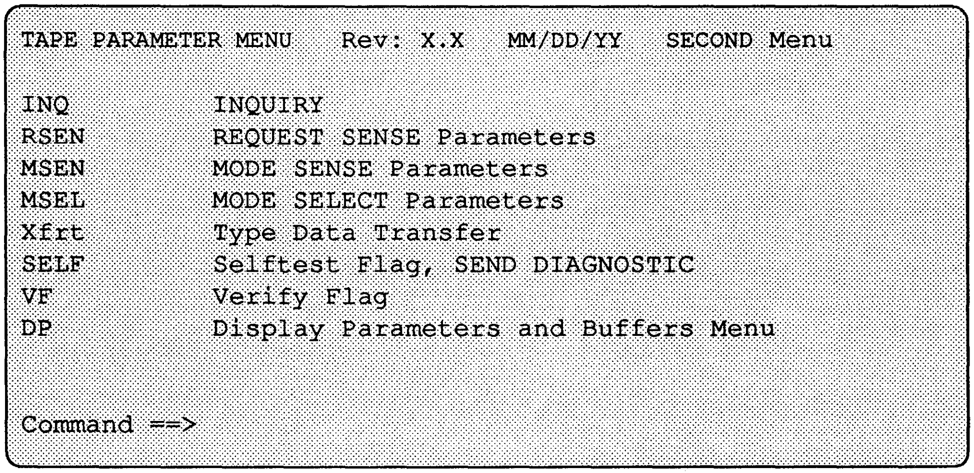
The options on the Tape Parameter First Menu are a subset of the options on the Disk Parameter First Menu, and they perform the same functions as the corresponding options on the Disk Parameter First Menu. For information about these options, see the section entitled "Disk Parameter First Menu," earlier in this chapter. On the Tape Parameter First Menu, the *Target Unit Number* and *Target Drive Number* options specify the numbers of tape, not disk, devices.

Two screens comprise the Tape Parameter Menu. To go to the second screen, choose `More` on the Tape Parameter First Menu.

Tape Parameter Second Menu

M

When you choose the *More on Next Menu* option on the Tape Parameter First Menu, the Tape Parameter Second Menu is displayed:



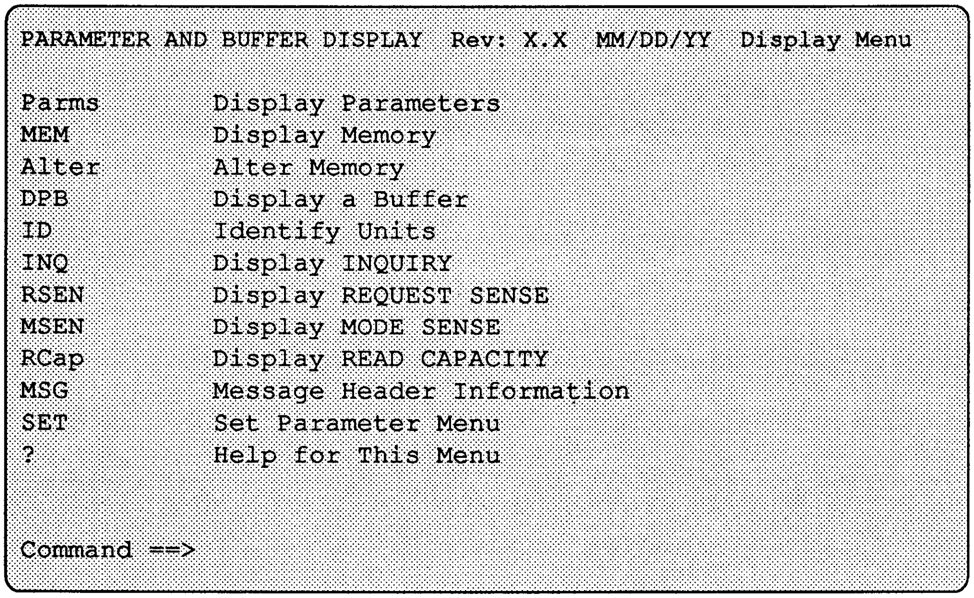
```
TAPE PARAMETER MENU   Rev: X.X   MM/DD/YY   SECOND Menu

INQ         INQUIRY
RSEN        REQUEST SENSE Parameters
MSEN        MODE SENSE Parameters
MSEL        MODE SELECT Parameters
Xfrt        Type Data Transfer
SELF        Selftest Flag, SEND DIAGNOSTIC
VF          Verify Flag
DP          Display Parameters and Buffers Menu


Command ==>
```

The options on the Tape Parameter Second Menu are a subset of the options on the Disk Parameter Second Menu. For information about these options, see the section entitled “Disk Parameter Second Menu,” earlier in this chapter.

Parameter and Buffer Display Menu

DP

When you choose the Display Parameters and Buffers Menu option on any menu on which it appears, the Parameter and Buffer Display Menu is displayed:



```
PARAMETER AND BUFFER DISPLAY  Rev: X.X  MM/DD/YY  Display Menu

Parms       Display Parameters
MEM         Display Memory
Alter       Alter Memory
DPB         Display a Buffer
ID          Identify Units
INQ         Display INQUIRY
RSEN        Display REQUEST SENSE
MSEN        Display MODE SENSE
RCap        Display READ CAPACITY
MSG         Message Header Information
SET         Set Parameter Menu
?           Help for This Menu


Command ==>
```

The options on the Parameter and Buffer Display Menu allow you to display the program default parameters. You can view the values only; if you want to change values, return to the Parameter Menu by choosing the `SET` option. The Parameter and Buffer Display Menu also allows you to view the contents of the four internal data buffers and the SCSI command input and output buffers.

P

The *Display Parameters* option on the Parameter and Buffer Display Menu allows you to display all program default parameters.

MEM

The *Display Memory* option on the Parameter and Buffer Display Menu allows you to display the contents of the memory range that you specify. The display consists of 16 lines of 16 bytes per line, and each line is preceded by the associated memory address.

A

The *Alter Memory* option on the Parameter and Buffer Display Menu allows you to change the contents of the byte location that you specify in memory.

DPB

The *Display a Buffer* option on the Parameter and Buffer Display Menu allows you to display the contents of one of the four internal data buffers.

ID

The *Identify Units* option on the Parameter and Buffer Display Menu allows you to identify the devices attached to the SCSI bus. Identification is accomplished by issuing INQUIRY commands to all addresses on the SCSI bus. The contents of the common INQUIRY buffer is displayed, followed by the contents of the INQUIRY buffers, each associated with an individual SCSI bus address.

INQ

The *Display INQUIRY* option on the Parameter and Buffer Display Menu allows you to display the contents of the common INQUIRY buffer.

RSEN

The *Display REQUEST SENSE* option on the Parameter and Buffer Display Menu allows you to display the contents of the REQUEST SENSE buffer.

MSEN

The *Display MODE SENSE* option on the Parameter and Buffer Display Menu allows you to display the contents of the MODE SENSE buffer.

RC

The *Display READ CAPACITY* option on the Parameter and Buffer Display Menu allows you to display the contents of the READ CAPACITY buffer.

MSG

The *Message Header Information* option on the Parameter and Buffer Display Menu allows you to display information related to the format of the message tokens.

SET

The *Set Parameter Menu* option on the Parameter and Buffer Display Menu allows you to go directly to the Parameter Menu, which you can use to change parameter values.

?

The *Help for This Menu* option on the Parameter and Buffer Display Menu displays a Help message that briefly describes the options on the current menu.

29.8. Supported SCSI Commands

The tables in this section list the SCSI commands supported by the internal embedded-SCSI Quantum disk controller, the CDC WREN IV SCSI disk controller, and the MT02 tape controller.

Internal Embedded-SCSI Quantum Disk Controller Commands

The SCSI commands supported by the internal embedded-SCSI Quantum disk are listed in the following table:

Opcode	*Command*	*Supported*
0x00	TEST UNIT READY	Yes
0x01	REZERO UNIT	Yes
0x03	REQUEST SENSE	Yes
0x04	FORMAT UNIT	Yes
0x07	REASSIGN BLOCKS	No
0x08	READ	Yes
0x0A	WRITE	Yes
0x0B	SEEK	Yes
0x12	INQUIRY	Yes
0x15	MODE SELECT	Yes
0x16	RESERVE	No
0x17	RELEASE	No
0x1A	MODE SENSE	Yes
0x1B	START/STOP UNIT	Yes
0x1D	SEND DIAGNOSTIC	Yes
0x25	READ CAPACITY	Yes
0x28	READ EXTENDED	No
0x2A	WRITE EXTENDED	No
0x2B	SEEK EXTENDED	No
0x2E	WRITE AND VERIFY	No
0x2F	VERIFY	Yes
0x37	READ DEFECT DATA	No
0x3B	WRITE BUFFER	No
0x3C	READ BUFFER	No

CDC WREN IV SCSI Disk Controller Commands

The SCSI commands supported by the external CDC WREN IV SCSI disk are listed in the following table:

Opcode	*Command*	*Supported*
0x00	TEST UNIT READY	Yes
0x01	REZERO UNIT	Yes
0x03	REQUEST SENSE	Yes
0x04	FORMAT UNIT	Yes
0x07	REASSIGN BLOCKS	No
0x08	READ	Yes
0x0A	WRITE	Yes
0x0B	SEEK	Yes
0x12	INQUIRY	Yes
0x15	MODE SELECT	Yes
0x16	RESERVE	No
0x17	RELEASE	No
0x1A	MODE SENSE	Yes
0x1B	START/STOP UNIT	Yes
0x1C	RECEIVE DIAGNOSTIC RESULTS	No
0x1D	SEND DIAGNOSTIC	Yes
0x25	READ CAPACITY	Yes
0x28	READ EXTENDED	No
0x2A	WRITE EXTENDED	No
0x2B	SEEK EXTENDED	No
0x2E	WRITE AND VERIFY	No
0x2F	VERIFY	Yes
0x37	READ DEFECT DATA	No
0x3B	WRITE BUFFER	No
0x3C	READ BUFFER	No

MT02 Tape Controller Commands

The SCSI commands supported by the MT02 tape controller are listed in the following table:

Opcode	*Command*	*Supported*
0x00	TEST UNIT READY	Yes
0x01	REWIND	Yes
0x03	REQUEST SENSE	Yes
0x05	READ BLOCK LIMITS	Yes
0x08	READ	Yes
0x0A	WRITE	Yes
0x10	WRITE FILE MARK	Yes
0x11	SPACE	Yes
0x12	INQUIRY	Yes
0x13	VERIFY	Yes
0x14	RECOVER BUFFERED DATA	No
0x15	MODE SELECT	Yes
0x16	RESERVE	No
0x17	RELEASE	No
0x18	COPY	No
0x19	ERASE	No
0x1A	MODE SENSE	Yes
0x1B	LOAD/UNLOAD	Yes
0x1D	SEND DIAGNOSTIC	Yes
0x1E	PREVENT/ALLOW MEDIUM REMOVAL	No

29.9. Messages

The messages displayed by the Sun-3/80 ESP SCSI Diagnostic are composed of two parts: a header consisting of tokens identifying the source of the message, and a string of descriptive comments and data.

Executive Message Numbers

Executive message numbers are in the range 800–999 and are followed immediately by a "$" delimiter. These informational messages perform three basic functions:

- Report on the progress of tests.
- Provide additional information after an error is reported.
- Support the software debugging of reported errors.

the Executive message numbers and their meanings are shown in the following table:

Number	*Interpretation*
800$	Status Update
803$	Stop-On-Error
804$	Input Parameters
805$	CDB Parameters
806$	Wait-On-Selftest
809$	FORMAT, parameter # 2

Number	*Interpretation*
810$	FORMAT, parameter # 3
811$	FORMAT, parameter # 4
814$	Progress
815$	Initialize
816$	Display
817$	Set Defaults
818$	Get Defaults
819$	User Query
822$	Incoming Data
823$	Data Requested
826$	Not Implemented

Executive Error Numbers

Executive error numbers are displayed in the leftmost section of a composite error message. Error numbers are within the range 0–799 and are followed immediately by a "$" delimiter. The available error numbers are listed in the following table:

Number	*General Error Description*
1$	TEST UNIT READY, error # 1
2$	TEST UNIT READY, error # 2
5$	INQUIRY, error # 1
6$	INQUIRY, error # 2
9$	REQUEST SENSE, error # 1
10$	REQUEST SENSE, error # 2
13$	MODE SENSE, error # 1
14$	MODE SENSE, error # 2
17$	MODE SELECT, error # 1
18$	MODE SELECT, error # 2
20$	READ, error # 1
21$	READ, error # 1
24$	READ CAPACITY, error # 1
25$	READ CAPACITY, error # 2
28$	WRITE, error # 1
29$	WRITE, error # 2
31$	SEEK, error # 1
32$	SEEK, error # 2

Number	*General Error Description*
35$	REZERO, error # 1
36$	REZERO, error # 2
39$	VERIFY, error # 1
40$	VERIFY, error # 2
43$	SEND DIAGNOSTIC, error # 1
44$	SEND DIAGNOSTIC, error # 2
45$	SEND DIAGNOSTIC, error # 3
47$	RECEIVE DIAGNOSTIC, error # 1
48$	RECEIVE DIAGNOSTIC, error # 2
51$	START/STOP, error # 1
52$	START/STOP, error # 2
55$	FORMAT, error # 1
56$	FORMAT, error # 2
70$	TAPE WRITE, error # 1
71$	TAPE WRITE, error # 2
74$	TAPE READ, error # 1
75$	TAPE READ, error # 2
78$	REWIND, error # 1
79$	REWIND, error # 2
82$	ERASE, ERROR # 1
83$	ERASE, error # 2
86$	LOAD, error # 1
87$	LOAD, error # 2
90$	MEDIUM, error # 1
91$	MEDIUM, error # 2
94$	READ BLOCK LIMIT, error # 1
95$	READ BLOCK LIMIT, error # 2
98$	TAPE SEND DIAGNOSTIC, error # 1
100$	SPACE, error # 1
101$	SPACE, error # 2
104$	WRITE FILE MARK, error # 1
105$	WRITE FILE MARK, error # 2

Number	*General Error Description*
108$	READ REVISION LEVEL, error # 1
109$	READ REVISION LEVEL, error # 2
112$	Buffer Compare, error # 1
113$	Copy Buffer, error # 1
114$	Check ASCII Digit, error # 1
117$	Check Tape BUSY, error # 1
118$	Check Tape BUSY, error # 2
119$	Check Tape BUSY, error # 3
120$	Check Tape BUSY, error # 4
121$	Check Tape BUSY, error # 5
122$	Check Tape BUSY, error # 6
123$	Check Tape BUSY, error # 7
124$	Check Tape BUSY, error # 8
125$	Check Tape BUSY, error # 9
128$	Check Shoebox Disk, error # 1
130$	Check Configuration, error # 1
131$	Check Configuration, error # 2
133$	Check Controller Type, error # 1
135$	Test Disk, error # 1
137$	Quick Disk Test, error # 1
138$	Quick Disk Test, error # 2
139$	Quick Disk Test, error # 3
140$	Quick Disk Test, error # 4
143$	Display Buffer, error # 1
144$	Display Buffer, error # 2
145$	Display Buffer, error # 3
147$	Default Disk, error # 1
148$	Default Disk, error # 2
150$	Default Starting LBA
152$	Default Ending LBA
154$	FORMAT setup, error # 1
155$	FORMAT setup, error # 2
156$	FORMAT setup, error # 3
157$	FORMAT setup, error # 4

Number	*General Error Description*
160$	Default DMA Flag
162$	Default Random Count
164$	Default Random Seed
166$	Pattern setup, error # 1
167$	Pattern setup, error # 2
168$	Pattern setup, error # 3
169$	Pattern setup, error # 4
170$	Pattern setup, error # 5
171$	Pattern setup, error # 6
174$	Load Buffer, error # 1
175$	Load Buffer, error # 2
176$	Load Buffer, error # 3
177$	Load Buffer, error # 4
180$	Compare Buffer, error # 1
181$	Compare Buffer, error # 2
182$	Compare Buffer, error # 3
183$	Compare Buffer, error # 4
184$	Compare Buffer, error # 5
185$	Compare Buffer, error # 6
186$	Compare Buffer, error # 7
190$	Copy Buffer, error # 1
191$	Copy Buffer, error # 2
192$	Copy Buffer, error # 3
193$	Copy Buffer, error # 4
194$	Copy Buffer, error # 5
195$	Copy Buffer, error # 6
196$	Copy Buffer, error # 7
199$	SEEK setup, error # 1
202$	READ setup, error # 1
203$	READ setup, error # 2
206$	WRITE setup, error # 1
207$	WRITE setup, error # 2
210$	VERIFY setup, error # 1
211$	VERIFY setup, error # 2
214$	Incrementing Byte Pattern, error # 1
215$	Incrementing Byte Pattern, error # 2

Number	*General Error Description*
218$	REQUEST SENSE setup, error # 1
219$	REQUEST SENSE setup, error # 2
222$	MODE SENSE setup, error # 1
223$	MODE SENSE setup, error # 2
226$	MODE SENSE setup, error # 1
227$	MODE SENSE setup, error # 2
230$	MODE SELECT setup, error # 1
231$	MODE SELECT setup, error # 2
234$	READ BLOCK LENGTH setup, error # 1
235$	READ BLOCK LENGTH setup, error # 2
238$	FORMAT Interleave Setup, error # 1
239$	FORMAT Interleave Setup, error # 2
242$	FORMAT Data, error # 1
243$	FORMAT Data, error # 2
246$	MODE SELECT Bytecount, error # 1
247$	MODE SELECT Bytecount, error # 2
250$	Error Summary
253$	Default Disk setup, error # 1
256$	ESP Send Command, error # 1
259$	Write/Read Via DMA, error # 1
260$	Write/Read Via DMA, error # 2
261$	Write/Read Via DMA, error # 3
262$	Write/Read Via DMA, error # 4
263$	Write/Read Via DMA, error # 5
264$	Write/Read Via DMA, error # 6
267$	Write/Read Via Bus, error # 1
268$	Write/Read Via Bus, error # 2
269$	Write/Read Via Bus, error # 3
270$	Write/Read Via Bus, error # 4
271$	Write/Read Via Bus, error # 5
272$	Write/Read Via Bus, error # 6
273$	Write/Read Via Bus, error # 7
276$	FUNCTION COMPLETE, error # 1
277$	FUNCTION COMPLETE, error # 2
278$	FUNCTION COMPLETE, error # 3

Number	*General Error Description*
279$	FUNCTION COMPLETE, error # 4
280$	FUNCTION COMPLETE, error # 5
281$	FUNCTION COMPLETE, error # 6
282$	FUNCTION COMPLETE, error # 7
283$	FUNCTION COMPLETE, error # 8
284$	FUNCTION COMPLETE, error # 9
287$	Initialize Unit, error # 1
288$	Initialize Unit, error # 2
289$	Initialize Unit, error # 3
290$	Initialize Unit, error # 4
291$	Initialize Unit, error # 5
294$	Reset SCSI Bus, error # 1
297$	Initialize Tape, error # 1
298$	Initialize Tape, error # 2
299$	Initialize Tape, error # 3
300$	Initialize Tape, error # 4
301$	Initialize Tape, error # 5
302$	Initialize Tape, error # 6
303$	Initialize Tape, error # 7
304$	Initialize Tape, error # 8
305$	Initialize Tape, error # 9
306$	Initialize Tape, error # 10
307$	Initialize Tape, error # 11
308$	Initialize Tape, error # 12
311$	Get Data Via Bus, error # 1
312$	Get Data Via Bus, error # 2
315$	Send Data Via Bus, error # 1
316$	Send Data Via Bus, error # 2
319$	Mapit, error # 1
320$	Mapit, error # 2
321$	Mapit, error # 3
330$	Recovery, error # 1
331$	Recovery, error # 2
334$	Main, error # 1
335$	Main, error # 3
336$	Main, error # 4
337$	Main, error # 5
341$	Set Autovector # 1, error # 1
342$	Set Autovector # 2, error # 1

Number	*General Error Description*
343$	Set Autovector # 3, error # 1
344$	Set Autovector # 4, error # 1
345$	Set Autovector # 5, error # 1
346$	Set Autovector # 6, error # 1
347$	Set Autovector # 7, error # 1
350$	TARGET Init Select, error # 1
351$	TARGET Init Select, error # 2
352$	TARGET Init Select, error # 3
355$	Get REQUEST SENSE, error # 1
356$	Get REQUEST SENSE, error # 2
359$	Non-DMA Wrapup, error # 1
360$	Non-DMA Wrapup, error # 2
361$	Non-DMA Wrapup, error # 3
362$	Non-DMA Wrapup, error # 4
363$	Non-DMA Wrapup, error # 5
364$	Non-DMA Wrapup, error # 6
365$	Non-DMA Wrapup, error # 7
366$	Non-DMA Wrapup, error # 8
369$	DMA Wrapup, error # 1
370$	DMA Wrapup, error # 2
371$	DMA Wrapup, error # 3
372$	DMA Wrapup, error # 4
373$	DMA Wrapup, error # 5
374$	DMA Wrapup, error # 6
375$	DMA Wrapup, error # 7
376$	DMA Wrapup, error # 8
379$	D/R, Get Data Command
380$	D/R, Send Data Command
400$	ABORT
401$	Invalid Entry
402$	Data Compare Error
403$	Bad LBA
406$	Can't Isolate
407$	No Error Occurred
410$	Disk NOT READY
413$	TAPE NOT AVAILABLE

Message Headers

Message headers are composed of tokens that identify the source of the message. You can enable or disable their display with the Tokens Flag option on the Disk Parameter Third Menu. These tokens, which are abbreviations for longer strings, are shown in the following table:

Token	*Description*
C|	User command
INTR|	Interrupts/exceptions
INIT|	Software initialization
I1|	Initialize Unit/Drive
I2|	Initialize Tape
X1|	Resetting the SCSI Bus
X3|	Initiating the ESP chip
X5|	TEST UNIT READY after a SCSI Bus Reset
X6|	Init all disk units after a SCSI Bus Reset
X7|	Sending a Command Descriptor Block (CDB) to Target
X8|	Sending a Command Descriptor Block (CDB) to Target
X9|	Sending a Command Descriptor Block (CDB) to Target
X10|	Resetting the Target
X14|	Checking Configuration
X15|	Checking Controller Type
X16|	Checking for a Shoebox Disk
X20|	Retrieving REQUEST SENSE data
X21|	Retrieving REQUEST SENSE data
X22|	Retrieving REQUEST SENSE data
X25|	Typing the Target
X30|	Initializing the Target
X32|	Non-DMA DATAIN
X34|	Non-DMA DATAOUT
X36|	Non-DMA data transfer
X38|	DMA data transfer
X50|	Waiting for a FUNCTION COMPLETE
WU|	Wrapup, non-DMA command
WA|	Wrapup, DMA command
TUR|	TEST UNIT READY command
RS|	REQUEST SENSE command
MS|	MODE SENSE command
MSL|	MODE SELECT command
RD|	READ command
RC|	READ CAPACITY command
WR|	WRITE command
SK|	SEEK command
RZ|	REZERO command
VF|	VERIFY command
SD|	SEND DIAGNOSTIC command
DR|	RECEIVE DIAGNOSTIC RESULTS command
SS|	START/STOP command
FM|	FORMAT command
IQ|	INQUIRY command

Token	*Description*
TRD|	Tape READ command
TRB|	Tape READ BLOCK LIMITS command
TWR|	Tape WRITE command
TRW|	Tape REWIND command
TER|	Tape ERASE command
TLD|	Tape LOAD command
TMM|	Tape PREVENT/ALLOW MEDIUM REMOVAL command
TRV|	Tape READ REVISION LEVEL command
Q10|	Quick Disk Test
Q20|	Quick Tape Test

29.10. Glossary

CDB	Command Descriptor Block.
COMMAND	SCSI bus CDB transfer phase.
DATAIN	SCSI bus data transfer phase.
DATAOUT	SCSI bus data transfer phase.
Drive	SCSI bus ID number for a device.
ESP	Extended SCSI Processor. The onboard SCSI bus controller chip.
LBA	Logical Block Address.
MSGIN	SCSI bus Target-to-ESP chip message phase.
MSGOUT	SCSI bus ESP-to-Target message phase.
LUN	Logical Unit Number.
SCSI	Small Computer System Interface.
STATUS	SCSI bus status information transfer phase.
Unit	Logical Unit Number in a SCSI Command Descriptor Block.

30

SPARCsystem 330 ESP SCSI Diagnostic

30

SPARCsystem 330 ESP SCSI Diagnostic

30.1. General Description

An ESP SCSI Diagnostic is provided to verify the functionality of the ESP chip in the SPARCsystem 330. The SPARCsystem 330 ESP SCSI Diagnostic is similar to the SCSI Subsystem Diagnostic described in this manual. With two exceptions, which are discussed under "The Diagnostic Menus" in this chapter, the options on the Disk Controller Diagnostics Menus and Tape Drive Diagnostics Menus are the same. An additional menu provides tests that are specific to the ESP SCSI subsystem.

30.2. Hardware Requirements

The following hardware is required to run the SPARCsystem 330 ESP SCSI Diagnostic:

- A SPARCsystem 330 CPU board with onboard ESP chip and minimum 4 Mbytes of RAM
- A tested and operational internal embedded SCSI disk drive
- All hardware supporting the onboard ESP chip previously tested and operational
- A monitor
- A keyboard
- A boot device (local disk, local tape, or remote disk over Ethernet)

30.3. Starting the Diagnostic

For information on starting the Exec, refer to Chapter 2, "Using the SunDiagnostic Executive." After you have started the Exec, choose the SPARCsystem 330 Diagnostic from the Diagnostics Menu.

30.4. User Interface

The user interface of the SPARCsystem 330 Diagnostic adheres to the menu standards of the Exec. Each test may be selected from a menu by typing the letter or letters displayed in upper case in the column on the left side of the menu.

30.5. The Diagnostic Menus

As mentioned earlier, the diagnostic performs similarly to the SCSI Subsystem Diagnostic. The differences include changes to the Disk Controller Diagnostics Menus and Tape Drive Diagnostics Menus and the addition of an ESP SCSI Test Menu.

Disk Controller Diagnostics Menus and Tape Drive Diagnostics Menus

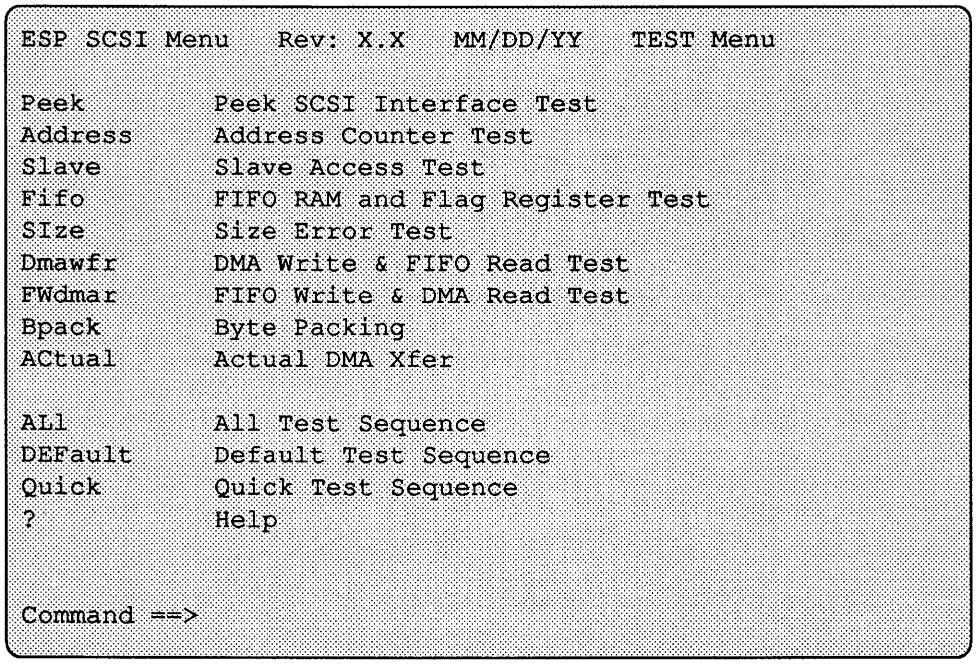
The options on the Disk Controller Diagnostics Menus and Tape Drive Diagnostics Menus are the same as those in the SCSI Subsystem Diagnostic, with two exceptions. The exceptions are these:

- The *Disconnect and Reconnect* option on the Disk Controller Diagnostics Command Menu is not available in the SPARCsystem 330 ESP SCSI Diagnostic.
- The *Write Until End of Tape* option on the Tape Drive Diagnostics Test Menu is not available in the SPARCsystem 330 ESP SCSI Diagnostic.

A new menu, the ESP SCSI Test Menu, is displayed on SPARCsystem 330 systems. The options available on this menu are described next.

ESP SCSI Test Menu

The ESP SCSI Test Menu provides access to the individual ESP tests:



```
ESP SCSI Menu   Rev: X.X   MM/DD/YY   TEST Menu

Peek        Peek SCSI Interface Test
Address     Address Counter Test
Slave       Slave Access Test
Fifo        FIFO RAM and Flag Register Test
SIze        Size Error Test
Dmawfr      DMA Write & FIFO Read Test
FWdmar      FIFO Write & DMA Read Test
Bpack       Byte Packing
ACtual      Actual DMA Xfer

ALl         All Test Sequence
DEFault     Default Test Sequence
Quick       Quick Test Sequence
?           Help


Command ==>
```

P

The *Peek SCSI Interface Test* option on the ESP SCSI Test Menu checks for the presence of the ESP chip and verifies that the CPU can access it.

A

The *Address Counter Test* option on the ESP SCSI Test Menu checks the DVMA Address Register and the ESP Count (low and high) Registers.

S

The *Slave Access Test* option on the ESP SCSI Test Menu simulates a slave access to the ESP controller. The data pattern written is read and compared.

F

The *FIFO RAM and Flag Register Test* option on the ESP SCSI Test Menu verifies the ESP FIFO RAM and FIFO Flag Registers. The test performs the following steps:

1. Writes a data pattern to FIFO RAM and checks to verify that the FIFO Flag Register increments.
2. Reads and compares the pattern read to the write pattern as the Flag Register decrements.

SI

The *Size Error Test* option on the ESP SCSI Test Menu verifies the function of the ESP when encountering a size error.

D

The *DMA Write and FIFO Read Test* option on the ESP SCSI Test Menu verifies the data path from main memory to FIFO RAM. With the ESP chip set to Target mode, a DMA write to FIFO is performed and the FIFO is read directly. This test verifies that the DMA path from system memory to FIFO is clear.

FW

The *FIFO Write and DMA Read Test* option on the ESP SCSI Test Menu verifies the data path from FIFO RAM to main memory. With the ESP chip set to Target mode, data is written to FIFO and a DMA read operation is performed on buffer memory. This test verifies that the DMA path from FIFO to system memory is clear.

B

The *Byte Packing Test* option on the ESP SCSI Test Menu tests the byte packing circuitry by loading four, three, two, and one bytes of data to FIFO and performing a DMA write operation to memory.

AC

The *Actual DMA Xfer Test* option on the ESP SCSI Test Menu verifies the DMA write operation by writing as many as four blocks (2048 bytes) of data to disk. The data written is read back and compared.

AL

The *All Test Sequence* option on the ESP SCSI Test Menu executes all of the tests on the menu in order.

DEF

The *Default Test Sequence* option on the ESP SCSI Test Menu executes all of the tests on the menu except the Actual DMA Xfer Test.

Q

The *Quick Test Sequence* option on the ESP SCSI Test Menu executes all of the tests on the menu except the Actual DMA Xfer Test.

?

The *Help* option on the ESP SCSI Test Menu displays brief Help messages describing each option on the menu.

30.6. Glossary

ESP	Extended SCSI Processor. The onboard SCSI bus controller chip.
FIFO	First In First Out RAM.
SCSI	Small Computer System Interface.

31

SCSI Subsystem Diagnostic

31

SCSI Subsystem Diagnostic

This diagnostic is associated with the SCSI Host Adaptor, Subsystem Disk Drive, Disk Controller and Tape Drive.

31.1. Introduction

Sun Microsystems supports several different SCSI Host Adaptors, Subsystem Disk Drives, disk controllers, and tape drives. Sun also supports embedded SCSI disk 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.

During loading time, the diagnostic determines 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

The embedded SCSI drives are:

1. CDC Wren IV 94171-344
2. CDC Wren IV 94161-155

The On-board SCSI3 host adaptor is used on Sun-4/100 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.

Embedded SCSI tape drives supported are:

1. Archive Viper 150 (QIC-120 and QIC-150 modes)
2. HP 88780 1/2-inch Front Load Tape *not available at this time*

31.2. 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.

31.3. 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 CPU board
2. A working keyboard and mouse
3. A working monitor
4. SCSI Subsystem Disk Controllers (Adaptec 4000, Emulex MD21 or embedded SCSI peripherals)
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.

31.4. 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 the `scsisub.exec` file and to start the test. Refer to Chapter 2 for further information on how to set up and run the test under the 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.*

31.5. 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.

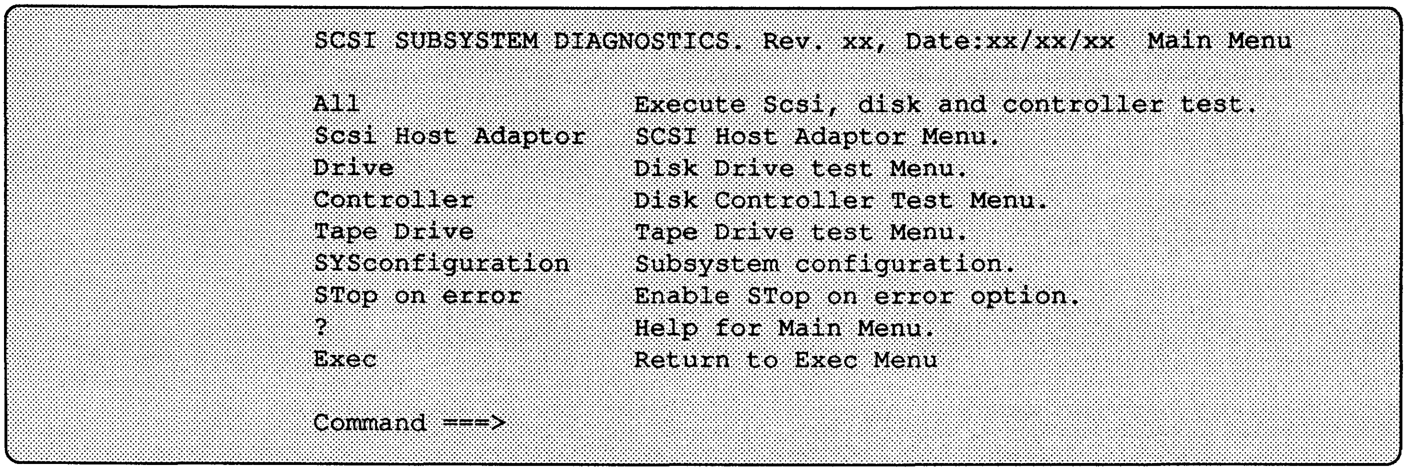
31.6. The User Interface

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 SUBSYSTEM 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:

A 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.

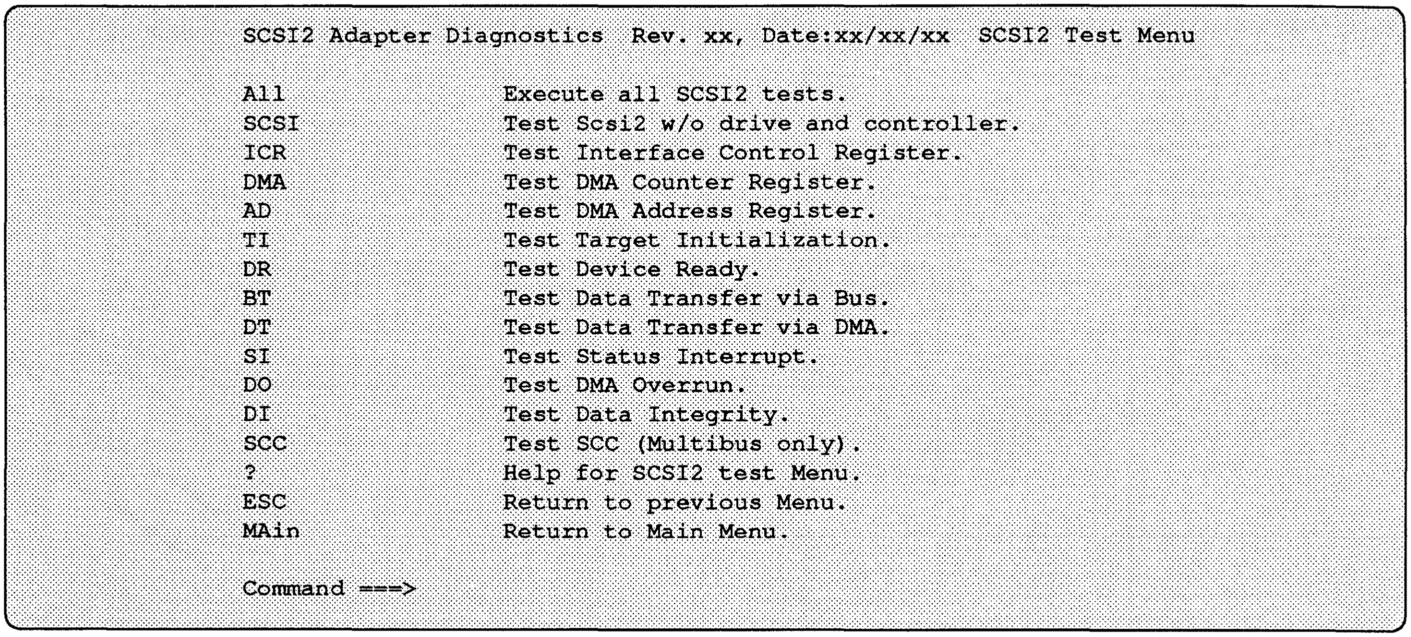
Exec
Entering E quits and returns to the Exec Main Menu.

SCSI Tests Menus

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 :

A 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:

A 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 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 :

A 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:

A When you execute this command, the program executes all the tests given in this menu.

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:

A 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:

A 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.

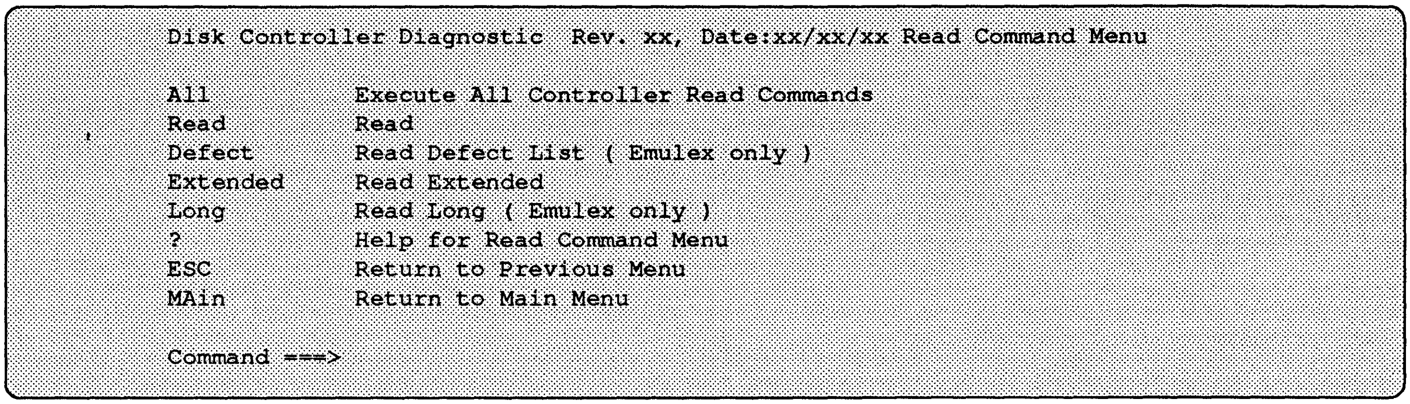
L This command requests that the controller perform a write operation of one data block and the six bytes of ECC information. The data and the six ECC bytes for the specified logical block are supplied by the host adapter during the Data Out phase. Only the Emulex controller supports this command.

? 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:

A 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.

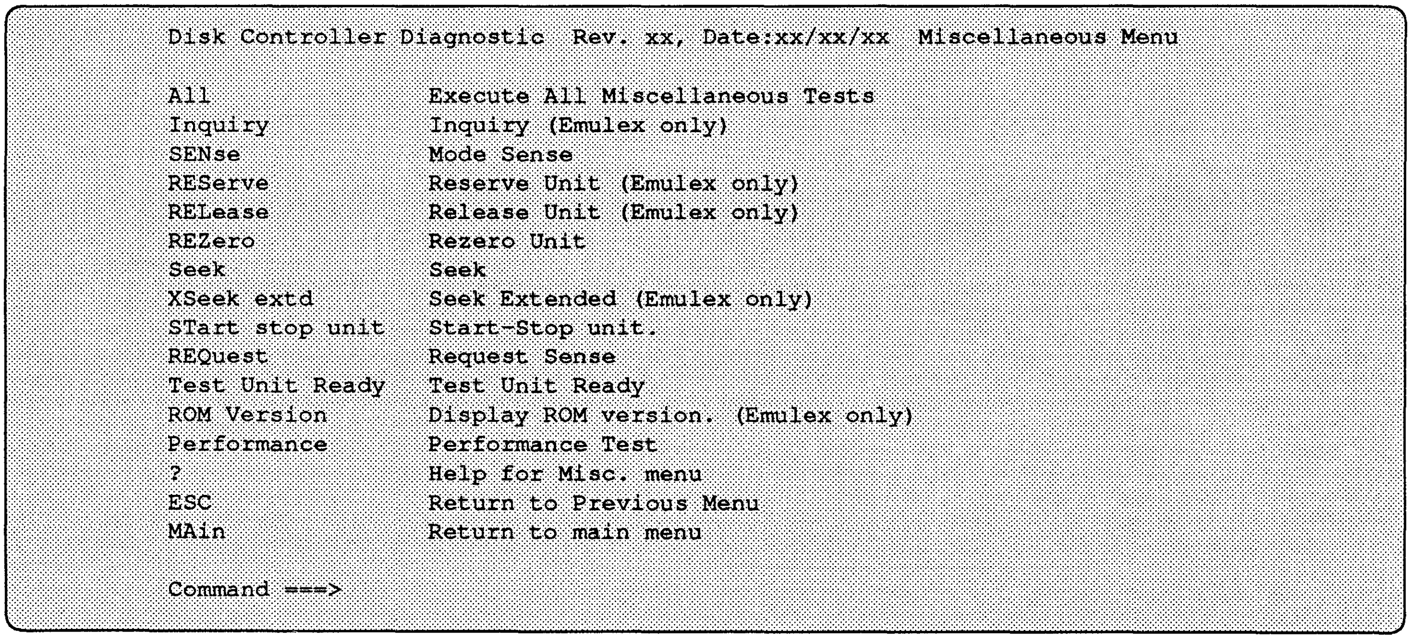
L This command requests that the controller perform a read operation of one data block and the six ECC bytes associated with that block. The data from the block and the ECC bytes are transferred to the host adapter during the Data In phase.

? 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:

A 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. xx, 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:

A 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.

A 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.

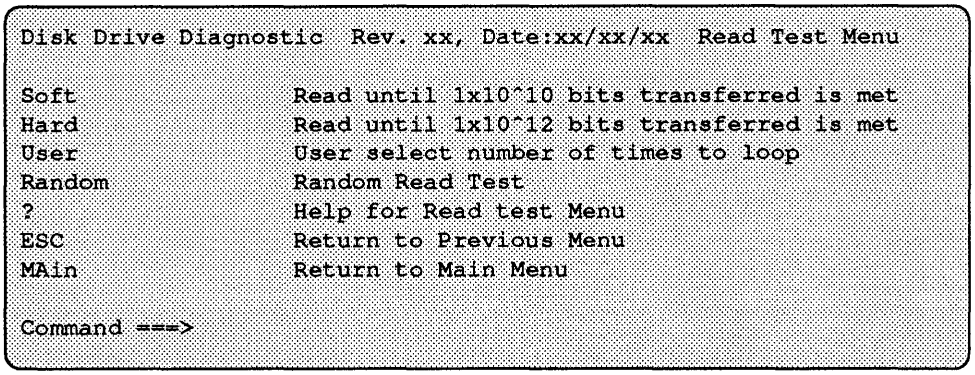
7A6E
This command writes the entire disk except the disk drive label and defect list block with the pattern 0x7A6E. This pattern is used to test the drives that are using RLL (Run Length Encode).

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:

A 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.

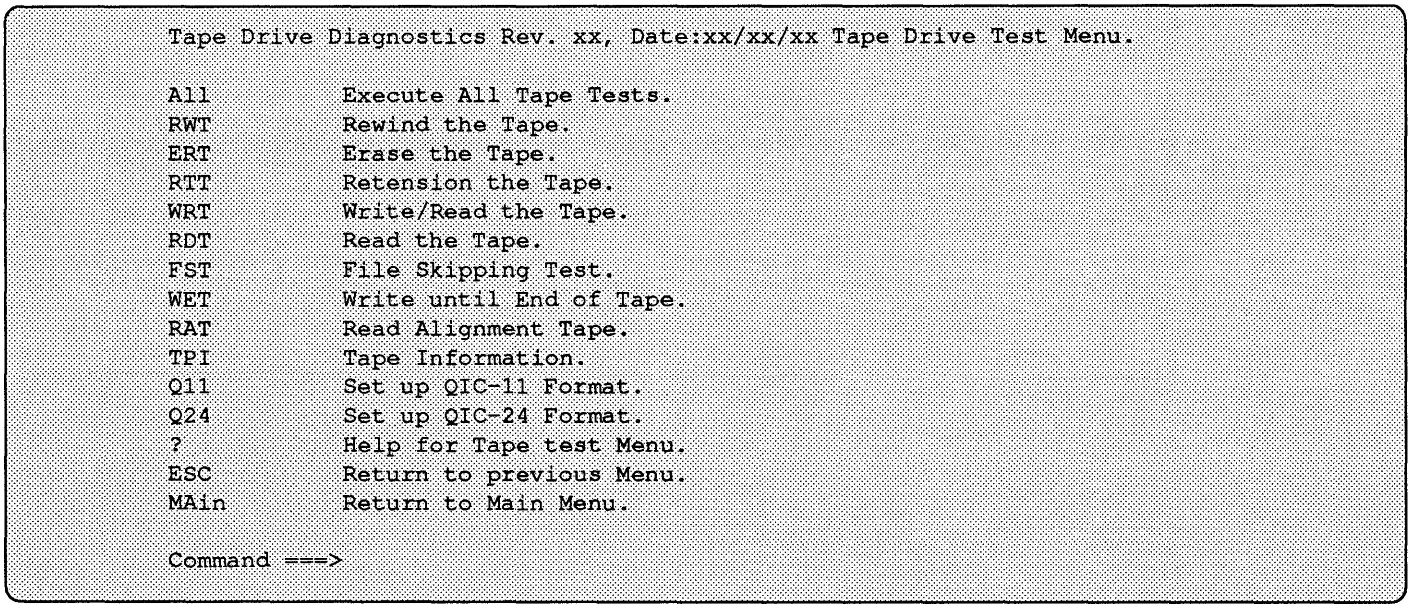
? This command displays information regarding usage of the commands in this menu.

MA This command exits the Seek Test menu and returns to the Main Menu.

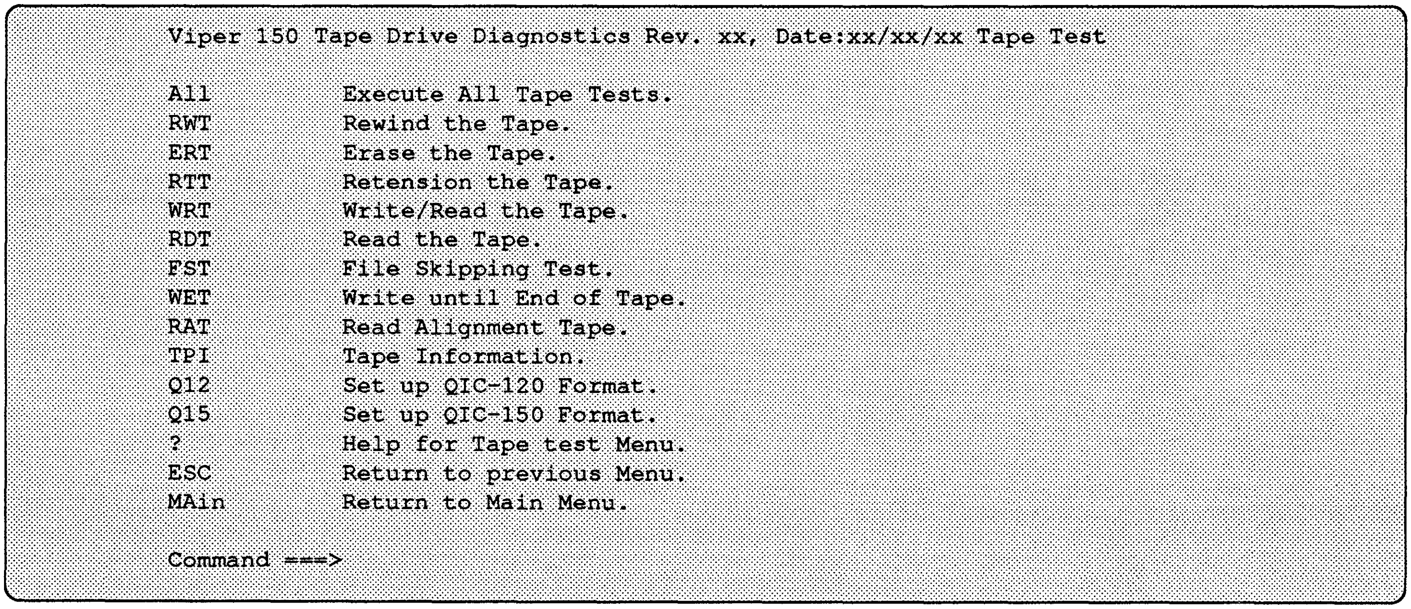
Tape Drive Tests Menu

Tape Drive menus differ slightly, depending on the type of drive under test. The menus shown on the following pages depict the standard SCSI tape drive menu, followed by the Viper 150 and HP88780 menus.

NOTE *Executing these tests when a tape is not inserted in the drive may result in error messages.*



```
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.
WRT         Write/Read the Tape.
RDT         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 ===>
```



```
Viper 150 Tape Drive Diagnostics Rev. xx, Date:xx/xx/xx Tape Test

All         Execute All Tape Tests.
RWT         Rewind the Tape.
ERT         Erase the Tape.
RTT         Retension the Tape.
WRT         Write/Read the Tape.
RDT         Read the Tape.
FST         File Skipping Test.
WET         Write until End of Tape.
RAT         Read Alignment Tape.
TPI         Tape Information.
Q12         Set up QIC-120 Format.
Q15         Set up QIC-150 Format.
?           Help for Tape test Menu.
ESC         Return to previous Menu.
MAin        Return to Main Menu.

Command ===>
```

```
HP88780 1/2" Tape Drive Diagnostics Rev. xx, Date:xx/xx/xx Tape Test

All         Execute All Tape Tests.
RWT         Rewind the Tape.
ERT         Erase the Tape.
WRT         Write/Read the Tape.
RDT         Read the Tape.
FST         File Skipping Test.
WET         Write until End of Tape.
TPI         Tape Information.
Q8          Set up 800 PE Format.
Q16         Set up 1600 PE Format.
Q62         Set up 6250 GCR Format.
QDC         Set up Data Compression 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 run-time 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 QIC24) 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 on.

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.

Q12

This command appears only when testing a Viper 150 tape drive and sets up QIC-120 tape format.

Q15

This command appears only when testing a Viper 150 tape drive and sets up QIC-150 tape format.

Q8 This command appears only when testing an HP 88780 1/2-inch tape drive, and sets up 800 PE tape format.

Q16

This command appears only when testing an HP 88780 1/2-inch tape drive, and sets up 1600 PE tape format.

Q62

This command appears only when testing an HP 88780 1/2-inch tape drive,

and sets up 6250 GCR tape format.

QDC

This command appears only when testing an HP 88780 1/2-inch tape drive, and sets up data compression format.

? 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.

31.7. 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.

31.8. 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.
```

31.9. 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.

31.10. 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.

32

FDC Diagnostic

32

FDC Diagnostic

32.1. General Description

The FDC (Floppy Disk Controller) is a chip with three registers that controls the action of the floppy disk drive and handles all data transfers between the floppy disk drive and the CPU board. The FDC Diagnostic verifies the functionality of the controller by providing two types of tests — one that allows visual verification of the mechanical motion of the floppy drive and another that tests data transfer functions.

32.2. What This Chapter Contains

Following an overview of the diagnostic and a list of required hardware, the FDC Diagnostic Main Menu and submenus are discussed. All menu options are described. The end of the chapter contains a list of error and informational messages and a glossary.

32.3. Overview of the Diagnostic

The FDC Diagnostic runs under the Exec and conforms to the interface standards of the Exec. In addition to individual tests, All, Default, and Quick test sequences are provided.

The All test sequence executes a continuous loop of tests that verify all FDC functions except eject, write protect, and motor operations that require your participation. The All test executes for approximately one hour or until an error is detected. This sequence affords a confidence level of at least 90%.

The Default test sequence executes a single pass of the data transfer tests. This sequence runs for approximately two minutes or until an error occurs and affords a confidence level greater than 85%.

The Quick test sequence executes a single pass of disk drive tests that verify the status pin signals and FDC circuitry path functions, such as motor operation, disk write-protect tab detection, recalibration, and seek operations. This test sequence runs for approximately one minute or until an error is detected.

The FDC Diagnostic generates and logs error messages for later retrieval. Online help is also provided.

32.4. Hardware Requirements

The following hardware is required to run the FDC Diagnostic:

□ A Sun-3/80 with MC68030 CPU

□ A Sun-3/80 power supply

- An 82072 Floppy Disk Controller
- A 3.5-inch floppy disk drive with cable
- Two high-density floppy disks, write enabled
- A monitor
- A keyboard
- A boot device (local disk, local tape, or remote disk over Ethernet)

32.5. User Interface

The user interface of the FDC Diagnostic adheres to the menu standards of the Exec. A Main Menu and submenus are provided. Each option may be selected from a menu by typing the letter or letters displayed in upper case in the column on the left side of the menu.

Two sets of submenus are provided. For both floppy disk drive and data transfer testing, a mode submenu allows you to execute tests in two ways — in a continuous sequence or individually, for specific debugging purposes. The individual tests are provided on the second submenu.

To return to the previous, higher menu level, press [ESC] then press [RETURN]. To exit the diagnostic and return to the Exec, press [ESC] when the Main Menu is displayed.

Additional parameters may be specified on the command line. The FDC Diagnostic supports the command syntax that allows you to execute a predefined script file. For more information on writing script files, see Chapter 2, "Using the SunDiagnostic Executive." The default pass count for all individual tests and sequences of tests is 1.

To display online Help for all the options on a menu, enter **?** on the command line.

32.6. Starting the Diagnostic

For information on starting the Exec, refer to Chapter 2, "Using the SunDiagnostic Executive." After you have started the Exec, choose the FDC Diagnostic from the Diagnostics Menu.

When the diagnostic starts, it checks for the presence of a floppy drive. If the drive is not detected, the following message is displayed:

```
Floppy Drive NOT responding to FDC controller!!!
```

If the drive is detected, the following message is displayed:

```
Floppy Drive detected by FFDC controller.
```

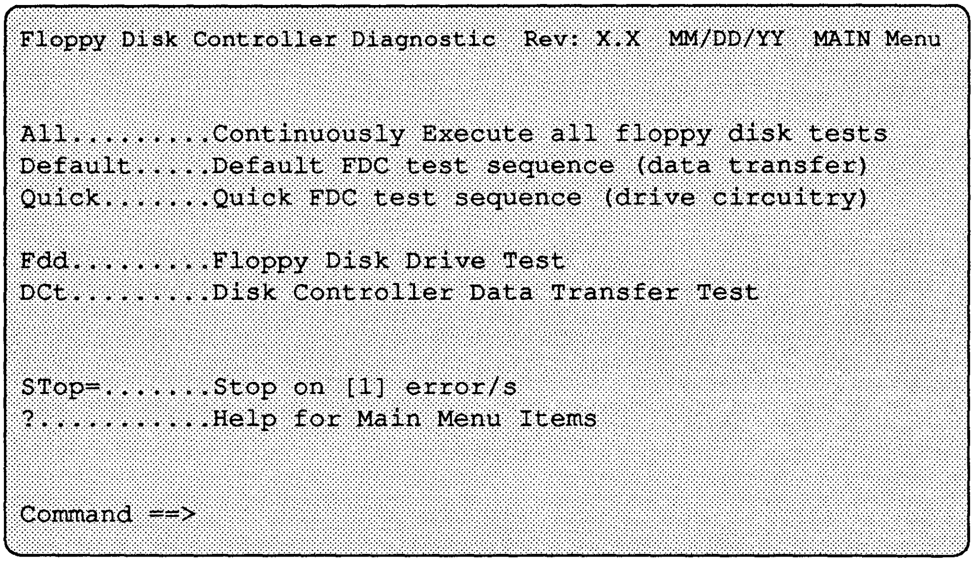
The diagnostic then displays the Main Menu.

32.7. The Diagnostic Menus

This section of the chapter provides a modular description of the FDC Diagnostic, beginning with the Main Menu and working down through the options available on each of the submenus. A list of messages generated by the diagnostic is given in the section entitled "Messages."

Main Menu

The FDC Main Menu, which displays when you start the FDC Diagnostic, provides access to the submenus of the individual tests:



```
Floppy Disk Controller Diagnostic  Rev: X.X  MM/DD/YY  MAIN Menu


All.........Continuously Execute all floppy disk tests
Default.....Default FDC test sequence (data transfer)
Quick.......Quick FDC test sequence (drive circuitry)

Fdd.........Floppy Disk Drive Test
DCt.........Disk Controller Data Transfer Test


STop=.......Stop on [1] error/s
?...........Help for Main Menu Items


Command ==>
```

Continuously Execute All Floppy Disk Tests

A

The *Continuously Execute All Floppy Disk Tests* option on the Main Menu executes all tests in the diagnostic that do not require your intervention. When the sequence completes, a summary of test results is displayed.

Default FDC Test Sequence

D

The *Default FDC Test Sequence* option on the Main Menu executes a single sequence of all of the data transfer tests. These tests do not require your intervention. When the sequence completes, a summary of test results is displayed.

Quick FDC Test Sequence

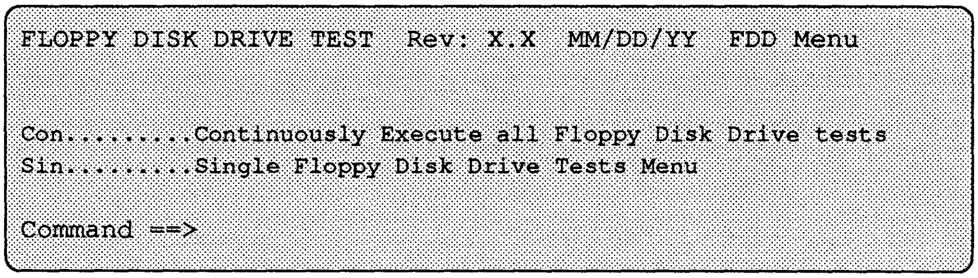
Q

The *Quick FDC Test Sequence* option on the Main Menu executes a single sequence of tests that check drive status and pin configuration. This sequence does not require your intervention. When the sequence completes, a summary of test results is displayed.

Floppy Disk Drive Test

F

When you choose **F** from the Main Menu, the Floppy Disk Drive Test Mode Menu is displayed:



```
FLOPPY DISK DRIVE TEST  Rev: X.X  MM/DD/YY  FDD Menu


Con.........Continuously Execute all Floppy Disk Drive tests
Sin.........Single Floppy Disk Drive Tests Menu

Command ==>
```

The options on this menu allow you to run a continuous sequence of disk drive tests or to access a submenu that provides individual tests for specific debugging purposes.

C

The *Continuously Execute All Floppy Disk Drive Tests* option on the Floppy Disk Drive Test Mode Menu executes a sequence of short versions of the single Floppy Disk Drive Tests. The first three tests in the sequence require your participation. At the end of the sequence, a summary of test results is displayed.

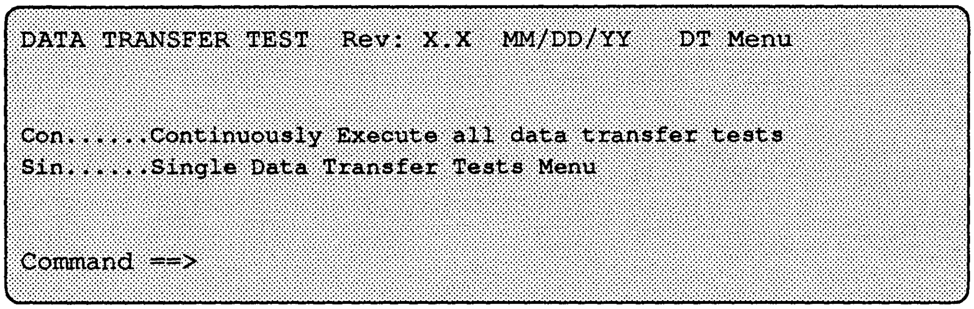
S

When you choose **S** from the Floppy Disk Drive Test Mode Menu, the Floppy Disk Drive Single Tests Menu is displayed. This menu provides access to individual tests that you can use for specific testing and debugging of the floppy disk drive. For a complete description of the options on this menu, see the section entitled "Floppy Disk Drive Single Tests Menu," later in this chapter.

Disk Controller Data Transfer Test

DC

When you choose **DC** from the Main Menu, the Disk Controller Data Transfer Test Mode Menu is displayed:



```
DATA TRANSFER TEST  Rev: X.X  MM/DD/YY   DT Menu


Con......Continuously Execute all data transfer tests
Sin......Single Data Transfer Tests Menu



Command ==>
```

The options on this menu allow you to run a continuous sequence of all data transfer tests or to access a submenu that provides individual tests for specific debugging purposes.

C

The *Continuously Execute All Data Transfer Tests* option on the Data Transfer Test Mode Menu executes a sequence of short versions of the Data Transfer Tests. At the end of the sequence, a summary of test results is displayed.

S

When you choose **S** from the Data Transfer Test Mode Menu, the Data Transfer Single Tests Menu is displayed. This menu provides access to individual tests that you can use for specific testing of the data transfer functions, such as formatting, reading, and writing disks. For a complete description of the options on this menu, see the section entitled “Data Transfer Single Tests Menu,” later in this chapter.

Stop on [1] Error/s

ST=

The *Stop on [1] Error/s* option on the Main Menu allows you to continue testing until the error count you specify is reached. The default value of this option is 1. The option affects all sequences of tests on all menus in the diagnostic.

Help

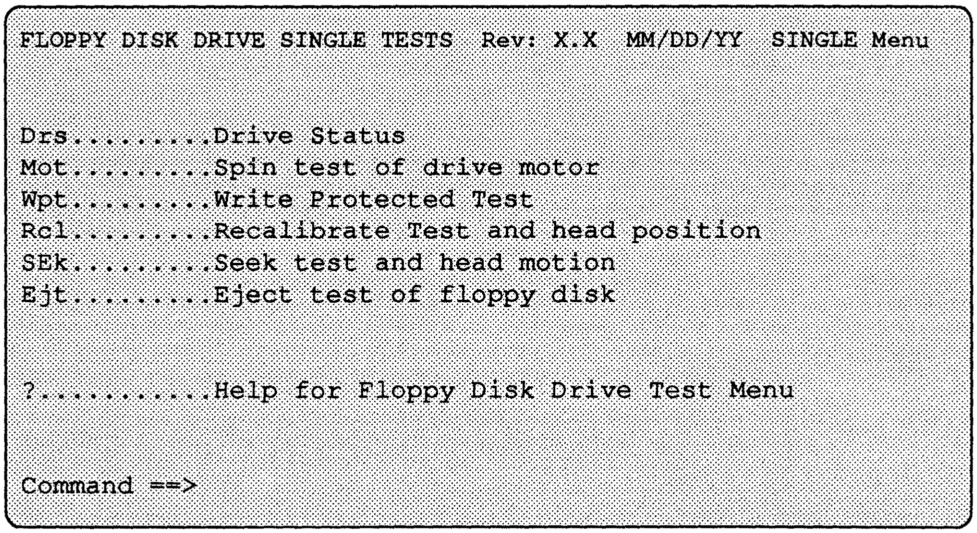
?

The *Help for Main Menu Items* option on the Main Menu displays help messages that describe each of the options on the menu.

Floppy Disk Drive Single Tests Menu

S

When you choose **S** from the Floppy Disk Drive Test Mode Menu, the Floppy Disk Drive Single Tests Menu is displayed:



```
FLOPPY DISK DRIVE SINGLE TESTS  Rev: X.X  MM/DD/YY  SINGLE Menu


Drs.........Drive Status
Mot.........Spin test of drive motor
Wpt.........Write Protected Test
Rcl.........Recalibrate Test and head position
SEk.........Seek test and head motion
Ejt.........Eject test of floppy disk


?...........Help for Floppy Disk Drive Test Menu


Command ==>
```

The options on this menu allow you to execute specific tests of individual disk drive functions.

D

The *Drive Status* option on the Floppy Disk Drive Single Tests Menu displays current disk drive status information. It shows the state of all pins that connect the FDC to the floppy disk drive. The state of any pin represented by `1` is high, by `0` is low.

This information is useful for debugging the connector and the associated response of the FDC when a pin state changes. For example, the state of the write protected pin (WP) should change if the disk write-protect tab is changed from open to closed.

M

The *Spin Test of Drive Motor* option on the Floppy Disk Drive Single Tests Menu prompts you to initiate a motor test. When you respond, the motor is turned on and off. To verify the response of the motor, it is recommended that you turn the floppy disk drive upside down to observe the spinning wheel inside the drive.

W

The *Write Protected Test* option on the Floppy Disk Drive Single Tests Menu tests whether the FDC can correctly interpret a write-protected signal. Verify the position of the write-protect tab before running this test then compare the test results displayed after the test executes. A low pin state means that the disk is not write protected, and a high state means that the disk is write protected.

R

The *Recalibrate Test and Head Position* option on the Floppy Disk Drive Single Tests Menu tests the ability of the drive to respond to a recalibration command. It then displays a message indicating whether the test passed or failed. In case of failure, the TRK0 pin should be checked as a probable cause.

SE

The *Seek Test and Head Motion* option on the Floppy Disk Drive Single Tests Menu moves the read/write head to the center of the disk then checks to verify its position.

E

The *Eject Test of Floppy Disk* option on the Floppy Disk Drive Single Tests Menu prompts you to initiate a disk ejection, then it ejects the disk from the drive.

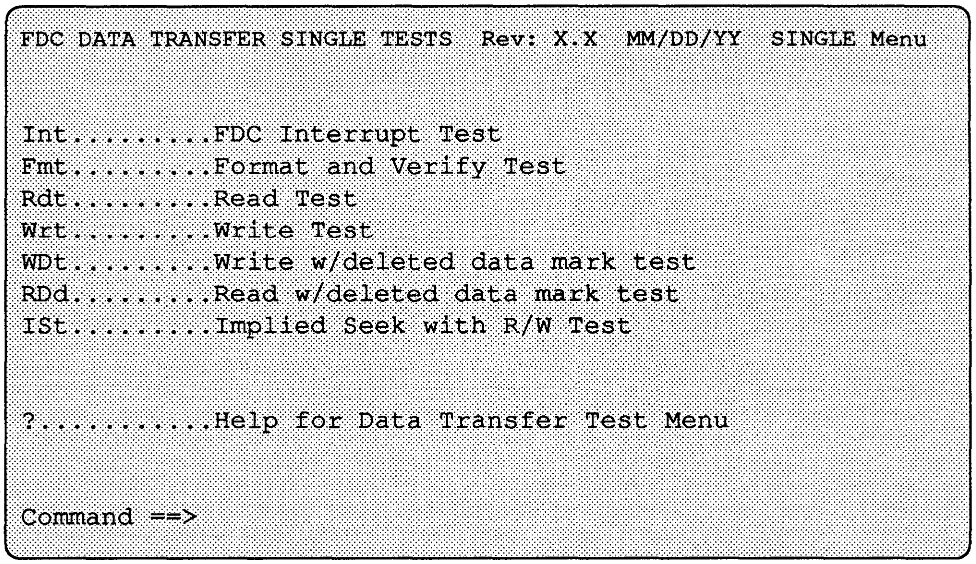
?

The *Help for Floppy Disk Test Menu* option on the Floppy Disk Drive Single Tests Menu displays help messages that describe each of the options on the menu.

Data Transfer Single Tests Menu

S

When you choose **S** from the Data Transfer Test Mode Menu, the Data Transfer Single Tests Menu is displayed:



```
FDC DATA TRANSFER SINGLE TESTS  Rev: X.X  MM/DD/YY  SINGLE Menu


Int.........FDC Interrupt Test
Fmt.........Format and Verify Test
Rdt.........Read Test
Wrt.........Write Test
WDt.........Write w/deleted data mark test
RDd.........Read w/deleted data mark test
ISt.........Implied Seek with R/W Test


?...........Help for Data Transfer Test Menu


Command ==>
```

The options on this menu allow you to perform individual tests that thoroughly exercise the data transfer functions of the FDC.

I

The *FDC Interrupt Test* option on the Data Transfer Single Tests Menu generates an interrupt by issuing a recalibrate command. It then checks to see if the interrupt was acknowledged by the CPU.

F

The *Format and Verify Test* option on the Data Transfer Single Tests Menu formats the number of tracks that you specify. The test displays the ID status of the last track formatted then verifies all tracks, from last to first.

R

The *Read Test* option on the Data Transfer Single Tests Menu reads the number of sectors that you specify. You are requested to supply the track and sector numbers of the starting sector and the number of sectors to read. The data read is displayed in hexadecimal, 16 bytes per line, in two segments for each sector. The first segment contains the data from bytes 0–255 of the sector, the second, from bytes 256–512.

W

The *Write Test* option on the Data Transfer Single Tests Menu writes the number of sectors that you specify with the pattern that you specify. You are requested to supply the track and sector numbers of the sector from which the write operation starts and the number of sectors to write. You may then select one of three data patterns to be written. When the write operation is complete, you can choose to read the written sectors.

WD

The *Write w/Deleted Data Mark Test* option on the Data Transfer Single Tests Menu writes the number of sectors that you specify with the pattern that you specify, adding the deleted data mark to the ID field on the disk. You are requested to supply the track and sector numbers of the sector from which the write operation starts and the number of sectors to write. You may then select one of three data patterns to be written.

RD

The *Read w/Deleted Data Mark Test* option on the Data Transfer Single Tests Menu reads sectors that were written with a deleted data mark. You must specify the range of sectors to read. The entire range that you specify should have been written to previously with the Write w/Deleted Data Mark Test option.

IS

The *Implied Seek with R/W Test* option on the Data Transfer Single Tests Menu performs read and write operations with the implied seek function turned on. You specify whether a read or a write operation is to be performed. If you choose a write operation, you are requested to choose a data pattern to be written. The test then performs the operation you specified. If both a write and a read are performed on the same area, a comparison is performed to verify that the data read matches the data written. At the end of the test, the implied seek is turned off.

?

The *Help for Data Transfer Test Menu* option on the Data Transfer Single Tests Menu displays help messages that describe each of the options on the menu.

32.8. Messages

The error and informational messages generated by the FDC Diagnostic tests are listed in this section. Any error messages generated are both displayed and written to the error log file.

If an error occurs during a read/verify operation, the following message is displayed:

```
Missing address mark
```

If an error occurs while writing to a sector, the following message is displayed:

```
Not writable, WP became 1 while executing
```

If an error occurs during a verify/seek operation, the following message is displayed:

```
No data, can't find sector
```

If an error occurs during a format/seek operation, the following message is displayed:

```
FIFO did not receive service in time
```

If a data error is detected during a read/verify operation, the following message is displayed:

```
CRC data error
```

If an error occurs during a read/seek operation, the following message is displayed:

```
Cannot detect address mark
```

If an error occurs during a verify/read operation, the following message is displayed:

```
Bad track
```

If an error occurs during a seek operation, the following message is displayed:

```
Wrong track
```

When a deleted data mark is detected during a read/verify operation, the following message is displayed:

```
Encountered deleted data mark
```

When an attempt is made to format or write to a disk that is write protected, the following message is displayed:

```
Write protected disk
```

32.9. Glossary

Deleted Data Mark
A setting in the ID field on a disk that specifies when a sector is marked for deletion.

FDC
Floppy Disk Controller.

Write Protect
The process by which the floppy disk drive detects the state of the disk's write-protect tab. The position of the write-protect tab determines whether or not a disk can be written to.

33

Sun SMD Diagnostic

33

Sun SMD Diagnostic

33.1. 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 33-1 *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 33-2 *Disk Controller Boards*

Xylogics	450	Multibus
Xylogics	451	Multibus
Xylogics	7053	VME

33.2. 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.**

33.3. 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 cnumberdnumber=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 cany numberdany 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.

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.**

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 whether 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.

33.4. 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 Disgnostic    REV 1 xx/xx/xx  Main Menu

Controller         Controller Tests Menu
Drive              Drive Tests Menu
All                All Test Sequence
Burnin             Burnin Controller tests
Quick              Quick Test Sequence
DEfault            Default Test Sequence
Utilities          Utilities Menu
Parameters         Enter Configuration Parameters

?                  Print the command syntax


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

DE*fault*

When you enter **de** for the default tests, the tests described for the `All` command are executed.

PAR*ameters*

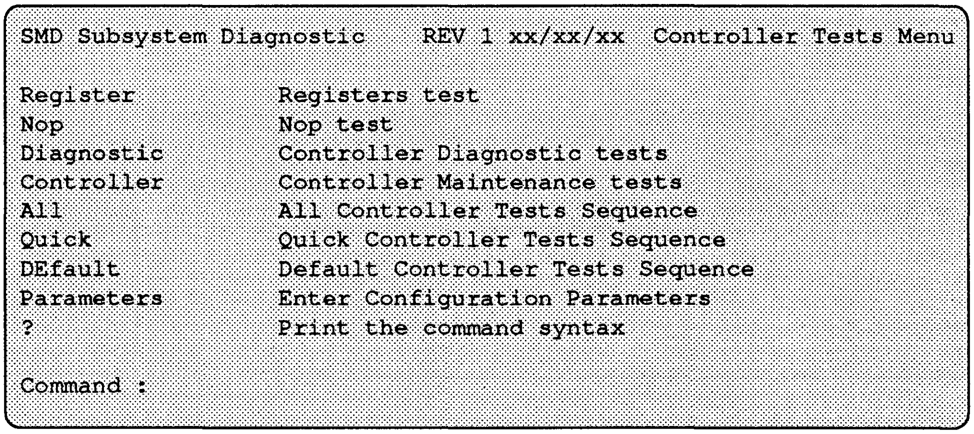
This command allows you to enter controller parameters. Enter **par** and the desired change, such as `CTO=xy451` or `C0D0=Fuj2333.`

?

The *Help Command* displays the syntax of all the commands. Enter **?** to invoke Help.

33.5. 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 Diagnostic    REV 1 xx/xx/xx  Controller Tests Menu

Register            Registers test
Nop                 Nop test
Diagnostic          Controller Diagnostic tests
Controller          Controller Maintenance tests
All                 All Controller Tests Sequence
Quick               Quick Controller Tests Sequence
DEfault             Default Controller Tests Sequence
Parameters          Enter Configuration Parameters
?                   Print the command syntax

Command :
```

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.

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.

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.

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.

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.

The *test* parameter is the test number, as follows:

Table 33-3 *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.

q The *quick* test sequence is the same as that described for the `All` command.

DE

The *default* command is the same as that described for the `All` command.

P*arameters*

This command allows you to enter controller parameters. Enter **p** and the desired change, such as `CTO=xy451` or `CODO=Fuj2333`.

?

The *Help* command displays the syntax of each command in the menu. Enter **?** to get help with the commands.

33.6. Drive Tests Menu

These tests check the SMD drive interface and drive operations. The menu looks something like this:

```
SMD Subsystem Diagnostic   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
Ecc              ECC Test
ALl              All Drive Tests Sequence
Quick            Quick Drive Tests Sequence
DEfault          Default Drive Tests Sequence
PARameters       Enter Configuration Parameters


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 the 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 33-4 *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=1*, 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, followed by a seek 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).

pat

The *Pattern test* writes and reads different patterns to the SMD drive. The test uses the patterns in the table below:

Table 33-5 *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.

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 argument is supplied, the test runs once.

Quick
The quick test sequence is the same as that described for the `All` command.

DE*fault*

The Default test sequence is the same as that described for the `All` command.

P*arameters*

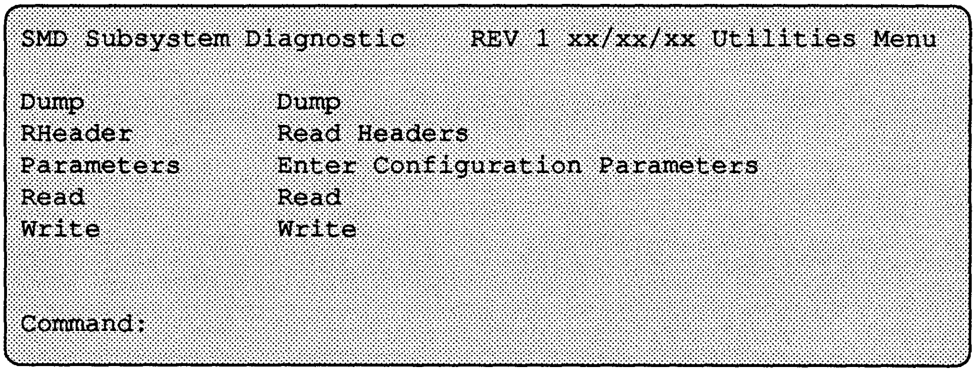
This command allows you to enter controller parameters. Enter **p** and the desired change, such as `CTO=xy451` or `COD0=Fuj2333`.

?

The *Help* command displays the syntax of each command. Enter **?** to get help with command syntax.

33.7. Utilities Menu

The Utilities Menu for an SMD drive will look something like this:



```
SMD Subsystem Diagnostic    REV 1 xx/xx/xx Utilities Menu

Dump             Dump
RHeader          Read Headers
Parameters       Enter Configuration Parameters
Read             Read
Write            Write


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.

P **ct***number***=** **c***number***d***number***=** *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. *number* should be replaced with the appropriate number. The defaults are:

```
ctl=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
33-5 )
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.

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 on the previous page.

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 below.

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.**

33.8. 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 33-6 *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.

33.9. 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).
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.
```

Program Errors, Continued

```
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]/[xx].
074     Bus 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].
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.
```

Program Errors, Continued

```
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.
```

33.10. 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.

34

1/2-Inch Tape Diagnostic

34

1/2-Inch Tape Diagnostic

34.1. General Description

The 1/2-inch Tape Subsystem Diagnostic tests all of the 1/2-inch tape transports currently supported by Sun, and supports the Xylogics 472 Tape Controller Board. This diagnostic does not support the Tapemaster tape controller board.

34.2. 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-inch tape controller (Xylogics 472).
- □ A 1/2-inch tape transport (Fujitsu M2444 or CDC 92181).
- □ A boot device (local disk, local tape or remote disk through Ethernet).
- □ A 1/2-inch 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.*

34.3. 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 ctnumber=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 cnumberdnumber=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.

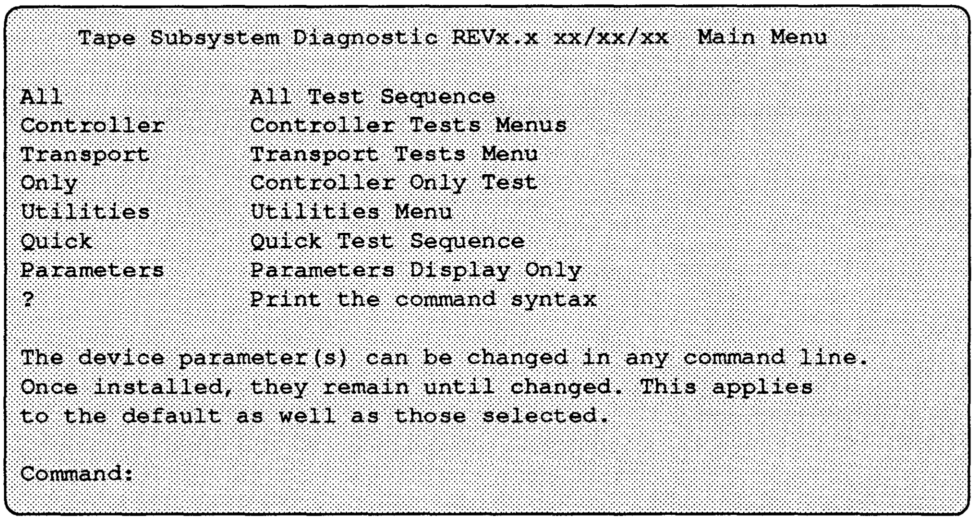
34.4. Menus

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:



```
    Tape Subsystem Diagnostic REVx.x xx/xx/xx  Main Menu

All             All Test Sequence
Controller      Controller Tests Menus
Transport       Transport Tests Menu
Only            Controller Only Test
Utilities       Utilities Menu
Quick           Quick Test Sequence
Parameters      Parameters Display Only
?               Print the command syntax

The device parameter(s) can be changed in any command line.
Once installed, they remain until changed. This applies
to the default as well as those selected.

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. Refer to the table that follows the Main Menu example for a list of valid *Mode=* entries.

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.**

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=*number* **c***number***d***number***=**

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:

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`

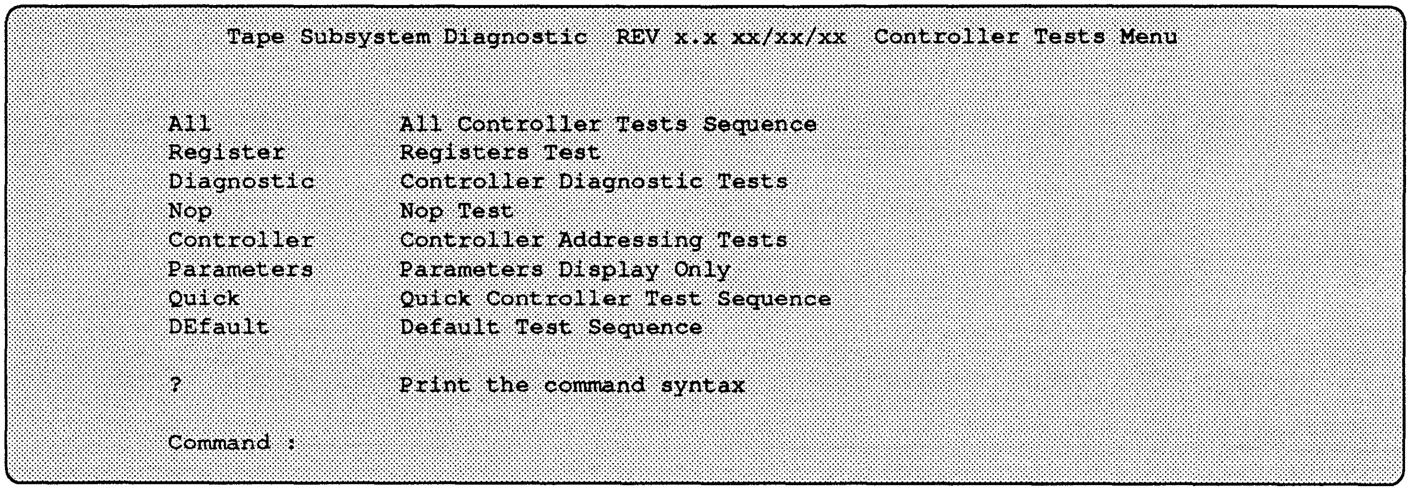
as drive types.

?

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:



```
        Tape Subsystem Diagnostic  REV x.x xx/xx/xx  Controller Tests Menu


   All              All Controller Tests Sequence
   Register         Registers Test
   Diagnostic       Controller Diagnostic Tests
   Nop              Nop Test
   Controller       Controller Addressing Tests
   Parameters       Parameters Display Only
   Quick            Quick Controller Test Sequence
   DEfault          Default Test Sequence

   ?                Print the command syntax

   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.

d *PASs=*

The *Controller diagnostic tests* command executes the controller's on-board diagnostics — if they are enabled.

n *PASs=*

The *Nop test* command makes the controller read the I/O process block (IOPB) and mark it complete.

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-inch transport is shown below:

```
     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
?                Print the command syntax

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=*

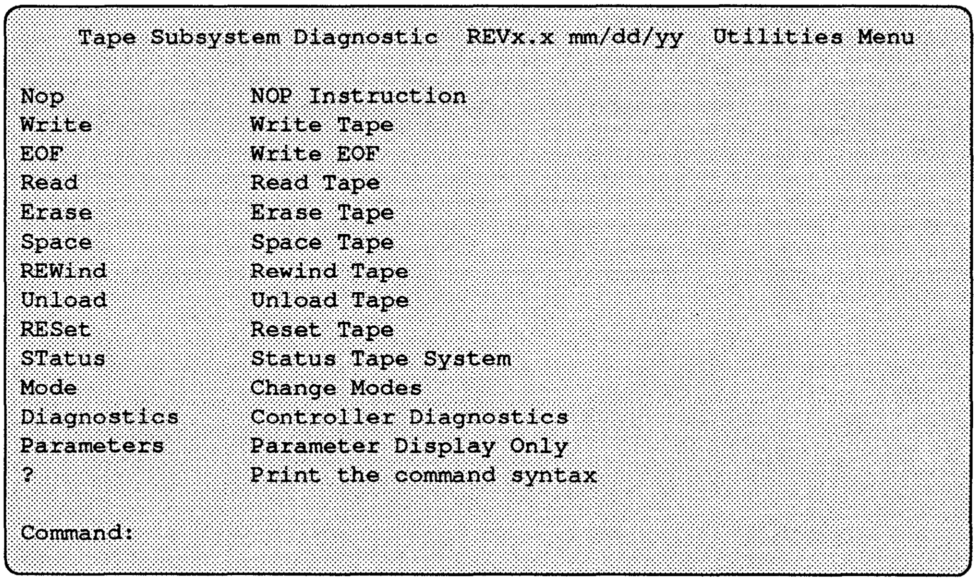
The *Quick test* command performs all of the tests specified by `All` command, but executes them in a shorter time period. You can enter the number of times the Quick test is executed.

?

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-inch tape subsystem.



```
    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 System
Mode            Change Modes
Diagnostics     Controller Diagnostics
Parameters      Parameter Display Only
?               Print the command syntax

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 run 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.

34.5. 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 34-1 *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 34-1 *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 'ctl' 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 34-2 *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.

34.6. 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-inch 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.

35

Sun Video Diagnostic

35

Sun Video Diagnostic

35.1. 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 the Color 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 for Sun-3 and Sun-4 products have the following characteristics:

- Sun-3/460 and Sun-3/80 (with Color frame buffer) — has eight color planes, an overlay plane, and an enable plane. The Overlay plane is 128 Kbytes long, beginning at 0x50400000; the enable plane is 128 Kbytes long, beginning at 0x50600000; and the color memory is 1 Mbyte long, beginning at 0x50800000.
- Sun-3/460 and Sun-3/80 (with Monochrome frame buffer) — has a monochrome frame buffer beginning at 0x504000000 and extending to 0x505FFFFF. Only 256 Kbytes are required for high resolution (1600 x 1280) and 128 Kbytes for standard resolution (1152 x 900).
- Sun-3/460 and Sun-3/80 (with P4 frame buffers) — has a register containing an ID number for the frame buffer type, video enable information, and video interrupt information.
- Sun-4/2xx — has a single monochrome frame buffer on the CPU board that is 256 Kbytes long (high resolution), starting at 0xFD000000.

- Sun-4/110 — has eight color planes, an overlay plane, and an enable plane. The Overlay plane is 128 Kbytes long, beginning at 0xfb400000; the enable plane is 128 Kbytes long, beginning at 0xfb600000; and the color memory is 1 Mbyte long, beginning at 0xfb800000.

35.2. Menus

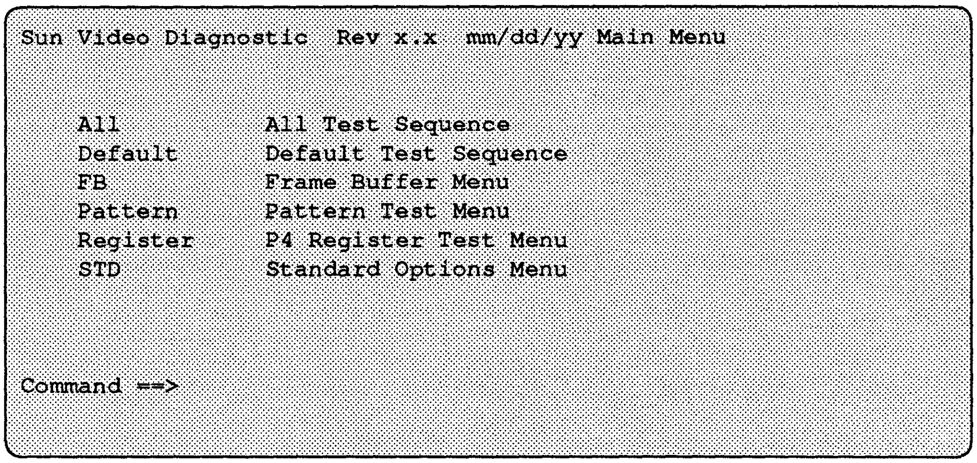
When testing any Sun-4 system other than the Sun-4/110, the Frame Buffer Menu comes up in place of the Main Menu, and represents the only test choices for those systems. The Main Menu for the Sun-4/110 includes a color pattern test menu, in addition to the Frame Buffer Menu.

The Main Menu for the Sun-3/400 or Sun-3/80 includes a color pattern test menu, 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.

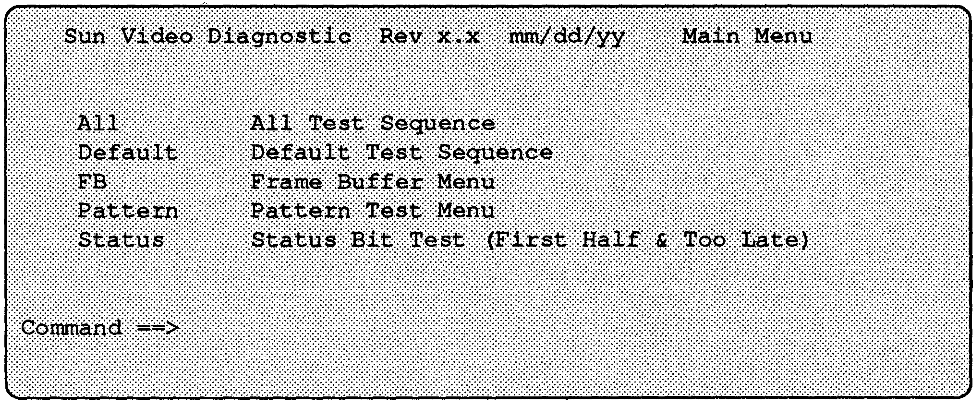
Monochrome frame buffer testing is used in most Sun-4s, and the color frame buffer testing is used for the combination of color and monochrome type frame buffers in the Sun-4/110 or Sun-3/400.

Here is an example of the Main Menu for the Sun-3/400:



```
Sun Video Diagnostic  Rev x.x  mm/dd/yy Main Menu


    All          All Test Sequence
    Default      Default Test Sequence
    FB           Frame Buffer Menu
    Pattern      Pattern Test Menu
    Register     P4 Register Test Menu
    STD          Standard Options Menu




Command ==>
```

Here is an example of the Main Menu for Sun-4/110:



```
   Sun Video Diagnostic  Rev x.x  mm/dd/yy    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 Main Menu choices:

All

The *All Test Sequence* runs the frame buffer tests automatically. This command runs the following tests in sequence:

```
AD ; 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
```

As they run, the tests record any errors in the error log and print status, information, and 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 status, information, and error messages on the screen. This is default test sequence:

```
AD ; 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
```

FB 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.

R

The *Register* command displays the P4 Register Test Menu for the Sun-3/80 or Sun-3/4xx systems.

STD

The *Standard Options Menu* command displays the Standard Options Menu. This menu is described later in the chapter.

35.3. Frame Buffer Menu

This test menu may be brought up from the Sun-4/110 or Sun-3/400 Main Menu. It automatically comes up *instead of the main menu* when testing any other Sun-4 system. The menu looks something like this:

```
Sun Video Diagnostic  Revx.x  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
    STD            Standard Option Menu

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:

```
AD ; C ; NTA ; U ; R
```

As they run, the tests record any errors they encounter in the error log and print status, information, and error messages to the screen.

Default

The *Default Test Sequence* runs a set of the frame buffer tests automatically. This command sequentially runs every test in the diagnostic, recording any errors it encounters in the error log, and printing error messages to the screen. The following default test sequence is for all Sun-3/400 and Sun-4 systems, except the Sun-4/110:

```
A ; C ; NTA ; U ; R ; frame buffer RAM tests
```

This default test sequence is for Sun-4/110 only:

```
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
```

AD *Offset= Datamode= Pass=*

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 default offset is 0, at the beginning of each frame buffer under test. The size of the monochrome plane is 0x20000 (128K Bytes) for standard resolution monochrome and color frame buffers and 0x40000 (256K Bytes) for high resolution monochrome frame buffers.

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` *Offset= Pattern= Datamode= Pass=*

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.

Defaults are:

Default Offset=0; beginning of each Frame Buffer under test.
Default pattern=0xa5; values 0x00 - 0xfffffff are acceptable.
Default datamode=2; 0=byte, 1=word 2=long
Default pass=1.

`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 35-1 *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

Default Offset=0x00, at the beginning of each Frame Buffer under test.
Default Pass=1.

`Random` *Offset= Datamode= Seed=*

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.

`Unique` *Offset= Datamode= Seed=*

The *Uniqueness Pattern Test* checks for address uniqueness. It runs in byte, word, or long word mode. 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.

`STD`

The *standard option menu* selection displays the standard options menu.

Pattern Menu

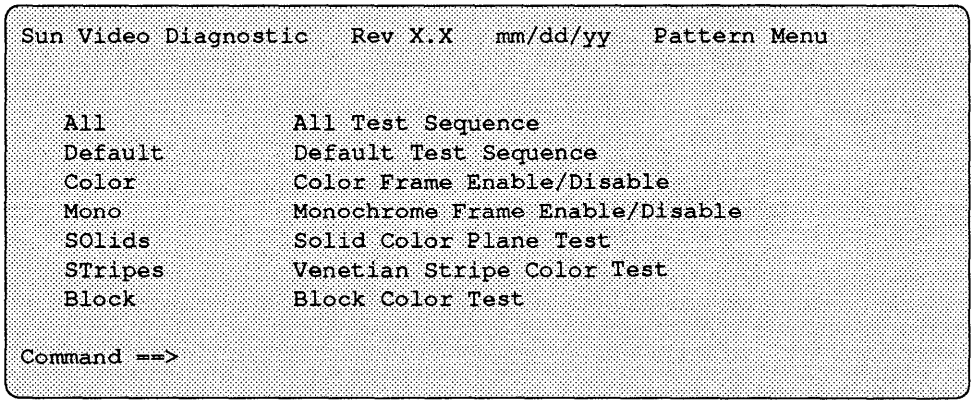
The main menu presented when a Sun-4/110 or Sun-3/400 is tested provides a `Pattern` menu choice. These patterns aid in debugging the hardware, using the video frame buffer RAM. Software has no access to this circuitry, so you must watch for failures. The `Mono` and `Color` tests are included in this menu to check the ability 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 table below lists the color parameters for this test. Using a value of 8 runs the test in each color.

Table 35-2 *Color Values*

Value	*Color*
0	black
1	red
2	green
3	blue
4	yellow
5	cyan
6	magenta
7	white
8	all



```
Sun Video Diagnostic   Rev X.X   mm/dd/yy   Pattern Menu


   All              All Test Sequence
   Default          Default Test Sequence
   Color            Color Frame Enable/Disable
   Mono             Monochrome Frame Enable/Disable
   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 ; M ; 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.

```
C ; M ; SO ; ST ; B
```

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.

Mono

The *Monochrome Frame Enable* command enables the monochrome frame buffer; the enable plane is filled with 1's.

Solid *color*

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.

The default color = 8. (Refer to the color/number chart in the previous section.)

Stripe *Color Color2*

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.

The defaults are:

Default Color=8.
Default Color2=1.

Refer to the color chart in the previous section.

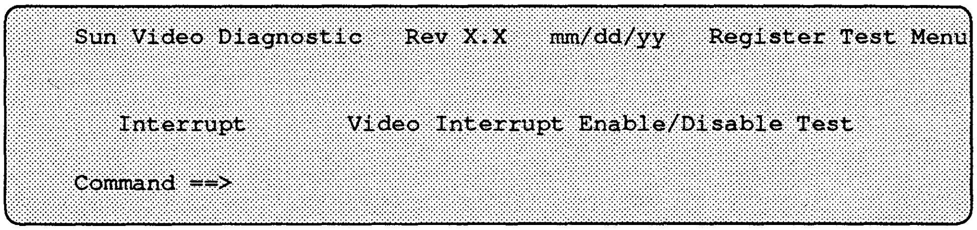
Block *color*

The *Block* 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.

Defaults are:

Default color=8.

Refer to the color chart in the previous section.



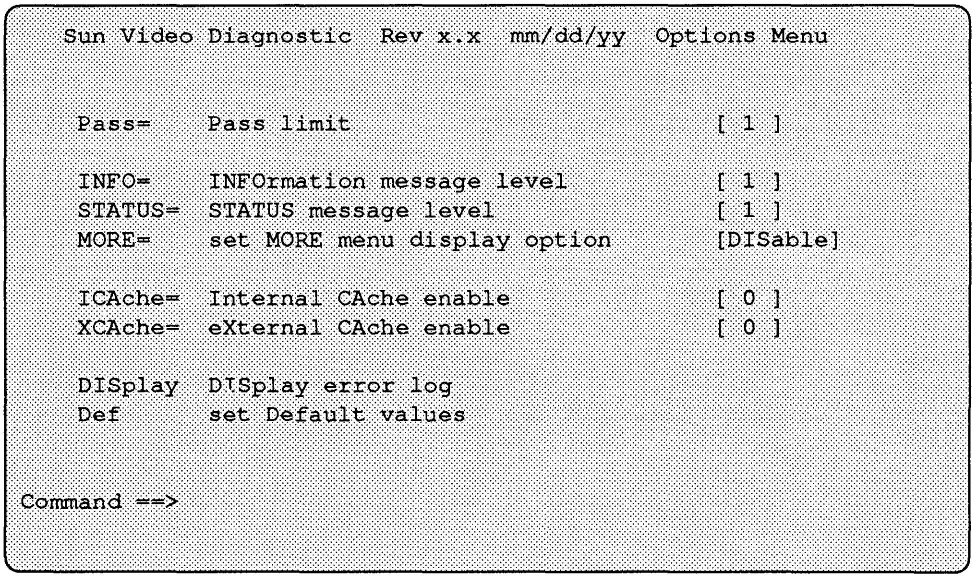
```
Sun Video Diagnostic   Rev X.X   mm/dd/yy   Register Test Menu


   Interrupt      Video Interrupt Enable/Disable Test

Command ==>
```

I

The *Interrupt* test disables the video interrupts and then installs an exception handler that indicates whether a video interrupt has been detected. The interrupt test verifies that no video interrupts are occurring while the video interrupts are disabled. Video interrupts are enabled and the test verifies that video interrupts are being received. It then disables the video interrupts, removes that exception handler, and restores the P4 and interrupt registers to their original states.

35.4. Standard Options Menu

This section describes the Standard Options Menu of the Sun Video Diagnostic.



```
   Sun Video Diagnostic  Rev x.x  mm/dd/yy  Options Menu


    Pass=    Pass limit                           [ 1 ]

    INFO=    INFOrmation message level            [ 1 ]
    STATUS=  STATUS message level                 [ 1 ]
    MORE=    set MORE menu display option         [DISable]

    ICAche=  Internal CAche enable                [ 0 ]
    XCAche=  eXternal CAche enable                [ 0 ]

    DISplay  DISplay error log
    Def      set Default values


Command ==>
```

Following are descriptions for the Options Menu:

P

Set the *Pass* limit to the number of times the test should be run. The default is one.

INFO

Option *info* sets the program information level to the value entered. The range is 0-3 and the default value is one.

STATUS

The *Status* option sets the program status level to the value entered. The range is 0-3, and the default value is one.

MORE

Entering *more* enables or disables more. The default is disabled.

ICA

Use the *ica* option to enable or disable the internal instruction cache and/or data cache The default is enable the instruction cache.

XCA

Use the *xca* option to enable or disable the external central cache and/or I/O cache. The default is enable both caches.

DIS

This option displays the list of errors recorded in the error log.

Default

The *default* option resets variables listed on the option menu to their default values.

35.5. Glossary

□ **Color Map** - 3 separate arrays, 256 bytes long, 8 bits wide, one each for Red, Green, and Blue, beginning at addresses 0xfb800000 on a Sun-4/110 and address 0x50200000 on a Sun-3/400. The color map is located on the board in the DAC s and occupies Type 1 space.

□ **Color Plane —** 1 Megabyte long frame buffer, eight (8) bits per pixel, located at address 0xfb800000 on a Sun-4/110 and at address 0x50200000 on a Sun-3/400. Used to index the color map locations.

□ **Frame Buffer —** Refers to the Memory (RAM) that the video circuitry uses to display the entire screen.

□ **Monochrome Plane —** The black and white or grayscale frame buffer.

□ **Enable Plane —** 128 Kbyte long Frame Buffer, Located at address 0xfb600000 on a Sun-4/110 and at address 0x50600000 on a Sun-3/400.

36

Sun Video Monitor Diagnostic

36

Sun Video Monitor Diagnostic

36.1. 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 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-4/2xx 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 Sun color or monochrome monitor display.*

The Sun Video Monitor Diagnostic is a set of patterns designed to allow the quick and easy test and setup of video monitors for Sun-3 and Sun-4 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.

36.2. Hardware Requirements

1. A Sun-3 or Sun-4 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.

36.3. 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.

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 Esc command returns you to the main menu of the Exec.
- The **e** 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.

36.4. 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.

36.5. 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 submenu 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.

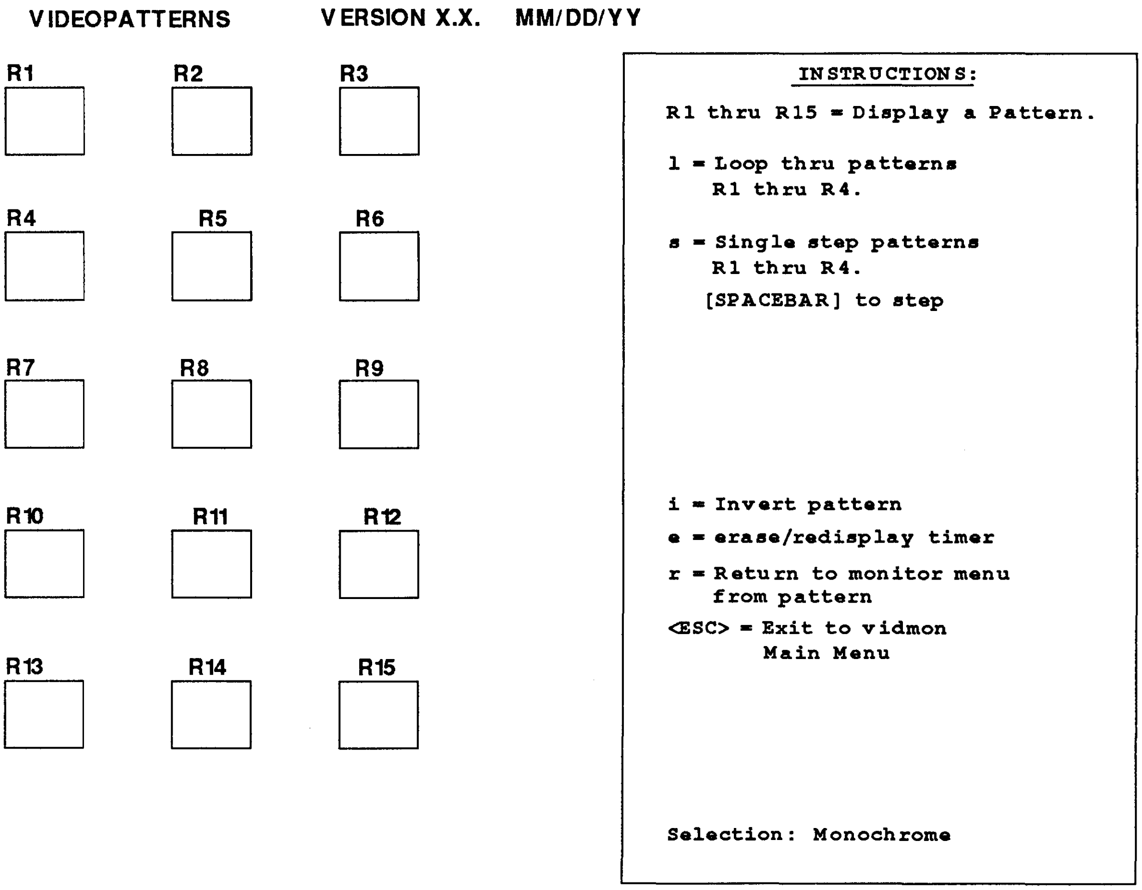
Color

Selecting *Color* from the Main Menu brings up the Color Menu. Use it only if the diagnostic is running on a color monitor.

36.6. 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 36-1 *Monochrome Video Pattern Menu*



```
INSTRUCTIONS:

R1 thru R15 = Display a Pattern.

1 = Loop thru patterns
    R1 thru R4.

s = Single step patterns
    R1 thru R4.
   [SPACEBAR] to step

i = Invert pattern
e = erase/redisplay timer
r = Return to monitor menu
    from pattern
<ESC> = Exit to vidmon
        Main Menu

Selection: Monochrome
```

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

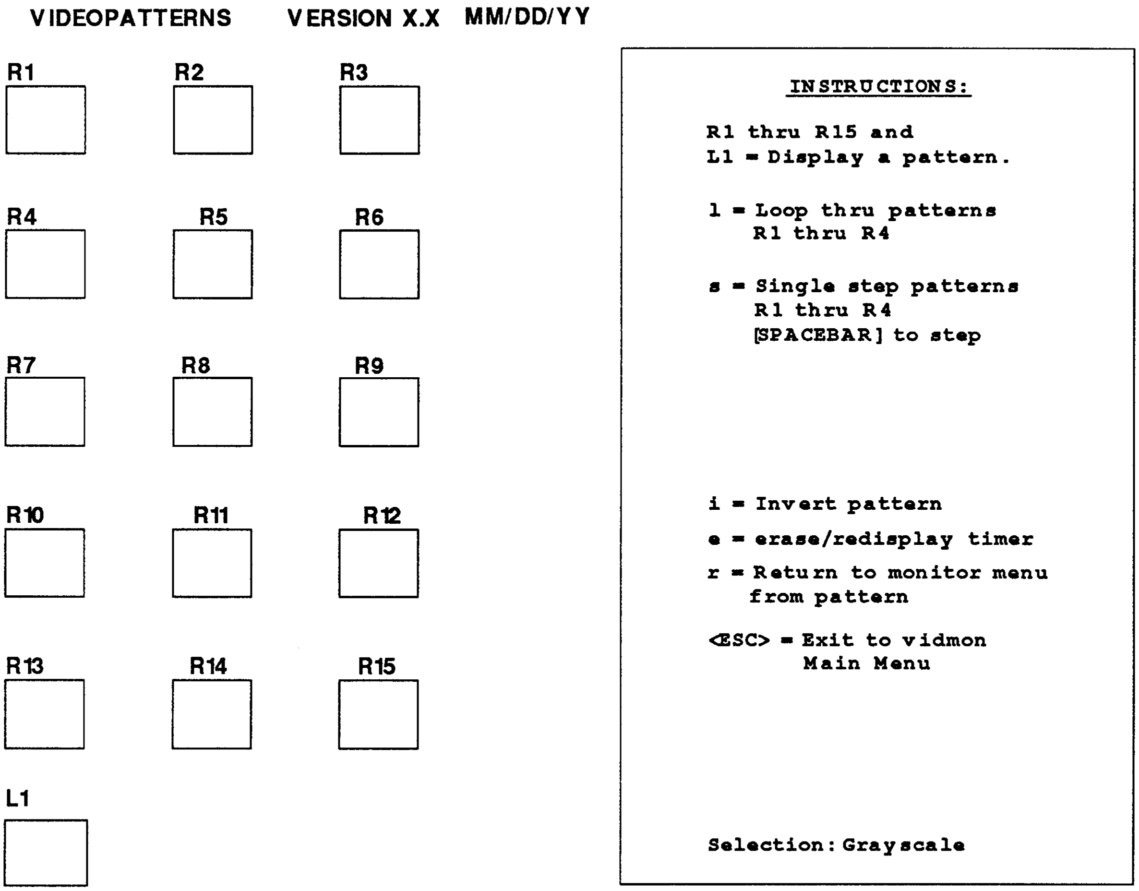
Description:

The patterns R5 thru R15 are useful for checking video shadowing (ringing), high voltage regulation, and screen geometry.

36.7. 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 36-2 *Grayscale Video Pattern Menu*



```
INSTRUCTIONS:

R1 thru R15 and
L1 = Display a pattern.

l = Loop thru patterns
    R1 thru R4

s = Single step patterns
    R1 thru R4
    [SPACEBAR] to step

i = Invert pattern

e = erase/redisplay timer

r = Return to monitor menu
    from pattern

<ESC> = Exit to vidmon
        Main Menu

Selection: Grayscale
```

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 beam converges properly.

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.

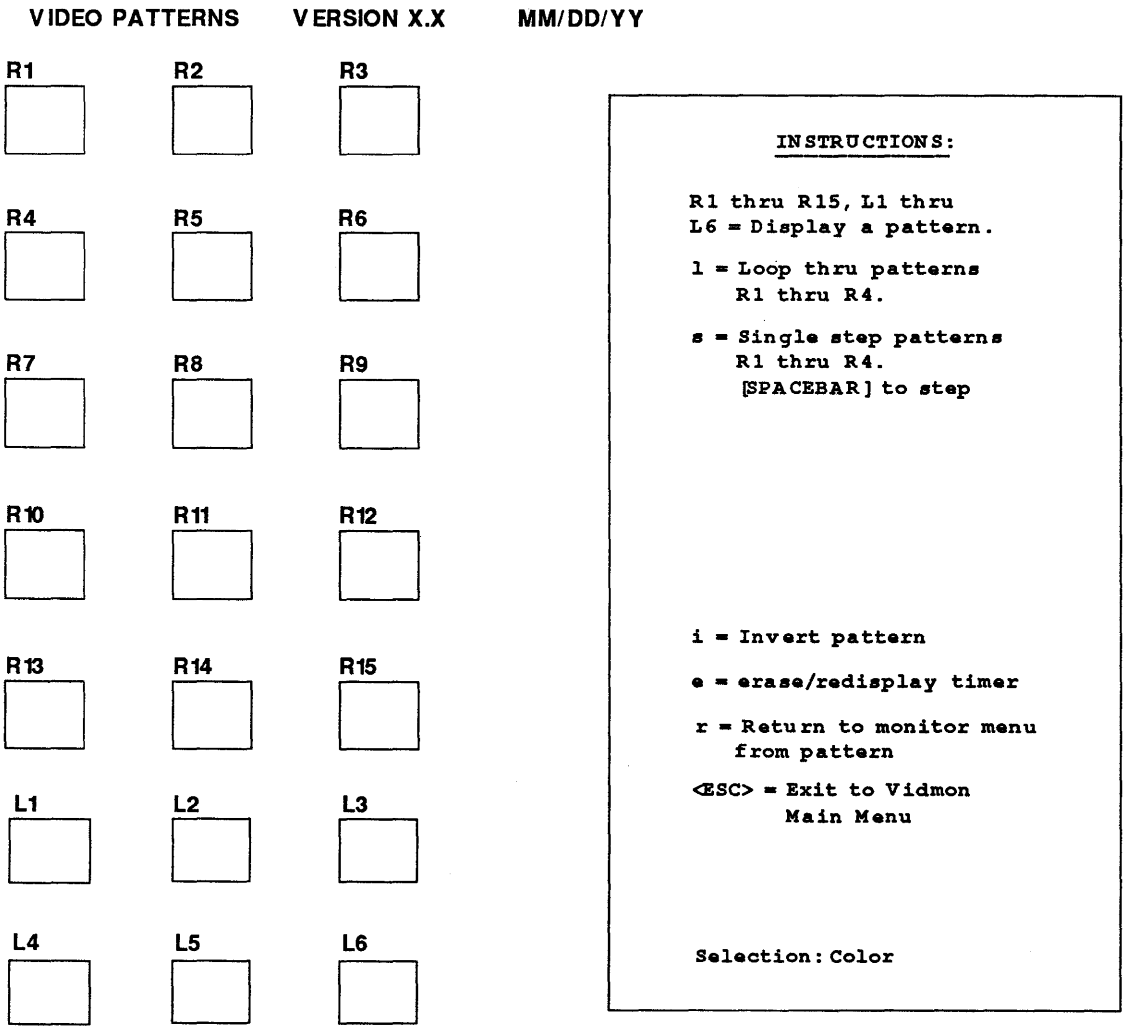
Description:

The pattern is made up of two rows of eight rectangles each on a gray background. The bottom 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 upper row of rectangles starts with a white rectangle. Each succeeding rectangle is more saturated until the last rectangle in the row is black.

36.8. 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 36-3 *Color Video Pattern Menu*



VIDEO PATTERNS VERSION X.X MM/DD/YY

```
INSTRUCTIONS:

R1 thru R15, L1 thru
L6 = Display a pattern.

1 = Loop thru patterns
    R1 thru R4.

s = Single step patterns
    R1 thru R4.
    [SPACEBAR] to step

i = Invert pattern

e = erase/redisplay timer

r = Return to monitor menu
    from pattern

<ESC> = Exit to Vidmon
        Main Menu

Selection: Color
```

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 white, yellow, cyan, green, magenta, red, blue and black. The top row consists of colors that are half intensities of the colors in the bottom row, in the order described above.

36.9. 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

37

Sun VME Interface Diagnostic

37

Sun VME Interface Diagnostic

37.1. 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.

NOTE *The Sun-4/110 CPU board does not have a VME Slave interface; therefore menus for Sun-4/110 VME Interface testing do not refer to Slave tests. Also due to the lack of a VME slave interface on the Sun-4/110 CPU board, it is not possible to use it as a Test Support CPU when testing another Sun-4/110 CPU board.*

37.2. 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 (other than a Sun-4/110).
- Sun Memory board(s) if no memory is on CPU board(s).
- Two Sun consoles (monitor and keyboard).
- A boot device (such as Ethernet).
- 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.
- An InterProcessor Communication Serial Cable (IPC Serial Cable) connecting the UUT to the TSCPU by way of the ''B'' serial ports.

37.3. Hardware Set-Up

The following is an example of a test system configuration that uses the hardware listed previously:

NOTE *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.*

- In a 12-slot pedestal, facing the rear of the system, install the Unit Under Test (UUT) CPU in Slot 1 (far left), and if a memory board is needed, install it in slot 2, 3, 4, 5, or 6. Remember that Sun-3/2xx and Sun-4/2xx memory boards must be terminated. Installation/Configuration procedures for these boards show the location of the terminating resistor.
- The Test Support CPU (TSCPU) should go in Slot 10, with the memory board in Slot 11. The ARBITER and the VME Clock must be disabled on the board that takes the role of TSCPU. (Consult the appropriate CPU board configuration procedure for jumper locations, and refer to the Jumper Placement table below.)
- A Sun console (monitor and keyboard) on the Unit Under Test is required for the user interface. A second console must be connected to the TSCPU for booting the SunDiagnostic Executive and viewing messages that may be displayed while testing is in progress.
- An interprocessor serial communication cable is connected between serial port B on the UUT and serial port B on the TSCPU.
- 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.

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. When the table below specifies that a jumper is IN, the jumper should be present on the board. If the table specifies OUT, the jumper should not be present.

Jumper Description	*UUT*	*TSCPU*
Requester only	OUT	IN
Arbiter/Requester	IN	OUT
Reset slave	OUT	IN
Reset master	IN	OUT
VMEclock enable	IN	OUT

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. Refer to the chapter on the EEPROM editing tool, or to the *PROM User's Manual* for information on how to change EEPROM parameters.

37.4. 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 space bar 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 Main Menu, if desired.

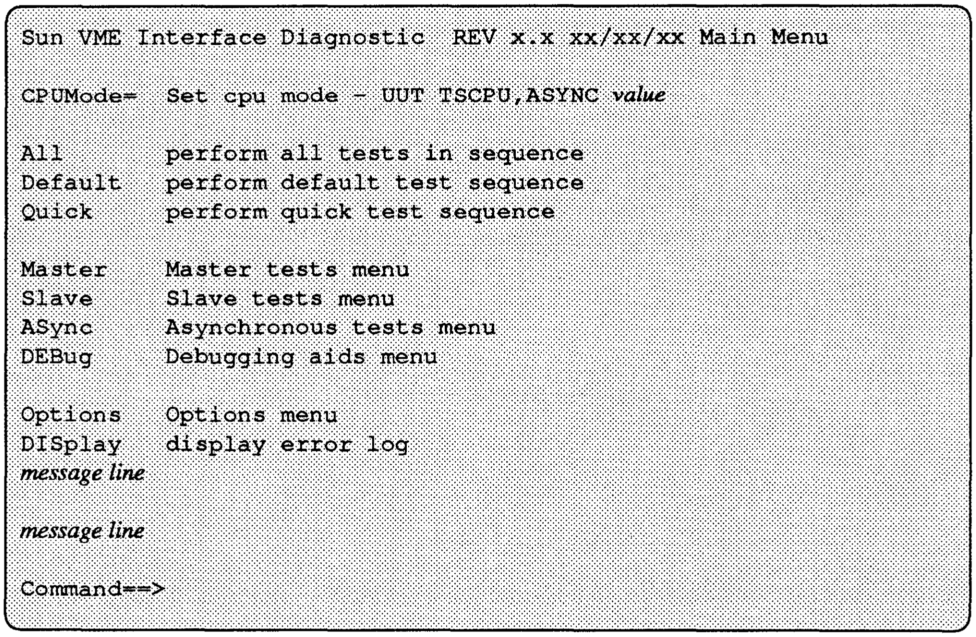
The UUT should display the Main Menu and the TSCPU should occasionally display the message `SLAVE: waiting for command.` ***Don't press any keys on the TSCPU keyboard!*** The TSCPU keyboard is not used unless the CPUMode is set to ASYNC and the Async Tests Menu is selected.

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 Main Menu. Asynchronous mode may be selected from the Main Menu by setting CPUMode=ASYNC.

37.5. 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

CPUMode=  Set cpu mode - UUT TSCPU,ASYNC value

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
DISplay   display error log
message line

message line

Command==>
```

CPUM= *value*

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

`All`

Typing this command runs all three functional units in sequence. 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

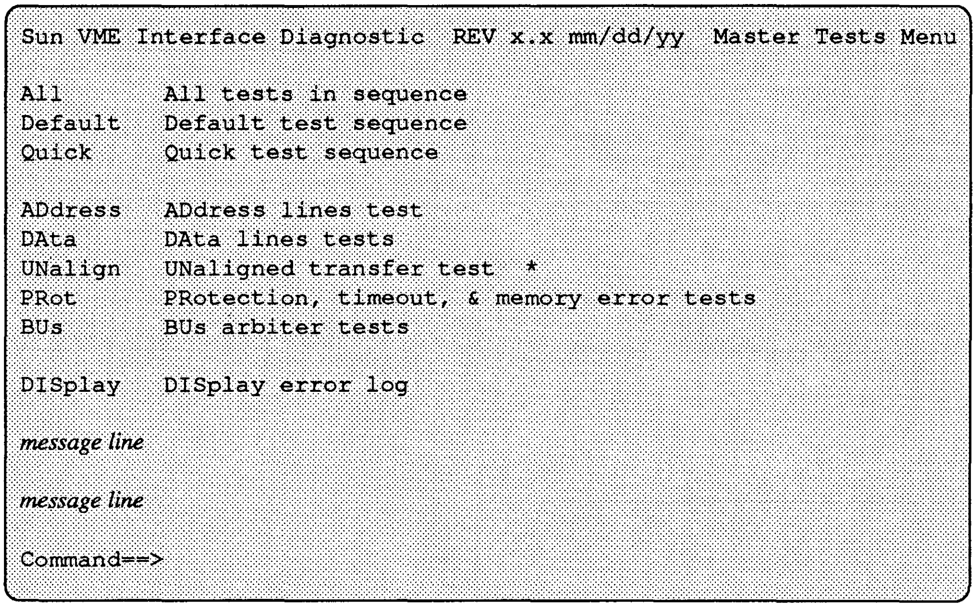
Typing this command brings up a menu of global options.

DIS

Typing this command displays the error log.

37.6. 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 mm/dd/yy  Master Tests Menu

All        All tests in sequence
Default    Default test sequence
Quick      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

DISplay    DISplay error log

message line

message line

Command==>
```

* This test is not available for Sun-4.

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 *DVma= OFFSET= OPSize= OPType= POrt= PType=* **=

Address line tests verify the Master correct manipulation of DTB address lines over specified address range and identify failing address line combinations.

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 *OPSize= OPType= PATtern= POrt= PType= **=*

Data line tests verify Master's correct manipulation of DTB data lines over specified data range. They also identify failing data line combinations. 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 *AIndex= OFFSet= OPType= POrt= SEnse= **=*

Unaligned transfer tests verify the Master's ability to handle non-aligned transfer without error.

NOTE *This test is not available for Sun-4 because SPARC™ architecture does not support unaligned transfers.*

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 *SUbtest=* **=

Protection, timeout and memory error tests verify correct operation of error reporting circuitry specific to the Master's VMEbus interface.

sub-test

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)*
sub-test	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 *ITerations= LEn=* **=

Bus arbiter tests verify correct bus arbiter operation on the Unit Under Test.

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.

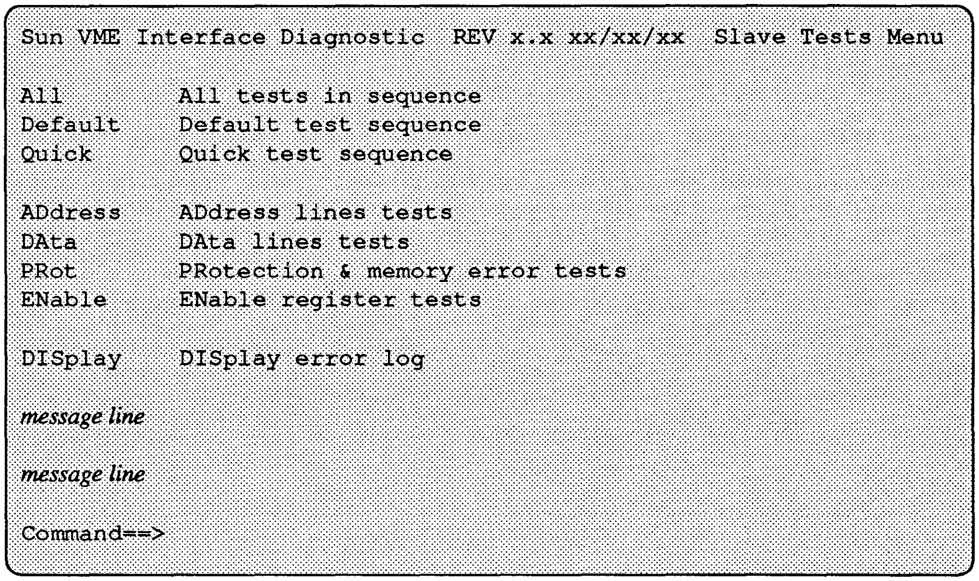
parameter	*default*	*range*	*base (radix)*
iterations	1000	1__f0000	hex
len	100	1__400	hex

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.

37.7. 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. Note that Sun-4/110 menus do not include these options due to the lack of a slave interface on the Sun-4/110 CPU board.



```
Sun VME Interface Diagnostic  REV x.x xx/xx/xx  Slave Tests Menu

All         All tests in sequence
Default     Default test sequence
Quick       Quick test sequence

ADdress     ADdress lines tests
DAta        DAta lines tests
PRot        PRotection & memory error tests
ENable      ENable register tests

DISplay     DISplay error log

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 *DVma= OFFSET= OPSize= OPType= POrt= PType= **=*

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.

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 *OPSize= OPType= PATtern= POrt= PType=* **=

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.

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 *SUbtest=* **=

Protection & memory error tests verify correct operation of Slave VMEbus interface error reporting circuitry.

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 *SUbtest= COntext=* **=

DVMA Enable register tests verify correct operation of the VMEbus DVMA enable circuitry for System and User DVMA.

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.

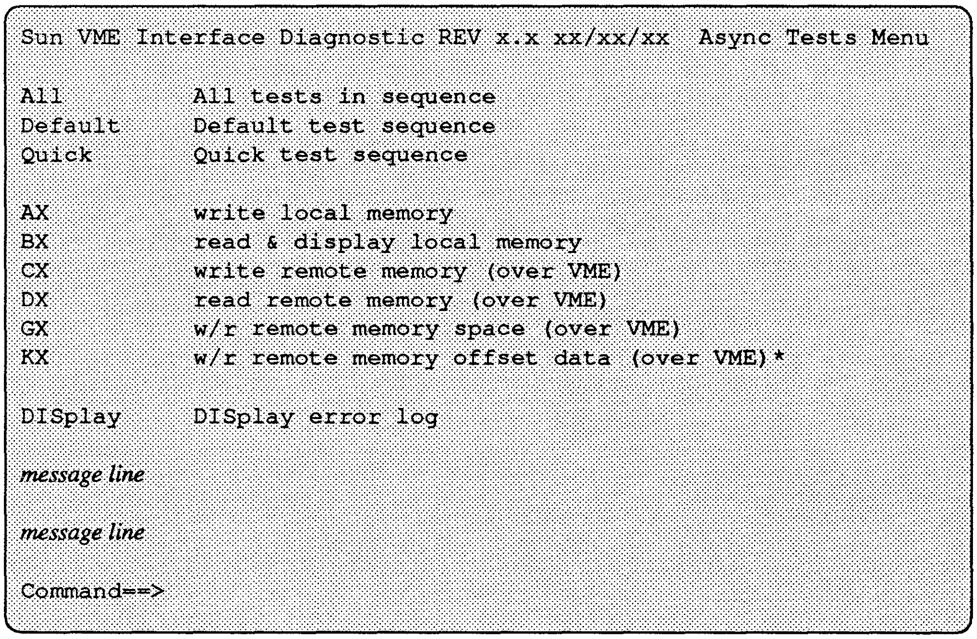
DISplay
This command displays the error log.

37.8. 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* PROM s. 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 looks something like this:



```
Sun VME Interface Diagnostic REV x.x xx/xx/xx  Async Tests Menu

All         All tests in sequence
Default     Default test sequence
Quick       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)*

DISplay     DISplay error log

message line

message line

Command==>
```

* This option is not available on the Sun-4 because SPARC architecture does not support unaligned transfers.

Asynchronous Test menu options are described in the paragraphs that follow.

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 *AIndex= OFFSet= OPSize= PATtern= PType= SEnse= **=*

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.

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 *AIndex= SEnse= OPSize= OFFSet= **=*

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.

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 *AIndex= OFFSet= OPSize= PATtern= POrt= PType= SEnse= **=*
Write Remote Memory (over VMEbus) writes a pattern into a remote memory location over the VMEbus.

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 *AIndex= OFFSet= OPSize= POrt= SEnse= **=*

Read Remote Memory (over VMEbus) reads from a remote memory location over the VMEbus and displays the observed value.

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 *DVma= OFFSet= OPSize= POrt= PType= **=*

Write/Read Remote Memory Space (over VMEbus) verifies correct operation while exercising all address bits.

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 Boot PROM monitor variables.

KX *AIndex= OFFSet= POrt= SEnse= **=*
Write/Read Remote Memory Offset Data (over VMEbus) verifies the

Master's ability to handle non-aligned transfer without error.

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

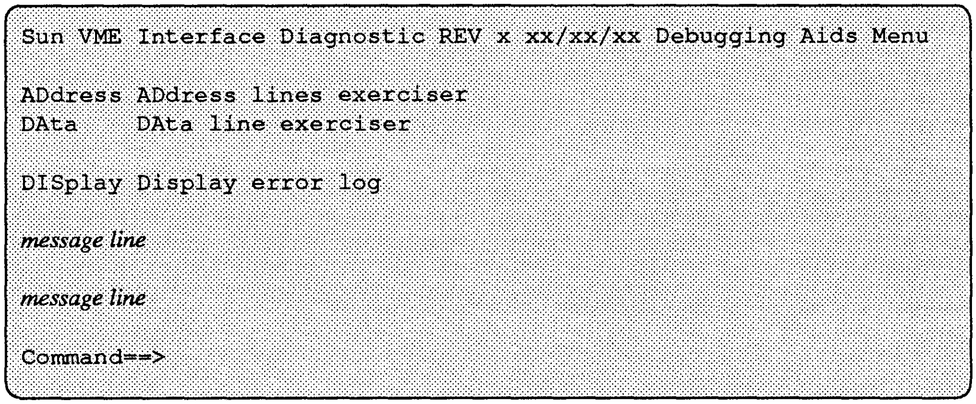
The `Write/Read Remote Memory Offset Data` test executes the test for the number of iterations specified in the `pass` count limit. For each iteration it forces remote writes/reads to/from locations not modulo 4 aligned. It compares written values with values actually read back to verify correct operation. 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.

DISplay
This option displays the error log.

37.9. 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 three visible options.



```
Sun VME Interface Diagnostic REV x xx/xx/xx Debugging Aids Menu

ADdress ADdress lines exerciser
DAta    DAta line exerciser

DISplay Display error log

message line

message line

Command==>
```

ADdress *AIndex= OFFSet= OPSize= OPType= POrt= **=*

The *Address line exerciser* exercises address lines for debugging with an oscilloscope or logic analyzer.

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` *AIndex= OFFSet= OPSize= OPType= PATtern= POrt= PType= SEnse=* **= The *Data line exerciser* exercises data lines for debugging with an oscilloscope or logic analyzer.

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.

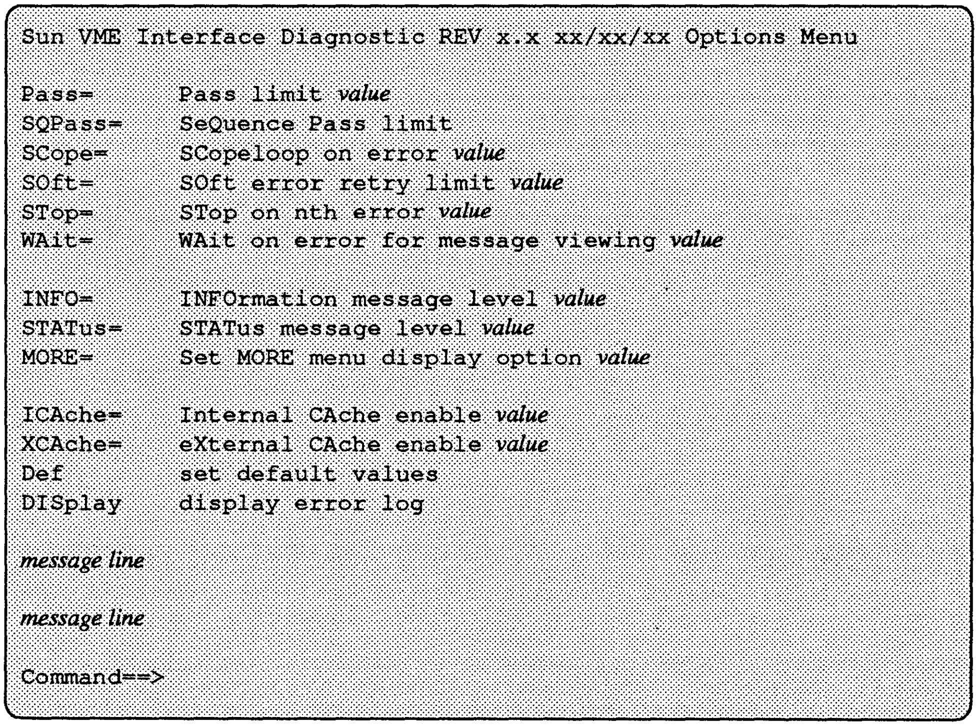
`DISplay`

This option displays the error log.

37.10. 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

Pass=      Pass limit value
SQPass=    SeQuence Pass limit
SCope=     SCopeloop on error value
SOft=      SOft error retry limit value
STop=      STop on nth error value
WAit=      WAit on error for message viewing value

INFO=      INFOrmation message level value
STATus=    STATus message level value
MORE=      Set MORE menu display option value

ICAche=    Internal CAche enable value
XCAche=    eXternal CAche enable value
Def        set default values
DISplay    display error log

message line

message line

Command==>
```

P=*value*

The value you enter after **P=** sets the number of times a test is executed.

To set this mode, enter

Pass=

followed by a hexadecimal value within the range shown below:

variable	*default*	*range*	*base (radix)*
pass	1	1:MAXINT	dec

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	dec

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:MAXINT	dec

`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:MAXINT	dec

`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=*value*

followed by **0** to turn the flag OFF, or **1** to turn the flag ON.

variable	*default*	*range*	*base (radix)*
wait	0	0:1	dec

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.

INFO=*value*

This parameter sets the amount of information displayed.

0 = off
1 = terse
2 = verbose
3 = most verbose

Default is `info=1`.

STATUS=*value*

This command sets the status message level.

0 = off
1 = terse
2 = verbose
3 = most verbose

Default is `status=0`.

MORE=

This option controls the MORE menu display option. Entries are as follows:

more=dis	*disables the MORE command*
more=ena	*enables the MORE command*

ICAche=*value*

This command enables internal cache(s).

0 = cache off
1 = enable instruction cache
2 = enable data cache
3 = enable both

Default is `ica=1`.

XCAche=*value*

This command enables external cache(s).

0 = cache off
1 = enable central cache
2 = enable I/O cache
3 = enable both

Default is `xca=3`.

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. They 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 you to continually go to The Options Menu.

The command syntax is:

TESTCMD TESTPARAMETER= any parameters listed below

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.

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.

INFO=
sets info message level. 0 = off, 1 = terse, 2 = verbose, 3 = more verbose.

STATUS=
sets status message level. 0 = off, 1 = terse, 2 = verbose, 3 = more verbose.

ICAche
enables internal cache(s) 0 = off, 1 = instruction, 2 = data, 3 = both.

XCAche
enables external cache(s) 0 = off, 1 = central, 2 = io, 3 = both.

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)*
Pass	1	1__MAXINT	dec
SCope	0	0__1	dec
SOft	0	0__MAXINT	dec
STop	0	0__MAXINT	dec
WAit	0	0__1	dec
INFO	1	0__3	dec
STATUS	0	0__3	dec
ICAche	1	0__3	dec
XCAche	3	0__3	dec

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.

37.11. 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-4 CPU board, containing the SF9010IU 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-4 CPUboard installed in the VMEbus cardcage and used to support testing of the UUT.

UUT
Unit Under Test is the Sun-4 CPU board whose VMEbus interfaces are being tested.

VMEbus
An interfacing system used to interconnect data processing, data storage, and peripheral control devices.

37.12. VME Map Table

Table 37-1 *VME Map Table*

	bit			*port/*	*VME bus*	*virtual*	*physical*
(i,p,s)	*no.*	*value*	*sense*	*context*	*address*	*address*	*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 37-1 *VME Map Table— Continued*

	bit			*port/*	*VME bus*	*virtual*	*physical*
(i,p,s)	*no.*	*value*	*sense*	*context*	*address*	*address*	*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 37-1 *VME Map Table— Continued*

	bit			*port/*	*VME bus*	*virtual*	*physical*
(i,p,s)	*no.*	*value*	*sense*	*context*	*address*	*address*	*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 37-1 *VME Map Table— Continued*

	bit			*port/*	*VME bus*	*virtual*	*physical*
(i,p,s)	*no.*	*value*	*sense*	*context*	*address*	*address*	*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 37-1 *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

Standalone Cache and ECC Tests

A

Standalone Cache and ECC Tests

A.1. Introduction

This appendix discusses the standalone Cache and ECC Memory Tests. The tests do not run under the SunDiagnostic Executive but are included on the tape. The Sun-3 and Sun-4 Cache and ECC Memory tests support the Sun-3/2xx and Sun-4/2xx workstations only.

A.2. Standalone Cache Test

The standalone Sun-x Cache Test is a building block test that assumes the minimal logic required to load and execute the test from the various Sun-x I/O diagnostic media.

The diagnostic isolates to the failing cache function and displays relevant good and bad status of the cache/memory system sufficient for Sun engineering, field, and manufacturing test personnel to troubleshoot and isolate to the failing chip or to the failing board. The program provides test selection thru the standard standalone user interface as well as a default, autosequence means of executing the tests.

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 errors messages.

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 associative 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.

A.3. Hardware Requirements

The Sun CPU board is assumed functional (MMU, Cache, video RAM, data path to memory) as checked out by the boot PROM.

Memory must have minimal functionality sufficient to permit loading and executing the Cache Diagnostic program from it.

Also, a boot path (Ethernet, disk, or tape) must be functional in order to load the standalone program into Sun main memory.

The CPU board should contain a PROM of revision level 1.6 or later.

A.4. Limitations

The program uses the current, Sun- standalone token eat/parse user command interface. No error logging is performed; errors are reported as they occur.

The program tests all cache functions but without concurrent DVMA or Ethernet activity with which the cache flush operation arbitrates on a cycle by cycle basis. Therefore, this test does not provide a 100 percent coverage of the cache operation but certainly provides better than 95 percent coverage of the cache.

The use of the diagnostic by other than by a skilled technician or engineer should be limited to running the short, default, or long test sequences that are selectable from the main menu.

A.5. 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 CPU 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 `cachex.diag` file number. The "x" will be either a 3 or a 4, depending on which tape you are booting. Boot the diagnostic directly off the tape, as shown in Chapter 2.

For example, if cache4.diag is already loaded onto disk, you may boot it with a command such as:

```
b stand/cache4.diag
```

User Command Interface

One main menu lists the tests described in this document. You enter the appropriate command to execute the test after specifying the test control options. To view the menu of tests, enter Return after the prompt:

```
Memory available to test = 0x40000 to 0x nnnnnn
Sun-xxx Cache Diagnostic Rev X.X mm/dd/yy

command:

Return
```

The menu that comes up looks something like this:

```
    d - cache data tests menu
    t - cache tag tests menu
    r - cache read hit menu
    w - cache write hit menu
    m - cache mmu tests 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
    E - exerciser menu
    X - exerciser tests
    q - quick tests
    D - default tests
    P - default single pass
    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 miss 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 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 that have occurred since starting the test sequence will be displayed.

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 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, discussed next.

The ^ command restarts the diagnostic.

Option Menu

The Option Menu provides eleven choices and access to other level menus. When the option is set it applies to all tests that execute after that. Enter **o** from the main menu and then Return to view the option menu, which looks something like this:

```
Sun xxx Cache Diagnostic Rev x.x MM/DD/YY

Selections:

    c - continue on error (default)
    h - halt on error
    d - clr halt on error
    l - loop on error
    k - clr loop on error
    n - inhibit error messages
    e - enable error messages (default)
    a - print run acknowledge
    r - inhibit print run acknowledge (default)
    ^ - pop up a menu level
    ? - help

  Command :
```

Option `c` causes the test to continue after an error.

Option `h` causes the test to halt upon error and return to the menu command input.

Option `d` clears the halt on error option.

Option `l` causes a loop on the failing test.

Option `k` clears the loop on error option.

Option `n` causes *only* fatal error messages to be printed.

Entering `e` enables printout of error messages.

Option `a` prints asterisks periodically to indicate that the program is still running.

Option `r` turns option `a` off.

The Cache Data Tests Menu

Enter **d** from the main menu, and then Return to view the Cache Data Tests Menu. The Cache Data Tests test the cache-control-space-accessed data RAM. The Cache Data Tests Menu looks something like this:

```
Sun xxx Cache Diagnostic Rev X.X MM/DD/YY

  Selections:

    d - cache data write/read test
    P - cache data pattern write/read test
    a - cache data address test
    A - cache inverse data address test
    o - cache data walking 1s data test
    z - cache data walking 0s data test
    p - cache data 3-pattern test
    m - cache data march test
    ^ - pop up a menu level
    ? - help
```

The Cache Data Tests Menu selections are described below. The tests themselves are described in detail later.

d *[addr] [size] [npass]*

Entering the `d` executes the *cache data write/read test*, starting from the address *addr* for the *size* in bytes, and for the number of test passes *npass*, where *addr* must be >= 0 and < 0x1fffc and *addr* + *size* must be less than 0x1fffc.

P *[addr] [size] [pattern] [npass]*

Entering the `P` command executes the *cache data write/read test*, starting from the address *addr* for the *size* in bytes, using the specified 32-bit write data pattern and for the *npass* number of test passes, where *addr* must be >= 0 and < 0x1fffc and *addr* + *size* must be less than 0x1fffc.

a *[addr] [size] [npass]*

Entering the `a` command executes the *cache data address test*, from the address entered for *addr* for the *size* in bytes, and for the number of test passes set by *npass*, where be*addr*must *addr* + *size* must be less than 0x1fffc.

A *[addr] [size] [npass]*

Entering the `A` command executes the *cache inverse data address test* from the address entered for *addr*, for the size in bytes, entered for *size*, and for the number of test passes entered for *npass*, where *addr* must be >= 0 and < 0x1fffc and size*addr*+ must be less than 0x1fffc.

o *[addr] [size] [pattern] [npass]*

Entering the `o` command executes the *cache data write/read test*, starting from the address given for *addr*, for the size in bytes entered for *size*, using a bit rotated data pattern of 0x11111111, and for the number of test passes entered for *npass*. *addr* must be >= 0 and < 0x1fffc and size*addr*+ must be less than 0x1fffc.

z *[addr] [size] [pattern] [npass]*

Entering the z command executes the *cache data write/read test*, starting from the address *addr* for the size in bytes entered for *size*, using a bit rotated 32-bit pattern of 0xEEEEEEEE write data pattern, and for the number of test passes entered for *npass*. *addr* must be >= 0 and < 0x1fffc and *addr* + *size* must be less than 0x1fffc.

p *[npass]*

Entering the p command executes the *cache data 3-pattern test* throughout the cache data RAM for *npass* test passes.

m *[npass]"*

The m command executes the *cache data march test* throughout the cache data RAM for *npass* test passes.

? Entering ? displays the Data Tests command syntax:

```
  Sun xxx Cache Diagnostic Rev X.X MM/DD/YY

    Selections:

      d [addr >=0 <=0x1fffc] [size <= 0x1fffe] [npass]
      P [addr >=0 <=0x1fffc] [size <= 0x1fffe] [npass]
      a [addr >=0 <=0x1fffc] [size <= 0x2000] [npass]
      A [addr >=0 <=0x1fffc] [size <= 0x2000] [npass]
      o [addr >=0 <=0x1fffc] [size <= 0x2000] [npass]
      z [addr >=0 <=0x1fffc] [size <= 0x2000] [npass]
      p [npass]
      m [npass]
```

Cache Tags Tests Menu

Enter **t** from the main menu, then Return to view the Cache Tags test menu, which looks something like this:

```
Sun xxx Cache Diagnostic Rev X.X MM/DD/YY

  Selections:

    d - cache tags write/read test
    P - cache tag pattern write/read test
    a - cache tag address test
    A - cache inverse tag address test
    o - cache data walking 1s tags test
    z - cache data walking 0s tags test
    p - cache tags 3-pattern test
    m - cache tags march test
    ^ - pop up a menu level
    ? - help
```

The Cache Tags Tests Menu selections are described below. The tests are described later.

d *[addr] [size] [npass]*

Entering the d command executes the *cache tags write/read test*, starting from the address given for *addr* for the size in bytes given for *size*, and for the number of test passes set by *npass*, where *addr* must be >= 0 and < 0x1fffc and *addr* + *size* must be less than 0x1fffc.

P *[addr] [size] [pattern] [npass]*

Entering the P command executes the *cache tag write/read test*, starting from the address given for *addr*, for the size in bytes given for *size*, using the specified 32-bit write data pattern, and for the number of test passes set by *npass*. *addr* must be >= 0 and < 0x1fffc and *addr* + *size* must be less than 0x1fffc.

a *[addr] [size] [npass]*

Entering the a command executes the *cache tags address test*, from the address entered for *addr*, for the size in bytes set by *size*, and for the number of test passes set by *npass*. *addr* must be >= 0 and < 0x1fffc and *addr* + *size* must be less than 0x1fffc.

A *[addr] [size] [npass]*

Entering the A command executes the *cache inverse tag address test*, from the address given for *addr*, for the size in bytes set by *size*, and for the number of test passes set by *npass*. *addr* must be >= 0 and < 0x1fffc and *addr* + *size* be less than 0x1fffc.

o [addr] [size] [pattern] [npass]

Entering the o command executes the *cache tag write/read test*, starting from the address given for *addr*, for the size in bytes specified by *size*, using a bit rotated tag pattern of 0x11111111, and for the number of test passes set by *npass*, where *addr* must be >= 0 and < 0x1fffc and size*addr*+ must be less than 0x1fffc.

z *[addr] [size] [pattern] [npass]*

The z command executes the *cache tag write/read test*, starting from the value entered for *addr*, for the size in bytes specified by *size*, using a bit rotated 32-bit pattern of 0xEEEEEEEE write tag pattern, and for the number of test passes set by *npass*. *addr* must be >= 0 and < 0x1fffc and *addr* + *size* must be less than 0x1fffc.

p *[npass]*

Entering the p command executes the *cache tags 3-pattern test*, throughout the cache tag RAM for *npass* number of test passes.

m *[npass]*

The m command executes the *cache tag march test* throughout the cache tag RAM for *npass* number of test passes.

Entering **?** displays the Cache Tags Test command syntax:

```
  Sun xxx Cache Diagnostic Rev X.X MM/DD/YY

    Selections:

      d [addr >=0 <=0x1fffc] [size <= 0x1fffe] [npass]
      P [addr >=0 <=0x1fffc] [size <= 0x1fffe] [npass]
      a [addr >=0 <=0x1fffc] [size <= 0x2000] [npass]
      A [addr >=0 <=0x1fffc] [size <= 0x2000] [npass]
      o [addr >=0 <=0x1fffc] [size <= 0x2000] [npass]
      z [addr >=0 <=0x1fffc] [size <= 0x2000] [npass]
      p [npass]
      m [npass]
```

The Cache Read Hit Tests Menu

Entering the **r** command from the main menu, followed by (Return), yields a menu that looks something like this:

```
Sun 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
    ^ - pop up a menu level
    ? - help

  Command :
```

The Cache Read Hit Tests Menu selections are described below.

r *[addr > 0x40000] [size] [npass]*

Entering the `r` command executes the *cache read hit test* to be performed from base address *addr* which must be greater than 0x40000 for memory *size* for a number of passes entered in place of *npass.*

c *[addr > 0x40000] [size] [npass]*

Entering the **c** command executes the *cache read hit (cx different) test* to be performed from base address *addr*, which must be greater than 0x40000 for memory size specified by *size* for a number of passes specified by *npass.*

b *[npass]*

Entering the `b` command executes the *cache read hit byte alignment test*, to be performed for a number of passes entered in place of *npass.*

? Entering ? displays the Cache Read Hit Test command syntax:

```
Sun XXX  Cache Diagnostic Rev X.X. MM/DD/YY

Selections:

  r [addr >= 0x40000] [size <= 0x20000] [npass]
  c [addr >= 0x40000] [size <= 0x20000] [npass]
  b [npass]
  l [npass]

 Command :
```

The Cache Write Hit Tests Menu

```
Sun 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 :
```

w *[addr > 0x40000] [size] [npass]*

Entering the w command executes the *cache write hit test*, performed from base address *addr* which must be greater than 0x40000 for memory size specified by *size*. The number of passes is specified by *npass*.

m *[addr > 0x40000] [size] [npass]*

Entering the m command executes the *cache modify write hit test* to be performed from base address *addr* which must be greater than 0x40000 for memory size specified by *size*, for a number of passes specified by *npass*.

b *[npass]*

Entering the b command executes the *cache write hit byte alignment test* to be performed for a given number of passes.

? Entering ? displays the Cache Write Hit command syntax:

```
    Sun Sirius Cache Diagnostic Rev X.X MM/DD/YY

    Selections:

        w [addr > 0x40000] [size] [npass]
        m [addr > 0x40000] [size] [npass]
        b [npass]

      Command :
```

Cache MMU Protection Menu

Entering **m** from the main menu, followed Return, brings up the Cache MMU Protection Menu, which looks something like this:

```
w - cache write hit protect violation test
^ - pop up a menu level
? - help
```

If you enter a question mark, the command syntax is shown:

```
w [addr >= 0x40000] [don't care] [npass]
```

w *[addr] . [npass]*

Entering w from this menu executes the *cache write hit protect violation test*, o be performed on the *addr* specified for a number of passes set by *[npass]*.

The Cache Read Miss Tests Menu

Entering **R** from the main menu, followed by Return, brings up a menu something like this:

```
Sun-XXX Cache Diagnostic Rev X.X MM/DD/YY

    n - cache read miss/no writeback (not dirty) test
    d - cache read miss/writeback (valid & dirty) test

  Command :
```

The Cache Read Miss Tests Menu selections are described below.

n *[addr > 0x40000] [size] [npass]*

Entering the **n** command executes the *cache read miss/no writeback (not dirty) test* to be performed from base address *addr* which must be greater than 0x40000 for memory size specified by *size* for a number of passes specified in place of *npass*.

d *[addr > 0x40000] [size] [npass]*

The d command executes the *cache read miss/writeback (valid & dirty) test* to be performed from base address *addr*, which must be greater than 0x40000 for memory size specified by *size* for a number of passes *npass*.

? If you enter a question mark, the Cache Read Miss command syntax is displayed:

```
SunXXX Cache Diagnostic Rev X.X MM/DD/YY

  n [addr > x040000 ] [size] [npass]
  i [addr > x040000 ] [size] [npass]
  d [addr > x040000 ] [size] [npass]

 Command :
```

The Cache Write Miss Tests Menu

Entering **W** from the main menu, followed with a (Return), brings up a menu that looks something like this:

```
Sun xxx Cache Diagnostic Rev X.X MM/DD/YY

  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.

n *[addr > 0x40000] [size] [npass]*

Entering the n command executes the *cache write miss/no writeback (not dirty) test*, to be performed from base address *addr* which must be greater than 0x40000 for memory size specified by *size* for a number of passes specified by *npass*.

d *[addr > 0x40000] [size] [npass]*

Entering the d command executes the *cache write miss/writeback (valid & dirty) test*, to be performed from base address *addr* which must be greater than 0x40000 for memory size specified by *size* for a number of passes specified by *npass*.

? Entering a question mark displays the Cache Write Miss command syntax:

```
Sun xxx Cache Diagnostic Rev X.X MM/DD/YY

  n [addr >= 0x40000] [size] [npass]
  d [addr >= 0x40000] [size] [npass]

 Command :
```

The Cache Writeback Error Tests Menu

Entering **e** from the main menu, followed with a [Return], brings up a menu that looks something like this:

```
Sun XXXX Cache Diagnostic Rev X.X MM/DD/YY

w - cache write miss/writeback timeout error test
r - cache read miss/writeback timeout error test
W - cache write miss/writeback translation error test
R - cache read miss/writeback translation error test
^ - pop up a menu level
? - help
Command :
```

w *[npass]*

The w command executes a write miss to an existing memory address that is not cached. This action should cause an attempt to writeback a block to a non-existent memory address. *npass* specifies the number of passes the test will execute.

r *[npass]*

Entering r causes a read miss to an existing memory address that is not cached. This action should cause an attempt to writeback a block to a non-existent memory address. *npass* specifies the number of passes the test will execute.

W *[npass]*

Entering the W command executes the *cache write miss/writeback translation error test* to be performed for the given number of passes. The test causes a write miss to an existing memory address that is not cached. This action should cause an attempt to writeback a block to an invalid memory address.

R *[npass]*

Entering the r command causes a read miss to an existing memory address that is not cached. This action should cause an attempt to writeback a block to an invalid memory address. *npass* specifies the number of test passes.

The Cache Flush Tests Menu

Entering **f** from the main menu, followed with Return, brings up a menu that looks something like this:

```
Sunxxx Cache Diagnostic Rev X.X MM/DD/YY

  Selections:

    c - context flush test
    p - page flush test
    s - segment flush test
    ^ - pop up a menu level
    ? - help

  Command :
```

The following paragraphs describe the Cache Flush Tests Menu selections.

c *[addr >=0x40000] [size <= 0x10000] [npass]*
Entering the **c** command executes the *context flush test*, to be performed from the base address *addr >=0x40000* for memory size *size <= 0x10000* for a number of passes specified by *npass*.

p *[page base addr] [size <= 0x2000] [npass]*
Entering the `p` command executes the *Page Flush test*, to be performed from the base address *page_base_addr* for memory size *size <= 0x2000* for a number of passes specified by *npass*.

s *[segment base addr] [size >= 0x20000] [npass]*
Entering the `s` command executes the *Segment Flush Test* to be performed from the base address *segment_base_address* for memory size *size >=0x20000* for a number of passes specified by *npass*.

If you enter **?**, the Flush Test command syntax is displayed:

```
Sun-xxx Cache Diagnostic Rev X.X MM/DD/YY

  Selections:

    c  [addr >=0x40000] [size <= 020000] [npass]
    p  [page_base_addr] [size <= 020000] [npass]
    s  [segment_base_addr] [size <= 020000] [npass]

  Command :
```

The Cache Physical Address Compare Tests Menu

Entering the **p** command from the main menu, followed by Return, brings up a menu that looks something like this:

```
Sun-xxx Cache Diagnostic Rev X.X MM/DD/YY

  Selections:

    r - cache read (not dirty)/physical address compare test
    R - cache read (dirty)/physical address compare test
    u - cache read (uncached)/physical address compare test
    ^ - pop up a menu level
    ? - help

  Command :
```

r *[address_is_mult_of_0x20000] [size >= 0x20000] [npass]*
Entering the `r` command executes the *Cache Read (not dirty)/Physical Address Compare test,* to be performed from the base address *address_is_mult_of_0x20000* for memory size *size >= 0x20000* for a number of passes specified by *npass.*

w *[address_is_mult_of_0x20000] [size >= 0x20000] [npass]*
Entering the `w` command executes the *Cache Read (dirty)/Physical Address Compare* test, to be performed from the base address *address_is_mult_of_0x20000* for memory size *size >= 0x20000* for a number of passes specified by *npass.*

u *[address_is_mult_of_0x20000] [size >= 0x20000] [npass]*
Entering the `u` command executes the *Cache Read (uncached)/Physical Address Compare* test, to be performed from the base address *address_is_mult_of_0x20000* for memory size *size >= 0x20000* for a number of passes specified by *npass.*

The Exerciser Tests Menu

Entering **E** from the main menu and then (Return) brings up a menu of exercise tests that looks something like this:

```
Sun xxx Cache Diagnostic Rev X.X MM/DD/YY

  Selections:

    w - cached execution/memory write/read test
    f - cached execution/memory write/flush/read test
    n - cached fetch nop test
    ^ - pop up a menu level
    ? - help

  Command :
```

The following paragraphs describe the Exerciser Tests Menu selections.

w *[addr] [size <= memory_available] [npass]*
Entering the `w` command executes the *Cached Execution Memory Write/Read test*, to be performed from the base address for the memory size entered in place of *size <= memory_available* for a number of passes specified by *npass.*

f *[addr] [size <= memory_available] [npass]*
Entering the `f` command executes the *Cached Memory Write/Flush/Read test*, to be performed from the base address for the memory size specified in place of *size <= memory_available* for a number of passes specified by *npass.*

n *[addr] [size <= 0x20000] [npass]*
Entering the `n` command executes the *Cache Fetch NOP Test* from the base address for the memory size and number of passes specified.

? Entering `?` displays the Exerciser Test command syntax:

```
Sun xxx Cache Diagnostic Rev X.X MM/DD/YY

  Selections:

    w  [addr >= 0x40000] [size <= mem available] [npass]
    f  [addr >= 0x40000] [size <= mem available] [npass]
    n  [addr >= 0x40000] [size <= 0x20000] [npass]
    ^ - pop up a menu level
    ? - help

  Command :
```

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 that have occurred since starting the default test sequence will be displayed.

A.6. 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
- □ Exerciser tests

Cache Data 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 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 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 0x00000040 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 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 Data 3-Pattern Test

This test performs three passes of pattern writes and reads of the entire cache data RAM address space. The patterns used for the three passes are 0xA5972C5A, 0x5AA5972C, and 0x2C5AA597 for the first pass; 0x5AA5972C, 0x2C5AA597, and 0x972C5AA5 for the second pass; and 0x2C5AA597, 0xA5972C5A, and 0x5AA5972C for the last pass.

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. It will then enter a scopeloop in which the failing write pattern is written to the failing address followed by a write to the next address of the inverse pattern, and finally, a read of the failing address.

```
addr 0x00000000 exp 0x5A972C5A, obs 0x5B972C5A
```

Cache Data March Test

The Cache Data March Test writes all zeros in cache data control space from the lowest to the highest address, then reads zeros and writes ones from the highest to the lowest address, and finally, reads ones and writes zeros from the lowest to the highest address. It is a test of the SRAM data integrity.

Cache Tag Tests

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 0x00000000 exp 0x5A972C5A, obs 0x5B972C5A
```

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 Tag Address Test

For all cache tag addresses, this test writes the address as data at each cache tag control space address. It then reads back all addresses and verifies that each cache tag address contains its address as data. This is a test of addressing uniqueness of the cache tag 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 tag space.

Error Description

Typical error messages are:

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

Cache Inverse Tag Address Test

For all cache tag addresses, this test writes the inverse of the address as data at each cache tag control space address. 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 Walking Ones Test

This test writes and reads each address of the cache tags in device control space for the operator specified address range with a bit shifted 1 in a 32-bit field of all zeros pattern, after "anding" the pattern with the 32-bit tag bit mask to eliminate nonexisting tag bits in the 32-bit word. This test writes the address with the operator specified pattern then inverts the pattern and writes the next address, then reads back the original address and compares it with the noninverted pattern. Upon error, it loops on failing cache tags long word address.

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 Tags 3-Pattern Test

Three passes of write/read tests are performed on the entire cache tag RAM address 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 0x0C5A1A00, 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 Tags March Test

The Cache Tag March Test writes all zeros in the cache tag control space from the lowest to the highest address then reads zeros and writes ones from the highest to the lowest address, and finally reads ones and writes zeros from the lowest to the highest address. It is a test of the SRAM data integrity.

Cache Read Hit Tests

Cache Read Hit Test

The Cache Read Hit Test verifies that doing a supervisor data operand read from system memory with a valid memory block address in the cache will cause a read from the cache and not from system memory. The following sequence is executed in testing cache hits over the virtual address range specified by the user.

(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 in the specified address range:

(a) clear the cache tag/data entries for the address,

(b) store the address in the cache tag entry,

(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 inverse of the virtual address, and, therefore, is not read from system memory.

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 Byte Alignment Test

The Cache Read Hit Byte Alignment Test verifies that byte alignment is correct for all byte move, read instructions that are read hits. It performs the operation with the cache enabled, then with the cache disabled for the same read data and compares the results.

Error Description

Typical error messages during the test are:

```
Read hit byte alignment error during a byte read

addr 0x40000 exp 0x000000FF obs 0x0000FF00
```

Cache Write Hit tests

Cache First Write Hit Test

The Cache First Write Hit Test verifies that doing a supervisor data operand write to cache with a valid memory block address in the cache will cause the cache data for that cache address to be updated and the modified bit to be set.

The sequence of steps performed within the test is as follows:

(1) Clear all cache tag field addresses in cache tag control space.

(2) Clear all cache data control space addresses.

(3) For the address range specified by the menu command:

(a) store the address in the cache tag entry and set the (valid,modified) bits to (1,0),

(b) store the longword address inverted as data in cache data control space,

(c) write the virtual address with the address,

(4) verify that the address in the cache is updated with data equal the noninverted address, and, that, (valid,modified) = (1,1) and that (wrt enable,supvsr only) = (1,0).

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 0x40000, exp 0x00040000, obs 0xFFFDFFFF
```

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 0x0000010, exp 0xc0000000, obs 0x80000000
```

Cache Modify Write Hit Test

The Cache Modify Write Hit Test verifies that doing a supervisor data operand write to cache with the memory block address in the cache valid will cause the cache data for that cache address to be updated and the modified bit to be set.

```
(1) Clear all cache tag field addresses in cache tag control space.

(2) Clear all cache data control space addresses.

(3) For the virtual address range specified by the usr menu command:

    (a) store the address in the cache tag entry and set
        the (valid,modified) bits to (1,1) and wr enable=1

    (b) store the longword address inverted as data in cache data
        control space,

    (c) write the virtual address with the address,

(4) verify that the address in the cache is updated with data
    equal the  noninverted address, and, that, (valid,modified)
    = (1,1) and that (wrt enable,supvsr only) = (1,0).
```

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 0x40000, exp 0x00040000, obs 0xFFFDFFFF
```

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 0x0000010, exp 0xc0000000
```

Cache Write Hit Byte Alignment Test

This test verifies that byte alignment is correct for all byte move, read instructions that are write hits. It performs the operation with the cache enabled and then with the cache disabled for the same write data and compares results.

Error Description

Typical test error messages are:

```
Write byte hit alignment error
cache addr 0x00000000, exp 0x000000FF, obs 0x0000FF00
```

MMU Protection Test

MMU Protection Test

The Cache Write Hit Protect Violation Test verifies that doing a supervisor data operand write hit system memory to a memory block address with cache valid and not write enabled will cause a bus error and a MMU protection violation status in the bus error status register.

The test is performed in the following sequence of steps:

(1) Clear all cache tag field addresses in cache tag control space.

(2) Clear all cache data control space addresses.

(a) clear the cache tag/data entries for the address,

(b) store the address in the cache tag entry,

(c) store the longword address inverted as data in cache data control space,

(d) attempt to write the virtual address and verify that a bus error occurs and no data is written to the protected address.

Error Descriptions

Typical error messages during the test are as follows:

```
A write hit to a write protected address shld cause a bus error
addr 0x40000

Attempting to write a write protected address shld not write it
addr 0x00040000, exp 0xFFFDFFFF, obs 0x00040000

A write protect error shld set prot err status in bus error reg

bus err reg: exp 0x00000040, obs 0x00000000
```

Cache Read Miss Tests

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 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 0x00040000, exp 0x00040000, obs 0xFFFDFFF
```

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.
mem addr 0x00040000, exp 0x00040000, obs 0xFFFDFFFF
```

Another possible error message is:

```
Cache entry not set valid or incorrect tag field
cache addr 0x0000000, exp 0x80000000, obs 0x00000000
```

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

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 0x00040000, obs 0x00000000
```

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.
write addr: 0x00040000, mem addr 0x00040000, exp 0xFFFDFFFF, obs 0x00040000
```

Physical Address Compare Tests

Cache Read (Not Dirty)/Physical Address Compare Test

This test maps all virtual segments (64 Kbytes) in virtual space to the same physical segment in memory, then sets cache tags to be valid and tags for a given virtual address block.

Next, the test reads a modulo 64 Kbyte address from the first virtual address of another virtual segment. Finally, the cache block's tags are updated to be valid with the tag address of the last access.

Cache Read (Dirty)/Physical Address Compare Test

This test first maps all virtual segments (64 Kbytes) in virtual space to the same physical segment in memory. It then sets the cache tags to be valid and dirty and tags for a given virtual address block. Next it reads a modulo 64 KB address from the first virtual address from another virtual segment.

Finally, the test verifies that the following have occurred:

a) the data returned is from the given virtual address cache block,

b) the cache block's tags are updated to be valid with the tag address of the last access.

Cache Read (uncached)/Physical Address Compare Test

This test first maps all virtual segments (64 Kbytes) in virtual space to the same physical segment in memory. It then sets cache tags to be invalid for a given virtual address block. Next, it reads a modulo 64 KB address from the first virtual address from another virtual segment. Finally, it verifies that the following has occurred:

a) the data returned is from the given virtual address cache block,

b) the cache block's tags are updated to be valid with the tag address of the last access.

Cache Writeback Error Tests

Cache Write Miss/Writeback Timeout Error Test

This test first sets up a cache block to attempt a writeback to a nonexistent memory address. It then performs a write miss to an existing memory address that is not cached. This action should cause an attempt to writeback a block to a nonexistent memory address.

Error Descriptions

If no level 15 interrupt occurs the following error message is displayed:

```
A writeback attempt to a nonexistent mem address shld cause a level 15
interrupt

writeback addr 0x8000000
```

If the bus error register does not indicate a writeback timeout error with the common error bit set the following error message is displayed:

```
A writeback to a nonexistent memory address shld set the timeout bit in
the Memory Err Register

mem err reg: exp 0x00000088 obs 0x00000000
```

Cache Read Miss/Writeback Timeout Error Test

This test first sets up a cache block to attempt a writeback to a nonexistent memory address. It then performs a read miss to an existing memory address which is not cached. This should cause an attempt to writeback a block to a nonexistent memory address.

Error Description

If no level 15 interrupt occurs the following error message is displayed:

```
A writeback attempt to a nonexistent mem address shld cause a level 15 interrupt

writeback addr 0x8000000
```

If the bus error register does not indicate a writeback timeout error with the common error bit set the following error message is displayed:

```
A writeback to a nonexistent memory address shld set the timeout bit in the Memory Err Register

mem err reg: exp 0x00000088 obs 0x00000000
```

Cache Write Miss/Writeback Translation Error Test

This test first sets up a cache block to attempt a writeback to an invalid memory page address. Then it performs a write miss to an existing memory address that is not cached, which should cause an attempt to writeback a block to an invalid memory address.

Error Description

If no level 15 interrupt occurs the following error message is displayed:

```
A writeback attempt to an invalid  address shld cause a level 15 interrupt

writeback addr 0x8000000
```

If the bus error register does not indicate a writeback timeout error with the common error bit set the following error message is displayed:

```
A writeback to a invalid memory address shld set the timeout bit in the Memory Err Register

mem err reg: exp 0x00000084 obs 0x00000000
```

Cache Read Miss/Writeback Translation Error Test

This test sets up a cache block to attempt a writeback to an invalid memory page address, then performs a read miss to an existing memory address that is not cached. This should cause an attempt to writeback a block to an invalid memory address.

Error Description

If no level 15 interrupt occurs the following error message is displayed:

```
A writeback attempt to an invalid address shld cause a level 15 interrupt

writeback addr 0x8000000
```

If the bus error register does not indicate a writeback translation error with the common error bit set the following error message is displayed:

```
A writeback to an invalid memory address shld set the writeback invalid bit
in the Memory Err Register

mem err reg: exp 0x00000084 obs 0x00000000
```

Cache Flush (context match) Test

This test sets a longword address pattern throughout memory below the test program and sets all cache entries to valid state with inverse address longword data for all blocks in the cache. One or more cache blocks are then set as valid and dirty. A cache context flush operation is then performed. The test then verifies that all dirty cache entries that match the current context are invalidated and written to memory and that no other memory locations are affected.

Error Descriptions

If one or more of the dirty cache blocks were not written to memory, the following error message is displayed:

```
One or more cache blocks were not flushed to memory.

mem addr 0x00000010, exp 0xFFFFFFEF, obs 0x00000010
```

If one or more of the dirty cache blocks for the current context were not invalidated in the associated cache tags, the following error message is displayed:

```
One or more cache blocks were not invalidated in cache.

tag addr 0x00000010, exp 0x00000000, obs 0xC0000000
```

Cache Flush (segment match) Test

This test sets a longword address pattern throughout memory below the test program and sets all cache entries to valid state. Then, one or more entries for each page address in the cache are set to valid and modified. It then sets, for these entries, inverse address longword data for all blocks in the cache. A cache segment flush operation is then performed. The test then verifies that all dirty cache entries that match the current segment are invalidated and written to memory and that no other memory locations are affected.

Error Description

If one or more of the dirty cache blocks were not written to memory, the following error message is displayed:

```
One or more cache blocks were not flushed to memory.

mem addr 0x00000010, exp 0xFFFFFFEF, obs 0x00000010
```

If one or more of the dirty cache blocks for the current context were not invalidated in the associated cache tags, the following error message is displayed:

```
One or more cache blocks were not invalidated in cache.

tag addr 0x00000010, exp 0x00000000, obs 0xC0000000
```

Cache Flush (page match) Test

The test sets a longword address pattern throughout memory below the test program and sets all cache entries to a valid state. It then sets one or more entries in the cache to valid and modified, and for these entries, sets inverse data in the cache. A cache page flush operation is then performed. The test then verifies that all dirty cache entries that match the current page are invalidated and written to memory and that no other memory locations are affected.

Error Description

If one or more of the dirty cache blocks were not written to memory, the following error message is displayed:

```
One or more cache blocks were not flushed to memory.

mem addr 0x00000010, exp 0xFFFFFFEF, obs 0x00000010
```

If one or more of the dirty cache blocks for the current context were not invalidated in the associated cache tags, the following error message is displayed:

```
One or more cache blocks were not invalidated in cache.

tag addr 0x00000010, exp 0x00000000, obs 0xC0000000
```

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 nops 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.

A.7. 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.

A.8. 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 (a USART)

Tags
Control/status bits within the cache that define status of the block

UART
Universal, Asynchronous, Receiver/Transmitter

VMEbus
Motorola bus interface connecting CPU board with other peripherals

A.9. Standalone ECC Memory Diagnostic

A.10. Introduction

This standalone diagnostic tests the error checking and correction (ECC) logic on Sun products. It resides on the SunDiagnostic Executive tape, but does not run under the Executive program. You must boot it separately, as described under Section A.13 , Loading And Starting . You must determine the tape file number of the diagnostic and enter its hexadecimal equivalent in your boot command.

NOTE *The user interface for this diagnostic is not the same as that for the diagnostics you execute from the Exec Diagnostics menu.*

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 processor only gets the byte, word, or long word requested.

The Error Checking and Correction logic is 8 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.

A.11. Hardware Requirements

The ECC Memory Diagnostic runs on any memory configuration. The memory configuration is recorded in the EEPROM, and 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 checked 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.

A.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-4/2xx implementation of the Sun-4 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-4 architecture. Sun-4/2xx memory also features a self timed refresh controller that may be enabled for memory scrub.

The Sun-4/2xx memory board is interconnected with the CPU board over the 64-bit bus.

Sun-4/2xx 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-4 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.

For Sun-3/200 Boards
: 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-4/2xx 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 128 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 0x200000 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 0x820000 to 0x1020000. 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.

A.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/eccmem4.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 `eccmem4.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/eccmem4.diag
[Return]
```

Finally, the diagnostic can be loaded from local tape. Assuming that the tape contains a bootable image of `eccmem4.diag`, the three command lines listed below can be used to load the diagnostic.

1) If `eccmem4.diag` resides on a SCSI tape, use this command line:

```
b st()
```

2) If `eccmem4.diag` resides on an archive tape, use this command line:

```
b ar()
```

3) If `eccmem4.diag` resides on a tape master, use this command line:

```
b mt()
```

A.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:

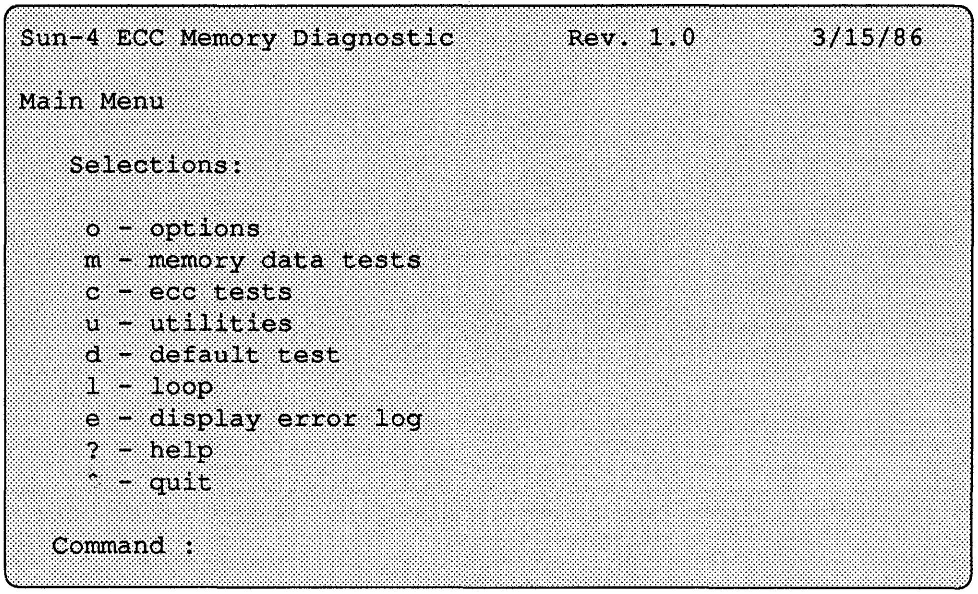
```
d ; l 5
```

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-4 ECC Memory Diagnostic        Rev. 1.0        3/15/86

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 submenu 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 *[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-4 ECC Memory Diagnostic    Rev. 1.0     3/15/86

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]*

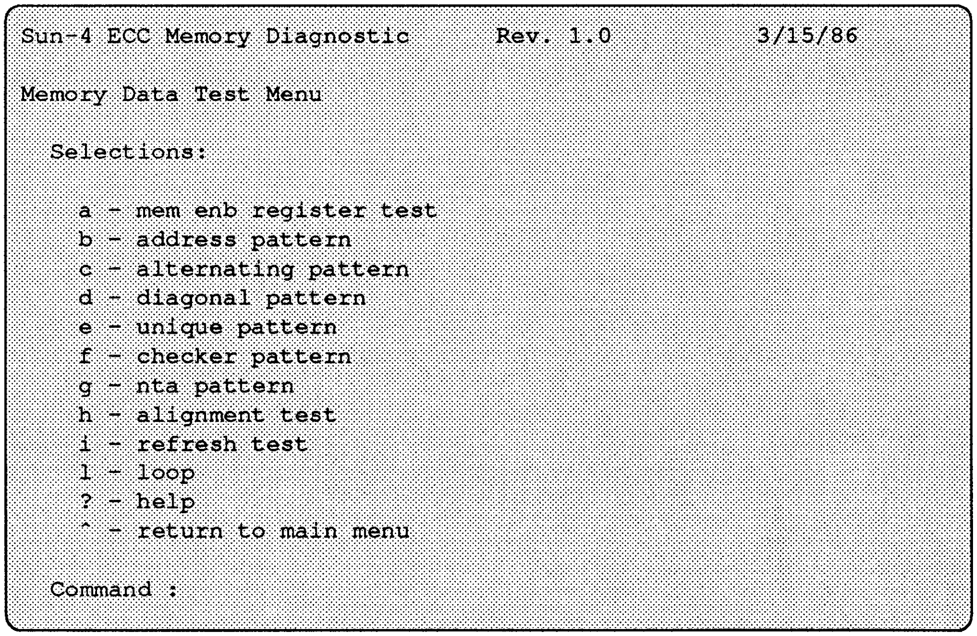
Option e enables or disables the display of all messages to the screen. Only the first 20 error messages are stored and display later if this option is disabled. The default is error messages enabled. The *number* argument may be 0 to disable error message display during execution of each test, or 1 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-4 ECC Memory Diagnostic      Rev. 1.0          3/15/86

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 0's. 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 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.

e *[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 processor's internal cache. If the test is too big to fit in the processor'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.

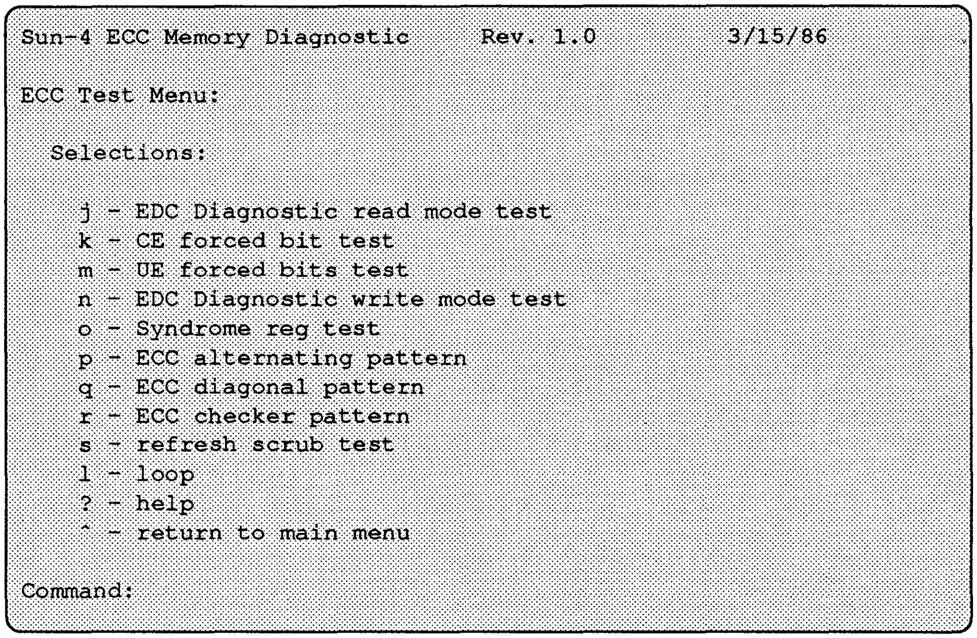
It tests the bcopy interface between the CPU board and the memory board by writing a block pattern into memory and then reading the block out into the 128 bit (16 byte buffer in the cache) then doing a block copy write to memory. It verifies that the data was copied correctly.

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-4 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 "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 invert the data; enter 0 if you do not. The fourth argument, *pascnt* also has been discussed, in the section titled *Memory Data Test.*

To abort the test and return to the test Menu, 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 fourth 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 fourth 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-4 ECC Memory Diagnostic          Rev. 1.0        3/15/86

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.

A.15. Error Handling

ECC Errors

ECC errors are handled separately from bus errors. ECC is now a non-maskable interrupt at Level 15, 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.

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 15) 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
expexpdata obs rddata xor expdata ^ rddata.

test name No CE Mem Cntrl reg failed @ addr
expexpdata obs rddata xor expdata ^ rddata.

test name No CE in Syndrome reg failed @ addr
expexpdata obs rddata xor expdata ^ rddata.
```

Bus Errors

Bus errors are trapped and a message is displayed as follows:

```
unexpected data_access_exception at pc0xsome number
Memory Address: 0xsome number
Instruction: xxxxxxxxxx Bus Error Register Value: 0xsome number
```

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)
```

A.16. Special Problems

At the time of this writing, when exiting from this diagnostic back into the PROM monitor, a `k1` command will cause 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.

A.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
1	o o
	o o
	o o
	0--0
2	o o
	o o
	0--0
	o o
3	o o
	0--0
	o o
	o o
4	0--0
	o o
	o o
	o o

A.18. Recommended Test Procedure

The recommended test procedure for minimal testing of the Sun-4/2xx 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.

A.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

A.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

B

Loopback Connector Pin Designations

B

Loopback Connector Pin Designations

This appendix provides pin assignments for loopback connectors called for in the CPU diagnostic. Please contact Sun customer support for information on a loopback connector kit. For information on HSI board connectors, refer to the installation and configuration manuals that come with the board. For information on RS449 loopback connectors used for the MCP and ALM2 diagnostic, refer to the chapter that describes the diagnostic.

B.1. 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.

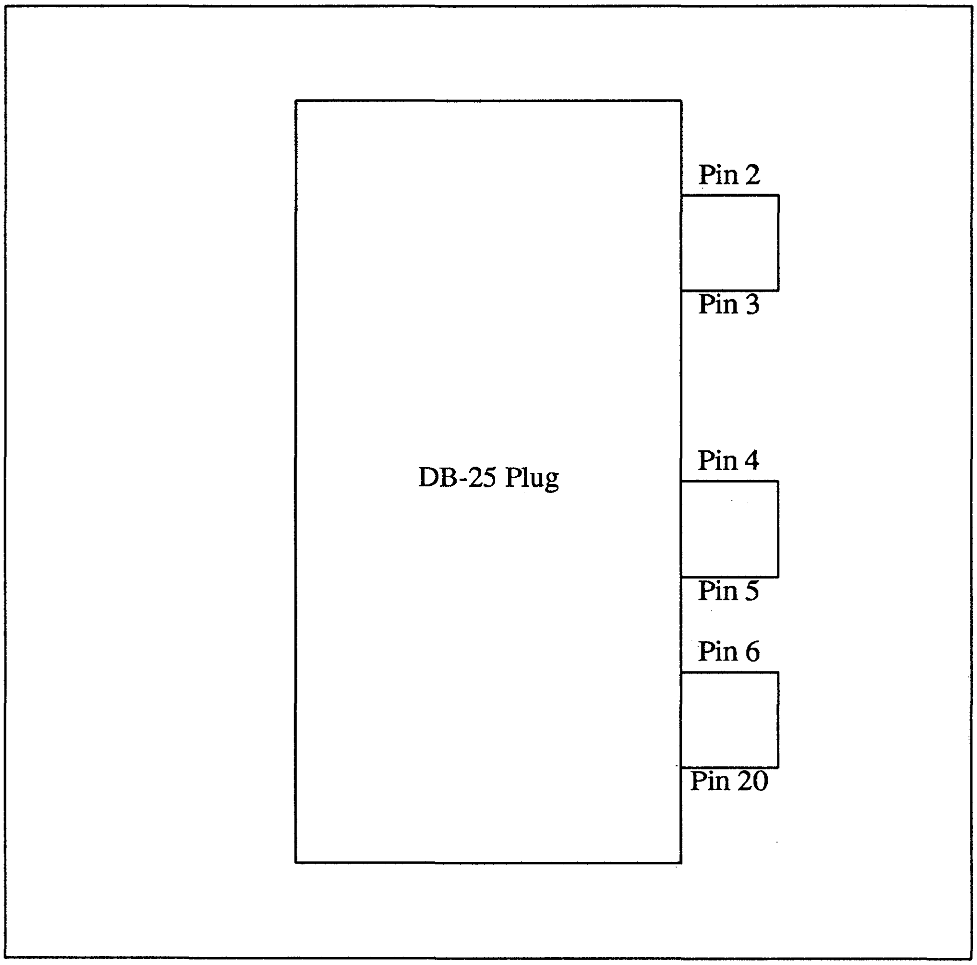
RS-232 Serial Port Connector

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 B-1 *RS-232 Loopback Connector*



9-pin Loopback Connector

Some systems now have 9-pin B, C and D Serial Ports. For those systems, the pin assignments are as follows:

(RXD) 2 <---------> 3 (TXD)

(DTR) 4 <---------> 6 (DSR)

(RTS) 7 <---------> 8 (CTS)

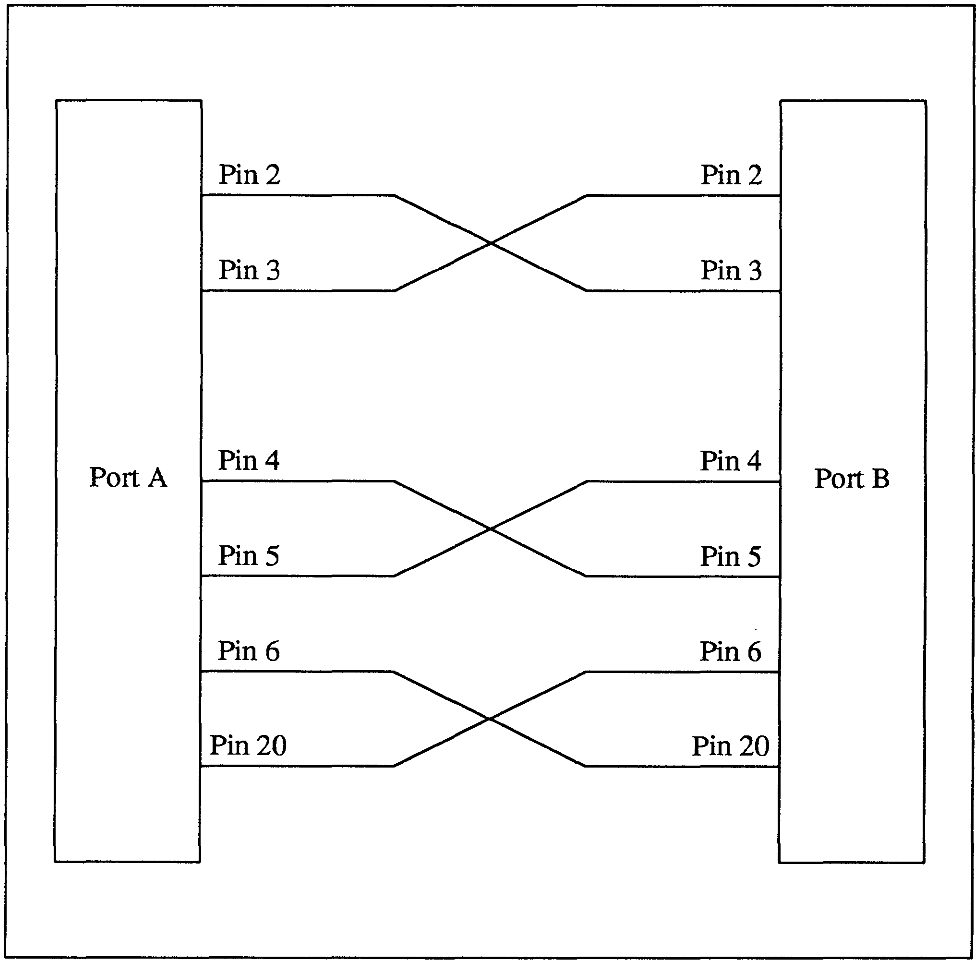
RS-232 Loopback Cable

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 B-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.*

Ethernet Loopback Connector

The Ethernet external loopback connector pin connections are as follows:

From Pin	***To Pin***
3	5
10	12
13	14

C

Diagnostic Executive Bug Report Form

C

Diagnostic Executive Bug Report Form

C.1. Overview

Use this form to report bugs found in the executive programs. To report a Sun-Diagnostic 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 this error occur on other systems?

Describe the Environment

Carefully describe the hardware environment of the failing system. 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.

C.2. Where to Send the Report

When you have completed this form, please mail it to the following address.

```
Attn: Customer Support Diagnostics
      Mail Stop M7-38
      Sun Microsystems
      2550 Garcia Avenue
      Mountain View, CA
            94043
```

If you have access to the UUCP network you may use electronic mail to send this report. If you do not have a support contract with Sun, send the information to *sun!bugs*. If you are under support contract, send this information to *sun!hotline*. Please try to conform to the format of this form as closely as possible.

C.3. Who We Can Contact

Please provide the name and telephone number of someone we can reach to learn more about the problem. It is also useful to have your account number for our records.

Contact

Name: ______________________________

Phone: ______________________________

Account

Name: ________________________________

Number: ______________________________

C.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 system:

____

Serial Number of System:

____

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 documentation

□ 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

____

____

____

____

____

____

____

____

____

____

____

Revision History

Dash Number	Date	Comments
50	24 February 89	Beta 68020,68030,SPARC Exec 1.2 Manual
A	30 June 89	FCS 68020,68030,SPARC Exec 1.2 Manual