HP Diagnostics Guide

V2500/V2600 Servers

First Edition

A5824-96002

HP Diagnostics Guide: V2500/V2600 Servers

Customer Order Number: A5824-90002

December 1999

Printed in: USA

Revision History

Edition: First

Document Number: A5824-90002
Remarks: Initial release. December, 1999.

Notice

Contents

Figures

Tables

Preface

This document describes the offline diagnostics for V2500 and V2600 servers. It is not intended to be a tutorial or troubleshooting guide but a reference guide that contains information on all utilties and scripts used to troubleshoot these systems.

Notational conventions

This section describes notational conventions used in this book.

`bold monospace`	In command examples, **`bold monospace`** identifies input that must be typed exactly as shown.
`monospace`	In paragraph text, `monospace` identifies command names, system calls, and data structures and types. In command examples, `monospace` identifies command output, including error messages.
italic	In paragraph text, *italic* identifies titles of documents. In command syntax diagrams, *italic* identifies variables that you must provide. The following command example uses brackets to indicate that the variable *output_file* is optional: `command` *input_file* [*output_file*]

Brackets ([ ])	In command examples, square brackets designate optional entries.
Curly brackets ({}), Pipe (|)	In command syntax diagrams, text surrounded by curly brackets indicates a choice. The choices available are shown inside the curly brackets and separated by the pipe sign (|). The following command example indicates that you can enter either `a` or `b`: `command {a | b}`
Keycap	**Keycap** indicates the keyboard keys you must press to execute the command example.

NOTE A note highlights important supplemental information.

CAUTION A caution highlights procedures or information necessary to avoid damage to equipment, damage to software, loss of data, or invalid test results.

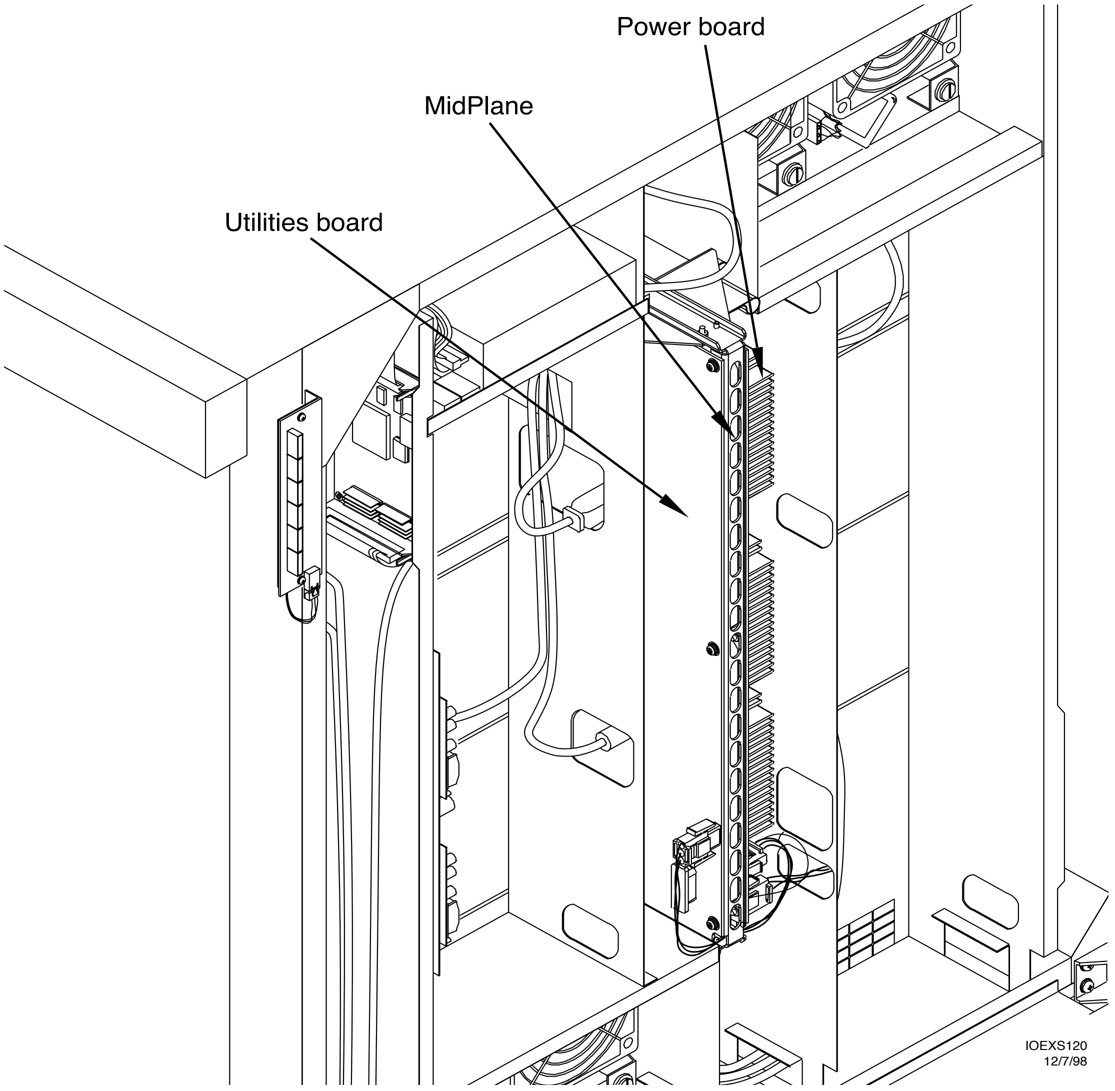
1 Introduction

This chapter presents an overview of the diagnostic mechanism for V2500/V2600 servers.

Utilities board

The diagnostic mechanism in the V2500 and V2600 servers is centered around the Stingray Core Utilities board (SCUB). The SCUB is mounted under the MidPlane Interconnect board (MIB) toward the front of the system. See Figure 1.

Figure 1 Location of the Utilities board



The following devices connect to the Utilities board:

- Core logic bus
- Environmental sensors
- Test points
- Liquid crystal display (LCD)
- Attention lightbar
- SSP

  The Service Support Processor (SSP) connects to the system via the ethernet and RS232 connections. The SSP is used for configuring, monitoring, testing, and error logging and is not required for normal operation of a node. A system will boot and operate without a SSP, and failure of the SSP will not cause interruption of the system.

Figure 2 shows the Utilities board functional layout.

The following hardware components comprise the Utilities board:

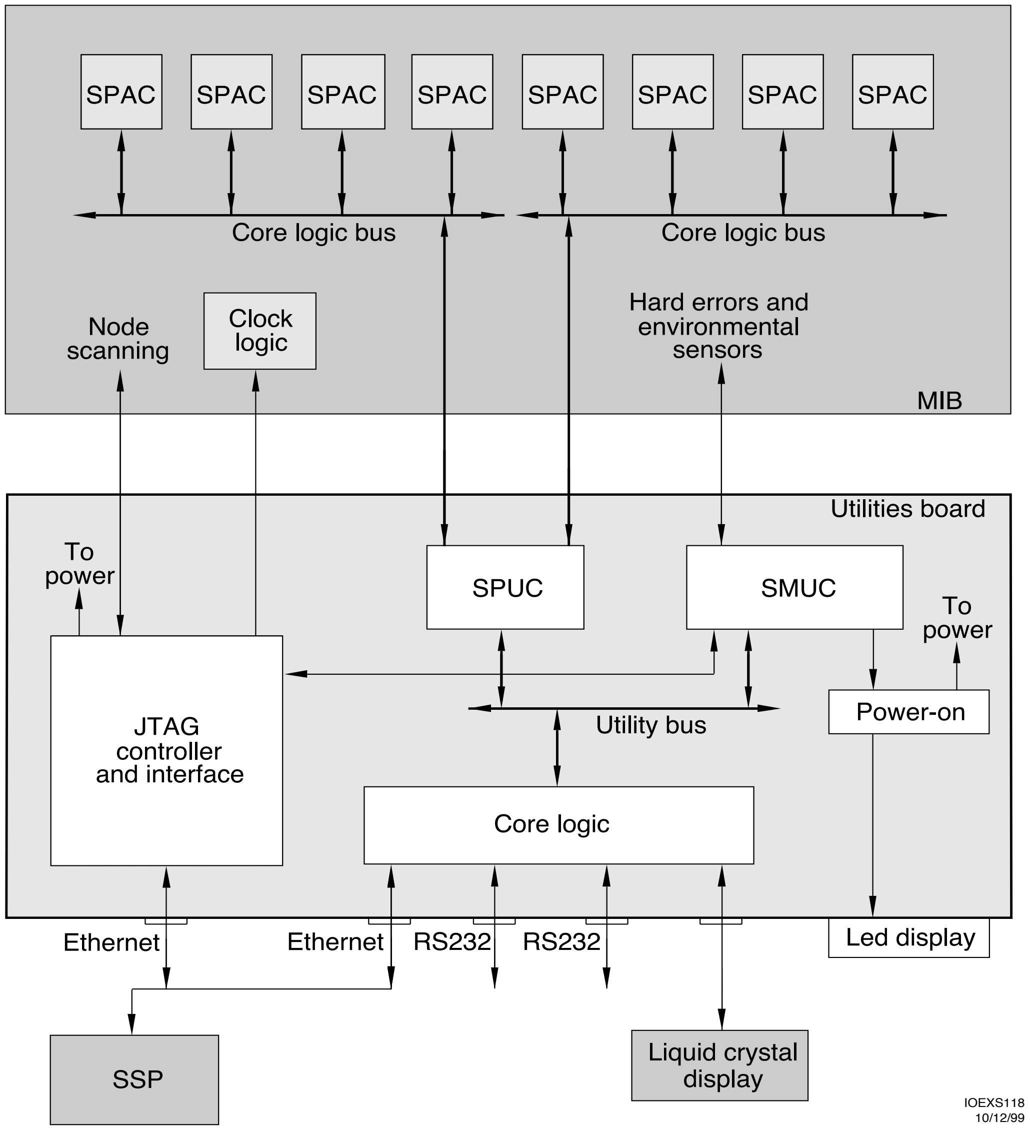
- Core logic—Contains initialization, booting firmware, controller for ethernet and RS-232 interface, and various memories.
- Stingray Monitor Utilities controller (SMUC)—Collects environmental interrupts.
- Power-On circuit—Controls powering up the entire system.

  Environmental sensors are located throughout the system and connect to the SMUC. The SMUC latches interrupts from these sensors as well as other interrupts. The SMUC and the power-on circuit together control system power-up. The power-on circuit drives the attention lightbar diagnostic display through which the operator can determine power-on status.
- Stingray Processor Utilities controller (SPUC)—Interfaces to the core logic bus.

  The SPUC connects to the two core logic buses. Each bus connects up to four Stingray Processor Agent Controllers (SPACs).
- JTAG (Joint Test Action Group) interface—Supports a SSP for running diagnostics. The V2500/V2600 servers use a test method called scanning to test boards and other hardware units.

The microprocessor-controlled JTAG interface captures incoming command packets and sends out scan information packets across the ethernet connection to the SSP. Through the SSP connection, one can read and write every CSR in the system.

Figure 2 Utilities board



Core logic

The core logic contains initialization and booting firmware and is described in the following sections.

Flash memory

The core logic contains a four-MByte electrically erasable programmable read only memory (EEPROM) storage for Processor-Dependent Code (PDC). PDC consists of Power-On Self Test (POST) and Open Boot PROM (OBP). The V2500/V2600 server uses these two components plus additional firmware called `spp_pdc` that is laid over OBP and interfaces OBP to HP-UX. Flash memory also contains all diagnostic test, utilities, and scripts.

Flash memory is configured as 512-KByte addresses by 32 data bits with only 32-bit read and write accesses allowed. EEPROM devices are used for flash memory so that it may be rewritten for field upgrades. It can also be written when the SPUC is *scanned*.

Nonvolatile static RAM

The core logic section contains a nonvolatile battery-backed 128-Kbyte RAM (NVRAM) for storing system log and configuration information. This RAM is byte addressable and can be accessed even after power failures.

DUART

A Dual Universal Asynchronous Receiver-Transmitter (DUART) provides to RS232 serial ports and a single parallel port. One serial port provides an interface to a terminal used as a local console to analyze problems, reconfigure the system, and provide other user access.The parallel port of the DUART drives the LCD. The second RS-232 port can be used for a modem for field service.

RAM

Random access memory (RAM) provides support for the core system functions. When the system powers up, the processors operate out of this RAM to run self test and configure the rest of the node. Once the system is fully configured, the processors execute out of main memory. The RAM is byte addressable and is 512 KBytes, configured as 128-KByte addresses by 32 data bits.

Console ethernet

The ethernet I/O port provides a connection to the SSP over LAN1.

Attention lightbar

The attention light bar displays environmental information, such as the source of an environmental error that caused the Utilities board to power down the node.

LCD

The liquid crystal display provides basic system information. The core logic drives the LCD through the parallel port on the DUART. The attention lightbar and LCD are detailed in “System displays” on page 12.

COP interface

A serial EEPROM (referred to as COP chip) is located on major boards with information such as serial number, assembly revision, wire revision, truncated board part number, and so on. The SMUC connects to the COP bus selector (CBS) chip on the MIB allowing each COP chip in a node to be read.

SPUC

The SPUC provides interrupts and error messages to and receives control messages from the processors through two 18-bit, bidirectional buses. Each bus connects up to four SPACs. The SPUC also provides core logic bus arbitration for the processors.

SMUC and Power-on

The SMUC registers system environmental parameters. It connects to the utilities bus so that processors can monitor the node by accessing the appropriate CSRs. The SMUC works in conjunction with the power-on circuit to power up the entire system, and it can operate when the rest of the node is powered off or in some indeterminate state.

The SMUC drives the environment LCD display. The SSP can also read the environmental LCD display using the `sppdsh` utility. See “sppdsh” on page 322.

SMUC environmental monitoring

The following environmental conditions are monitored:

- ASIC installation error sensing
- FPGA configuration and status
- Thermal sensing
- Fan Sensing
- Power failure sensing
- 48-V failure
- 48-V maintenance
- Ambient air temperature sensing.
- Power-on

Table 1 **Environmental conditions monitored by the SMUC and power-on circuit**

Condition	Type	Action
ASIC Not Installed OK	Environmental error	Power not turned on, LED indication
FPGA not OK	Environmental error	Power not turned on, LED indication
48-V Fail	Environmental error	Power turned off, LED indication
MIB power fail	Environmental error	Power turned off, LED indication
Board over temp	Environmental error	Power off in one second, LED indication interrupt
Fan not turning	Environmental error	Power off in one second, LED indication interrupt
Ambient air hot	Environmental error	Power off in one second, LED indication interrupt
Other power fail	Environmental error	Power off in one second, LED indication interrupt

Condition	Type	Action
Ambient air warm	Environmental warning	LED indication, interrupt
48-Volt maintenance	Environmental warning	LED indication, interrupt
Hard error	Hard error	LED indication, interrupt

Environmental condition detected by power-on function

The power-on function detects environmental errors (such as ASIC Not Installed OK or FPGA Not OK). It does not turn on power to the node until the conditions are corrected. It also detects environmental errors such as 48-V Fail while the system is powering up and MIB Power Fail after the system has powered up. If a failure is detected in these two cases, the power-on circuit turns off power to the system.

Environmental warnings such as 48-Volt maintenance are also detected by the power-on circuit.

In all cases, the power-on circuit sets an environmental attention light bar code. The code is prioritized so that it displays the highest priority error or warning. See “Attention light bar” on page 16 for a list of codes.

Environmental conditions detected by SMUC

The SMUC detects most of the environmental conditions. It samples error conditions during a time period derived from a local 10-Hz clock that drives the power-on circuit. It registers all the environmental error conditions twice and then logically ORs them together. If the conditions persist for 200 mS, the environmental error bit is set, and an environmental error interrupt is sent to the SPUC, which sends it on to the processors. The SMUC then waits 1.2 seconds and commands the power-on circuit to power down the system.

This same procedure exists for an environmental warning, except that an environmental warning interrupt is sent and the power-on circuit does not power down the system.

The environmental error interrupt and the 1.2 second delay provide the system adequate time to read CSRs to determine the cause of the error, log the condition in NVRAM, and display the condition on the attention lightbar.

After the system is powered down, the Utilities board is still powered up, but all outputs are disconnected from the system.

Environmental control

The Utilities board performs the following functions to control the node environment.

Power-on

When the power switch is turned on, the outputs of the 48-Volt power supplies become active. Several hundred milliseconds after the Utilities board 5-Volt supply reaches its nominal level, the power-on circuit starts powering up the other DC-to-DC converters of the node in succession.

The power-on circuit does not power up the node if an ASIC is installed incorrectly. It keeps the system powered up unless an environmental condition occurs that warrants a power-down.

Voltage margining

Voltage margin is divided into four groups called quadrants. The user can margin quadrants separately. When setting the upper margin, for example, all boards in that quadrant are margined for upper.

Clock margining

Parallel ports on the core logic microprocessor select the nominal, upper, or external clock that drives the node.

JTAG interface

The JTAG interface supports a SSP and a mechanism to fanout JTAG to all the boards in a node. It is used only for testing.

JTAG functions are described in the following sections.

SSP interface

The SSP can be a PA-RISC based workstation. The interface to the SSP is an ethernet AUI port for flexibility in connecting to many workstations. It is also easily expandable.

DC test of a node

To perform the `DC test`, the Test Bus Controller (TBC) first scans data to all boards in a node. Then each JTAG device performs a capture step that completes the movement of the test data from the driver to the receiver. This step is described in the JTAG 1149.1 specification.

AC test of a node

To perform the `AC test,` the Test Bus Controller (TBC) scans data to all boards in a node and then loads an `AC test` instruction into all ASICs on one board at a time. The scan ring on each board is paused.

Once all boards have been loaded with the `AC test` instruction, the TBC takes all boards out of pause mode simultaneously, causing them all to exit update together and execute the `AC test`.

The `AC test` enables clocks inside the ASICs so that they test internal and external paths at the system clock rate. They all execute on the same system clock.

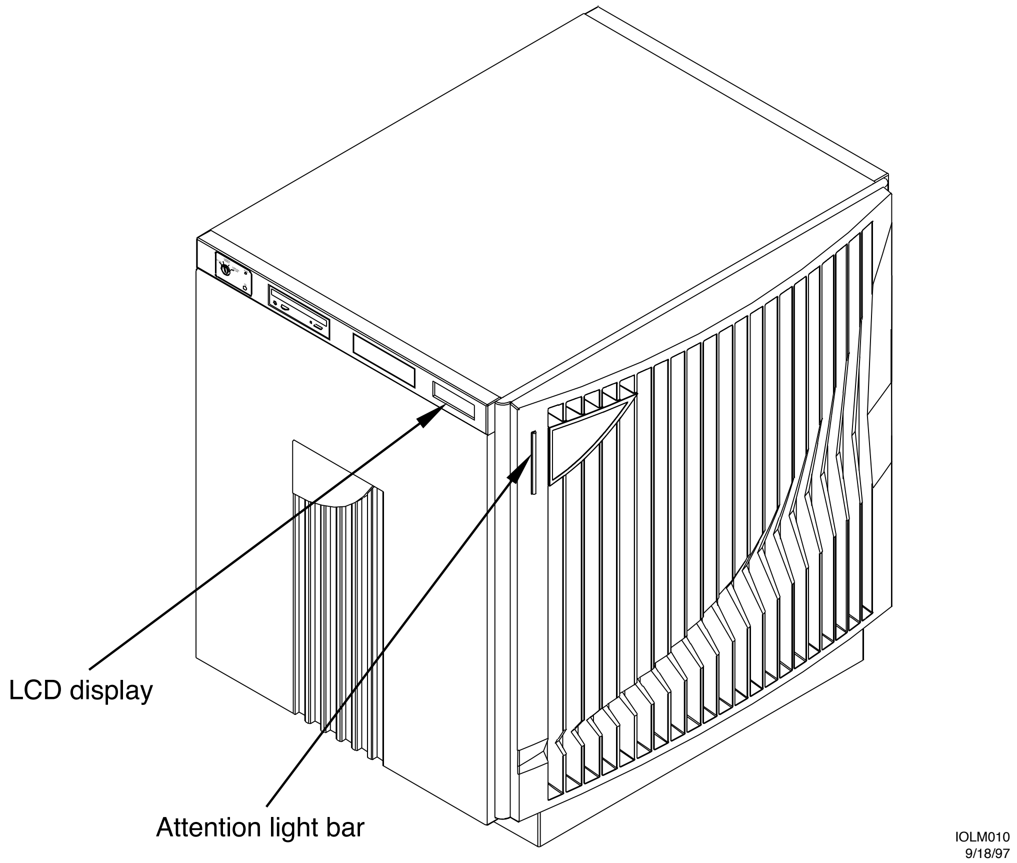
JTAG fanout

The SSP interface is thin ethernet. In addition to the SSP, this port is also used for the console ethernet. There is one cable that connects to all the nodes and to the SSP (if it exists) and to whatever device or network that will display the console.

System displays

The V2500/V2600 server provides two means of displaying status and error reporting: an LCD and an Attention light bar.

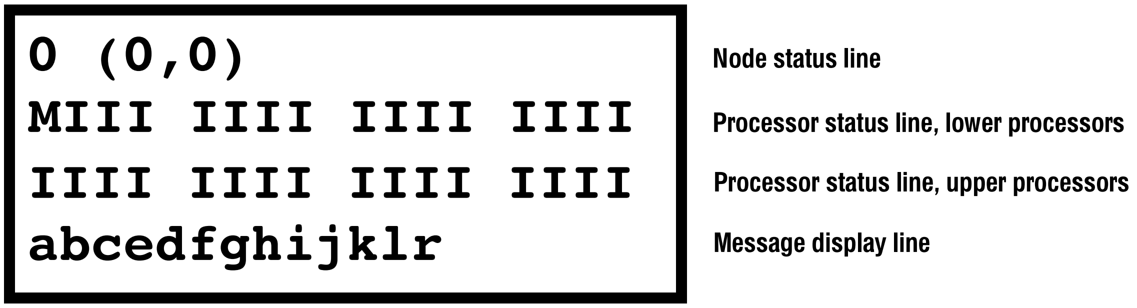
Figure 3 System displays



Front panel LCD

The front panel is a 20-character by 4-line liquid crystal display as shown in Figure 4.

Figure 4 **Front panel LCD**



When the node key switch is turned on, the LCD powers up but is initially blank.

Node status line

The Node Status Line shows the node ID in both decimal and X, Y topology formats.

Processor status line

The processor status line shows the current run state for each processor in the node. Table 2 shows the initialization step code definitions and Table 3 shows the run-time status codes. The M in the first processor status line stands for the monarch processor.

Table 2 **Processor initialization steps**

Step	Description
0	Processor internal diagnostic register initialization
1	Processor early data cache initialization.
2	Processor stack SRAM test.(optional)
3	Processor stack SRAM initialization.
4	Processor BIST-based instruction cache initialization.

Step	Description
5	Processor BIST-based data cache initialization
6	Processor internal register final initialization.
7	Processor basic instruction set testing. (optional)
8	Processor basic instruction cache testing. (optional)
9	Processor basic data cache testing. (optional)
a	Processor basic TLB testing (optional)
b	Processor post-selftest internal register cleanup. (optional)

Table 3 Processor run-time status codes

Status	Description
P	POST interactive: processing interactive console input.
R	RUN: Performing system initialization operations.
I	IDLE: Processor is in an idle loop, awaiting a command.
M	MONARCH: The main POST initialization processor.
H	HPMC: processor has detected a high priority machine check (HPMC).
T	TOC: processor has detected a transfer of control (TOC).
S	SOFT_RESET: processor has detected a soft RESET.
D	DEAD: processor has failed initialization or selftest.
d	DECONFIG: processor has been deconfigured by POST or the user.
-	EMPTY: Empty processor slot.
?	UNKNOWN: processor slot status in unknown.

Message display line

The message display line shows the POST initialization progress. This is updated by the monarch processor. The system console also shows detail for some of these steps. Table 4 shows the code definitions.

Table 4 **Message display line**

Message display code	Description
a	Utilities board (SCUB) hardware initialization.
b	Processor initialization/selftest rendezvous.
c	Utilities board (SCUB) SRAM test. (optional)
d	Utilities board (SCUB) SRAM initialization.
e	Reading Node ID and serial number.
f	Verifying non-volatile RAM (NVRAM) data structures.
g	Probing system hardware (ASICs).
h	Initializing system hardware (ASICs).
i	Probing processors.
j	Initialing, and optionally testing, remaining SCUB SRAM.
k	Probing main memory.
l	Initializing main memory.
m	Verifying multi-node hardware configuration.
n	Multinode initialization starting synchronization.
o	Multinode hardware initialization.
p	Multinode hardware verification.
q	Multinode initialization ending synchronization.
r	Enabling system error hardware.

Power supply indicators

When the keyswitch on the operator panel is in the **DC ON** position both the AC power (amber) LED and the DC power (green) LED on each of the power supplies should be on.

Attention light bar

The Attention light bar is located at the top left corner on the front of the V2500/V2600 server as shown in Figure 3 on page 12. The light bar displays system status in three ways:

- OFF—dc power is turned off. Either the key switch or the side circuit breaker is in the off position.
- ON—Both the side circuit breaker and the keyswitch are in the on position and no environmental warning, error, or hard error exists.
- Flashing—There is an environmental error, warning, or hard error condition. Also indicates scanning during diagnostic execution.

NOTE The light bar flashing during initial start up does not indicate a fault.

The types of environmental conditions that are monitored include:

- ASIC installation error sensing
- ASIC configuration or status
- 48V failure

NOTE 48V failures are cleared only after a power cycle.

- Power failure sensing
- Fan sensing
- Thermal sensing

Types of environmental control functions monitored include:

- Power-on
- Voltage margining (SSP interface)

Environmental errors

Environmental errors are detected by two basic systems in the V2500/ V2600 server: Power-On and Environmental Monitor Utility Chip (MUC).

Power-On detected errors such as ASIC install or ASIC not OK are detected immediately and will not allow dc power to turn on until that condition is resolved.

MUC detected errors such as Ambient Air Hot allows the dc power to turn on for approximately 1.2 seconds before the dc power is turned off. If two or more fans fail simultaneously, the MUC will shut off dc power. Other MUC detected errors such as Ambient Air Warm will flash the LED and not turn off dc power.

Error codes may be viewed by using the SSP utility command `pce` to read the status of the CUB. However, this feature will only work after database generation is complete, not before.

Using the SSP utility man LEDs to decode the CUB status nibbles.

The current environmental temperature set-points are:

- Warm = 32 degrees Celsius (89.6 degrees Fahrenheit)
- Hot = 37 degrees Celsius (98.6 degrees Fahrenheit)

Displaying the SCUB LED values using pce

Use the `sppdsh` command `pce` to display the value of the LEDs on the SCUB.

Step 1. Bring up the `sppdsh` prompt at a sppuser window by entering:

```
$ sppdsh
```

Step 2. Use the `pce` command to display the LED values for all nodes, enter:

```
sppdsh: pce all
```

```
Node      IP address Clocks      LEDS   @C U SHPT Supply1 Supply2 Supply3 Supply4
------------------ ------ --------- ---- ------ ------- ------- ------- -------
  0   15.99.111.116 Normal   0x00       25 1 0000 Nominal Nominal Nominal Nominal
  2   15.99.111.117 Normal   0x00       25 1 0000 Nominal Nominal Nominal Nominal
```

For more information about the `pce` command see the sppdsh man page.

Step 3. Decode the LED values using Appendix A, “LED codes” .

Identifying a node with the blink command

The blink command is used to physically identify a node. This command forces the node attention light bar to blink or turns off blinking, provided an error does not exist on the node.

Step 1. Bring up the `sppdsh` prompt at a sppuser window by entering:

```
$ sppdsh
```

Step 2. Use the `blink` command to cause the attention light bar to blink on a specific node by entering the `blink` command followed by the node number. For example:

```
sppdsh: blink 0
```

For more information about the `blink` command see the sppdsh man page.

Step 3. After you have physically identified the node cause the attention light bar to return to a steady state by entering:

```
sppdsh: blink 0
```

LED display

The Attention LED on the core utilities board (CUB) turns on, and the Attention light bar on the front of the node flashes to indicate the presence of an error code listed Table 98. Additionally, only the highest priority error is displayed. Once remedied, an error that is cleared may expose a lesser priority error.

2 Configuration management

The SSP allows the user to configure a node using the `ts_config` utility. `ts_config` sets up the SSP to communicate with the node. The SSP daemon, `ccmd`, monitors the node and reports back configuration information, error information and general status. `ts_config` must be run before using `ccmd`.

Two additional utilities, `sppdsh` and `xconfig`, allow reading or writing configuration information and changing it. Another utility covered in this chapter, `report_cfg`, provides configuration reports. OBP can also be used to modify the configuration. For information concerning `sppdsh`, see “sppdsh” on page 322.

SSP

The SSP is used for configuring, monitoring, testing, and error logging. It is not required for normal operation of a node.

The SSP communicates with the JTAG interface in the nodes. The JTAG port remains idle if no SSP is connected to it. It receives communications packets, interprets requests, and generates responses to them. The hardware on the node can read board information, system configuration, device revisions, and environmental conditions. When a SSP is present, all of these parameters are read or written by the configuration management tools.

The configuration management daemon, `ccmd`, initiates communications between the SSP and the nodes.

ts_config

```
ts_config [-display display name]
```

Any V2500/V2600 nodes added to the SSP must be configured by `ts_config` to enable diagnostic and scan capabilities, environmental and hard-error monitoring, and console access.

Once the configuration for each node is set, it is retained when new SSP software is installed.

`ts_config` tasks include:

- Configuring a node—Adding and removing a node to the SSP configuration
- Configuring the terminal mux—Configuring and removing the terminal mux on the SSP
- Installing a node—Upgrading JTAG firmware, configuring a node scub_ip address, and resetting a node
- Configuration of Multiple-node complex—Configuring V2500/V2600 nodes into a single complex and splitting V2500/V2600 nodes out of a multiple-node complex.
- Operational support—Resetting a V2500/V2600 node or multiple-node complex and starting console sessions.

The user must have root privilege to configure a node or the terminal mux, because several HP-UX system files are modified during the configuration.

Starting ts_config

To start `ts_config` from the SSP desktop, click on an empty area of the background to obtain the Workspace menu and then select the `ts_config (root)` option. Enter the root password.

To start `ts_config` from a shell (local or remote), ensure that the DISPLAY environment variable is set appropriately before starting `ts_config`.

For example:

```
$ DISPLAY=myws:0; export DISPLAY     (sh/ksh/sppdsh)
% setenv DISPLAY myws:0       (csh/tcsh)
```

Also, the `-display` start-up option may be used as shown below:

For example:

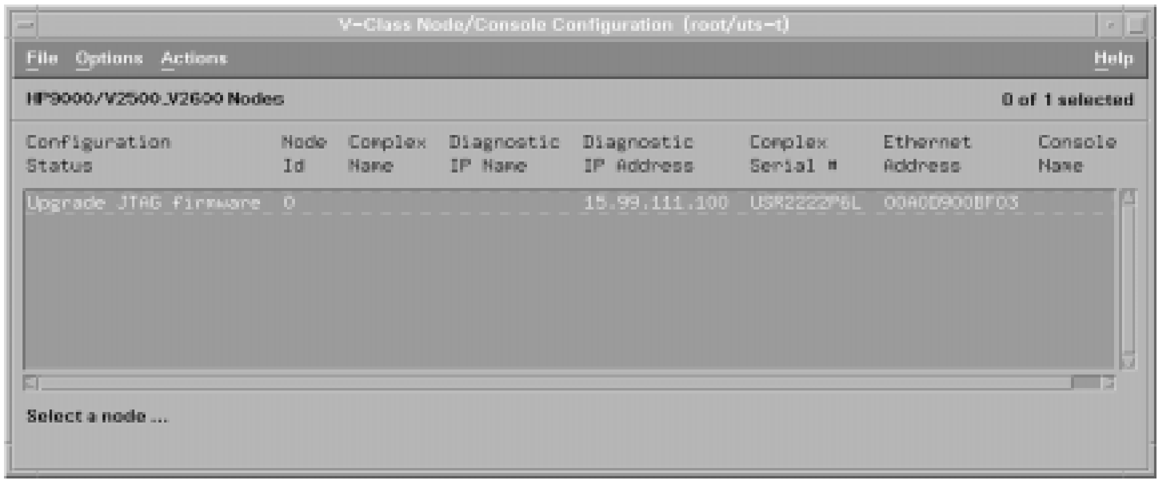
```
# /spp/bin/ts_config  -display myws:0
```

NOTE For shells that are run from the SSP desktop, the DISPLAY variable is set (at the shell start-up) to the local SSP display.

ts_config operation

The `ts_config` utility displays an active list of nodes that are powered up and connected to the SSP diagnostic LAN. The operator selects a node and configures the selected node. A sample display is shown below.

Figure 5 **`ts_config`** sample display



V-Class Node/Console Configuration (root/uts-t)

HP9000/V2500_V2600 Nodes   0 of 1 selected

Configuration Status	Node Id	Complex Name	Diagnostic IP Name	Diagnostic IP Address	Complex Serial #	Ethernet Address	Console Name
Upgrade JTAG firmware	0			15.99.111.100	USR2222P6L	00A0D900BF03	

Select a node ...

The window has three main parts: the drop-down menu bar, the display panel, and the message panel. The display panel contains a list of nodes and their status. To select a node, click with the left-mouse button the line containing the desired node entry in the list. When a node is selected, information about that node is shown in the message panel at the bottom of the `ts_config` window. If an action needs to be performed to configure the node, specific instructions are included.

`ts_config` automatically updates the display when it detects either a change in the configuration status of any node or a newly detected node. The node display is not updated while an Action is being processed or while the user is entering information into an Action dialog.

The upper right corner of the `ts_config` window indicates whether a node has been selected.

The `ts_config` window title includes in parenthesis the name of the effective user ID running `ts_config`, either root or sppuser along with the host name of the SSP.

The `ts_config` display shows the configuration status of the nodes. Table 5 shows the possible status values.

Table 5 **`ts_config`** status values

Configuration Status	**Description**	**Action Required**
Upgrade JTAG firmware	The version of JTAG firmware running on the SCUB does not support the capabilities required to complete the node configuration process.	Select the node and follow the instructions given at the bottom of the ts_config window. `ts_config` guides the operator through the JTAG firmware upgrade procedure.
Not Configured	`ts_config` has detected the node on the Diagnostic LAN and the JTAG firmware is capable of supporting the node configuration activity and the node needs to be configured.	Select the node and follow the instructions given at the bottom of the `ts_config` window. `ts_config` guide the operator through the node configuration procedure described later in this document.

Configuration Status	Description	Action Required
Active	The node is configured and answering requests on the Diagnostic LAN.	None required. This is the desired status.
Inactive	The SSP node configuration file contains information about the specified node, but the node is not responding to requests on the Diagnostic LAN.This status is also shown if a node was configured and then removed from the SSP LAN without being deconfigured.	Power-up the node and/or check for a LAN connection problem. If the node information shown is for a node that has been removed, select the node then select “Actions,” “Deconfigure Node,” and click “Yes.”
Node Id changed	The node is configured and answering requests on the diagnostic LAN, but the node ID currently reported by the node does not match the SSP configuration information.	Select the node to obtain additional information. If the node COP information was changed to a different node ID and the new node ID is correct, select “Actions,” “Configure Node,” then click “Configure.” The SSP configuration information is updated using the new node ID.

Configuration procedures

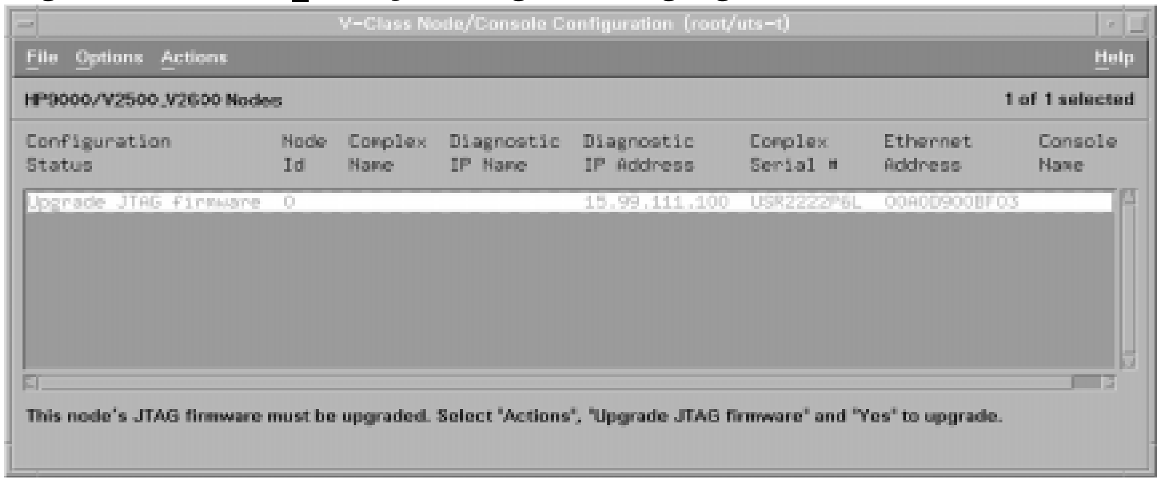
The following procedures provide additional details about each configuration action and are intended as a reference. `ts_config` automatically guides the user through the appropriate procedure when a node is selected.

Upgrade JTAG firmware

NOTE If the node shows “Not Configured,” do not perform this procedure. Perform the following procedure only when the status shows “Upgrade JTAG firmware.”

Step 1. Select the node from the list in the display panel. For example, clicking on node 0 in the list highlights that line as shown in Figure 6.

Figure 6 **`ts_config`** showing node 0 highlighted



HP9000/V2500_V2600 Nodes 1 of 1 selected

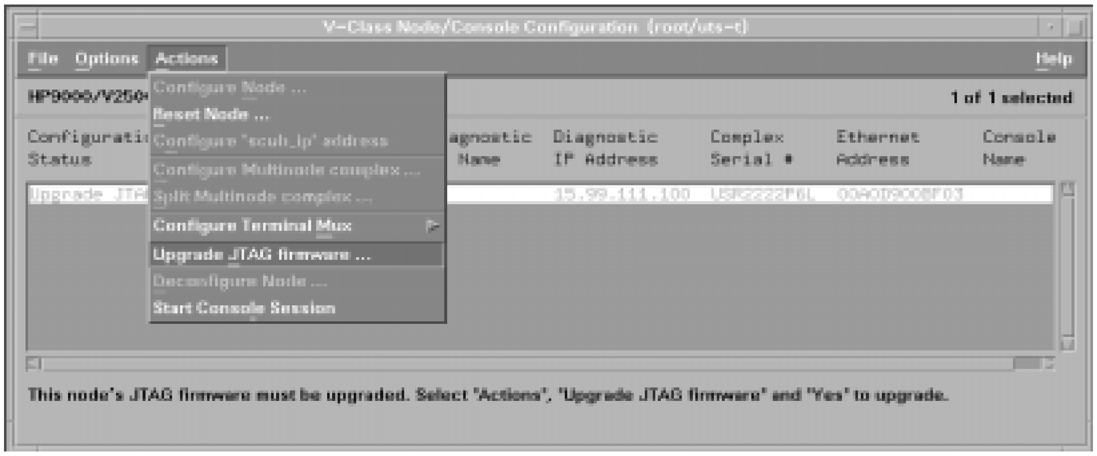
Configuration Status	Node Id	Complex Name	Diagnostic IP Name	Diagnostic IP Address	Complex Serial #	Ethernet Address	Console Name
Upgrade JTAG firmware	0			15.99.111.100	USR2222P6L	00A0D9008F03	

This node's JTAG firmware must be upgraded. Select "Actions", "Upgrade JTAG firmware" and "Yes" to upgrade.

Notice that after the node has been highlighted that `ts_config` displays information concerning the node. In this step, it tells the user what action to take next, "This node's JTAG firmware must be upgraded. Select "Actions," "Upgrade JTAG firmware" and "Yes" to upgrade."

Step 2. Select "Actions" to drop the pop-down menu and then click "Upgrade JTAG firmware," as shown in Figure 7.

Figure 7 **`ts_config`** "Upgrade JTAG firmware" selection.


V-Class Node/Console Configuration (root/uts-t)
File
Options
Actions
Help
Configure Node ...
Reset Node ...
Configure 'scub_ip' address
Configure Multinode complex ...
Split Multinode complex ...
Configure Terminal Mux
Upgrade JTAG firmware ...
Deconfigure Node ...
Start Console Session

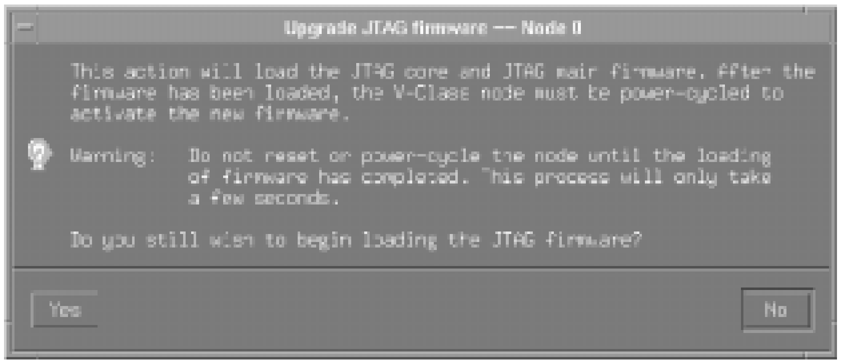
HP9000/V250 1 of 1 selected

Configurati Status		agnostic Name	Diagnostic IP Address	Complex Serial #	Ethernet Address	Console Name
Upgrade JTA			15.99.111.100	USR2222P6L	00A0D9008F03	

This node's JTAG firmware must be upgraded. Select "Actions", "Upgrade JTAG firmware" and "Yes" to upgrade.

Step 3. A message panel appears as the one shown in Figure 8. Read the message. If this is the desired action, click "Yes" to begin the upgrade.

Figure 8 **Upgrade JTAG firmware confirmation panel**



Upgrade JTAG firmware -- Node 0

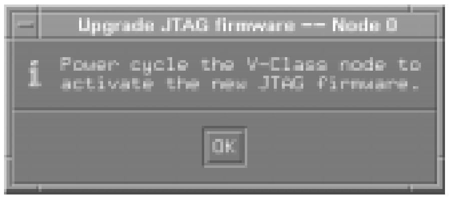
This action will load the JTAG core and JTAG main firmware. After the firmware has been loaded, the V-Class node must be power-cycled to activate the new firmware.

Warning: Do not reset or power-cycle the node until the loading of firmware has completed. This process will only take a few seconds.

Do you still wish to begin loading the JTAG firmware?

Step 4. After the firmware is loaded a panel appears as the one shown in Figure 9. Click "OK" and then power-cycle the node to activate the new firmware.

Figure 9 **`ts_config` power-cycle panel**



Upgrade JTAG firmware -- Node 0

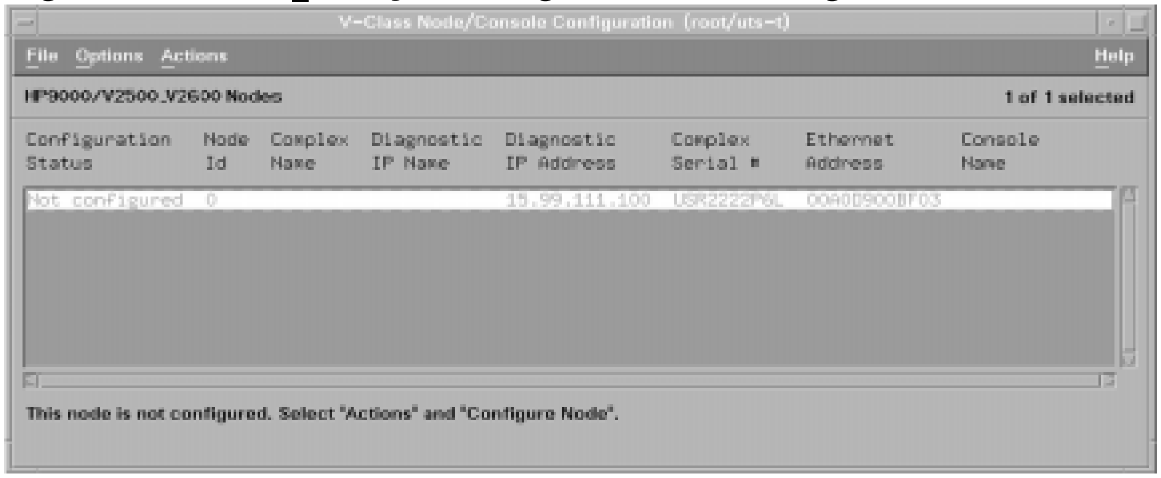
Power cycle the V-Class node to activate the new JTAG firmware.

When the node is powered up, the "Configuration Status" should change to "Not Configured."

Configure a Node

Step 1. Select the desired node from the list of available nodes. When the node is selected, the appropriate line is highlighted as shown in Figure 10. Notice the bottom of the display indicates the Node 0 is not configured and provides the steps necessary to configure the node.

Figure 10 **`ts_config`** indicating Node 0 as not configured



HP9000/V2500_V2600 Nodes

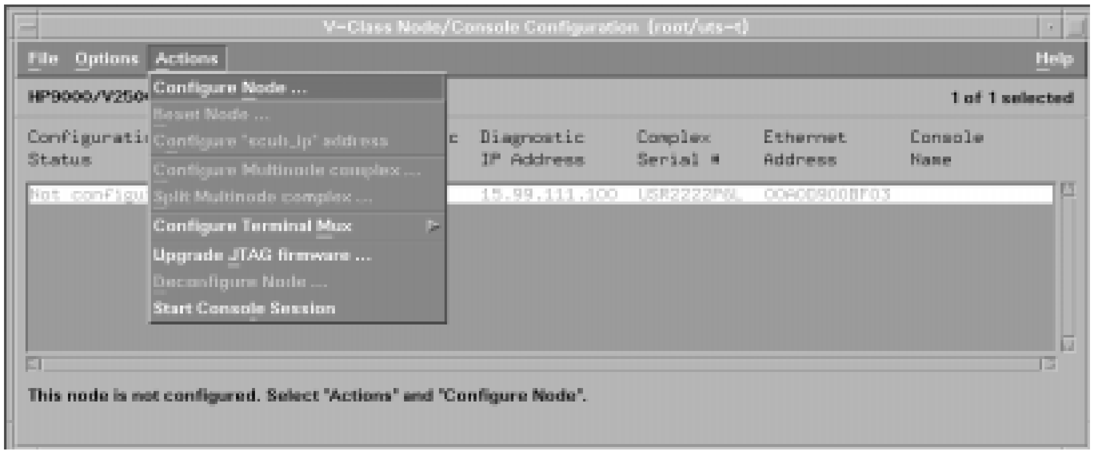
1 of 1 selected

Configuration Status	Node Id	Complex Name	Diagnostic IP Name	Diagnostic IP Address	Complex Serial #	Ethernet Address	Console Name
Not configured	0			15.99.111.100	USR2222P6L	00A0D900BF03	

This node is not configured. Select "Actions" and "Configure Node".

Step 2. Select "Actions" and then click "Configure Node," as shown in Figure 11.

Figure 11 **`ts_config`** "Configure Node" selection.


V-Class Node/Console Configuration (root/uts-t)
File
Options
Actions
Help
Configure Node ...
Reset Node ...
Configure "scub_ip" address
Configure Multinode complex ...
Split Multinode complex ...
Configure Terminal Mux
Upgrade JTAG firmware ...
Deconfigure Node ...
Start Console Session

HP9000/V250

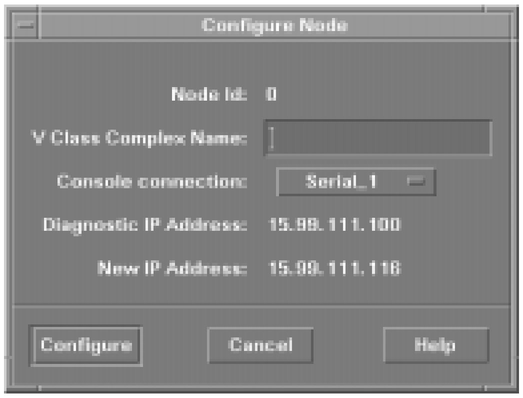
1 of 1 selected

Configurati Status		Diagnostic IP Address	Complex Serial #	Ethernet Address	Console Name
Not configu		15.99.111.100	USR2222P6L	00A0D900BF03	

This node is not configured. Select "Actions" and "Configure Node".

After invoking `ts_config` to configure the node, a node configuration panel appears as the one in Figure 12.

Figure 12 **`ts_config`** **node configuration panel**



Configure Node

Node Id: 0

V Class Complex Name:

Console connection: Serial_1

Diagnostic IP Address: 15.99.111.100

New IP Address: 15.99.111.116

Step 3. Enter a name for the V2500/V2600 System. The SSP uses this name as the “Complex Name” and to generate the IP host names of the Diagnostic and OBP LAN interfaces. Select a short name that SSP users can easily relate to the associated system (for example: **`hw2a`**, **`swtest`**, etc.).

Step 4. Select an appropriate serial connection for the V2500/V2600 console from the pop-down option menu in the node configuration panel.

`ts_config` automatically assigns the first unused serial port. If the terminal mux has been configured, the terminal mux ports are included in the list of available serial connections.

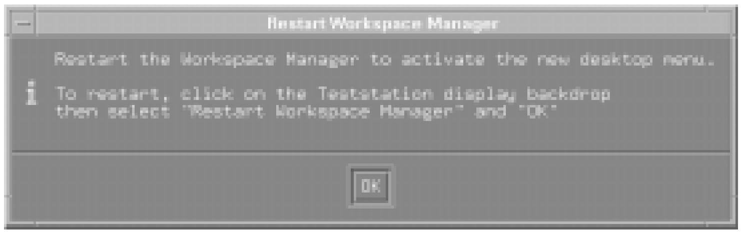
The IP address information for the Diagnostic interface is provided. The `ts_config` utility automatically changes the IP address of the diagnostic LAN interface to prevent a duplicate when other nodes are added to this SSP configuration.

`ts_config` automatically updates the local /etc/hosts file with the names and addresses of the Diagnostic and OBP LAN interfaces.

Step 5. Click “Configure.”

This updates several SSP files. The node configuration confirmation panel appears as the one in Figure 13.

Figure 13 **`ts_config`** **restart workspace manager panel.**



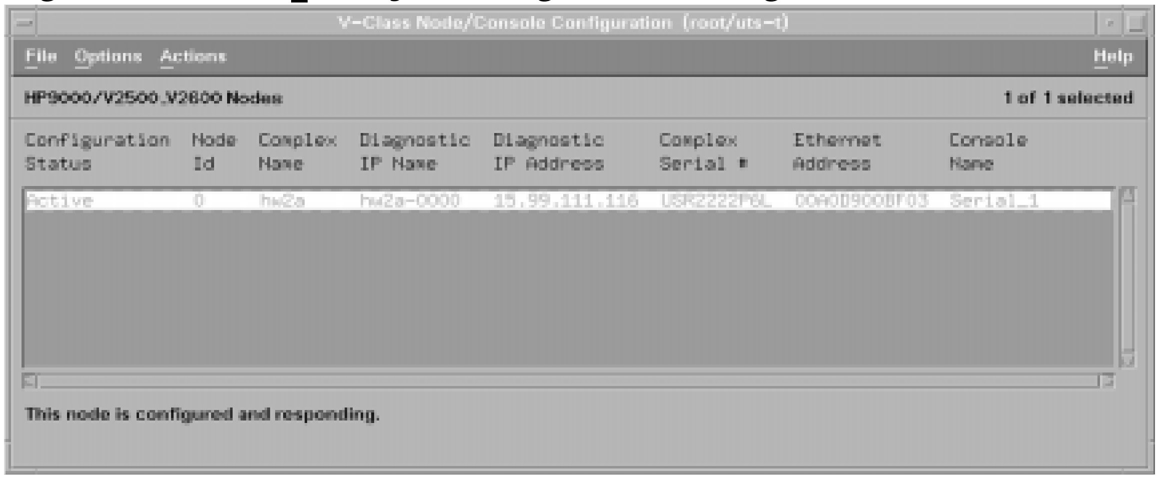
Restart Workspace Manager

Restart the Workspace Manager to activate the new desktop menu.

To restart, click on the Teststation display backdrop
then select "Restart Workspace Manager" and "OK"

Step 6. Read the panel and click "OK." When the configuration process is complete, the "Configuration Status" of the node changes to "Active," as shown in Figure 14.

Figure 14 **`ts_config`** **indicating Node 0 is configured**



V-Class Node/Console Configuration (root/uts-t)

HP9000/V2500,V2600 Nodes 1 of 1 selected

Configuration Status	Node Id	Complex Name	Diagnostic IP Name	Diagnostic IP Address	Complex Serial #	Ethernet Address	Console Name
Active	0	hw2a	hw2a-0000	15.99.111.116	USR2222P6L	00A0D900BF03	Serial_1

This node is configured and responding.

Step 7. Restart the Workspace Manager: Click the right-mouse button on the desktop background to activate the root menu. Select the "Restart" or "Restart Workspace Manager" option, then "OK" to activate the new desktop menu.

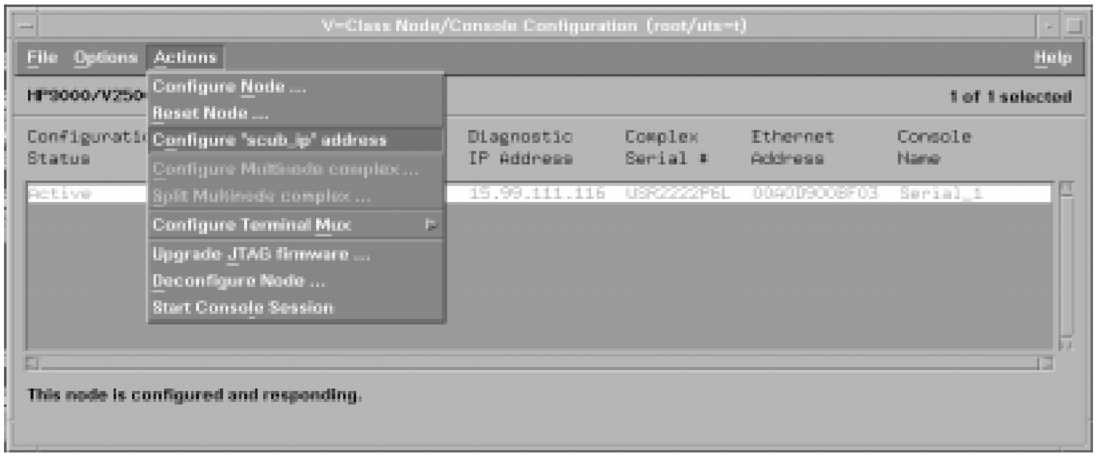
NOTE If adding multiple nodes to the SSP, wait until the final node is added before restarting the Workspace Manager.

Configure the scub_ip address

Step 1. Select the desired node from the list of available nodes.

Step 2. In the `ts_config` display panel, select “Actions” and then “Configure ‘scub_ip’ address,” as shown in Figure 15.

Figure 15 **`ts_config` “Configure ‘scub_ip’ address” selection**


V-Class Node/Console Configuration (root/uts-t)
File
Options
Actions
Help
Configure Node ...
Reset Node ...
Configure 'scub_ip' address
Configure Multinode complex ...
Split Multinode complex ...
Configure Terminal Mux
Upgrade JTAG firmware ...
Deconfigure Node ...
Start Console Session

HP9000/V250 — 1 of 1 selected

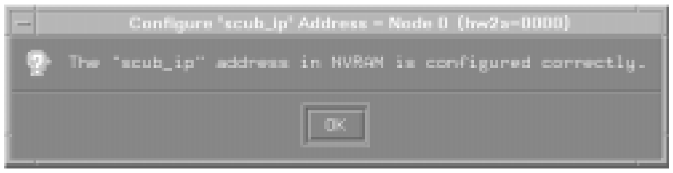
Configuration Status	Diagnostic IP Address	Complex Serial #	Ethernet Address	Console Name
Active	15.99.111.116	USR2222P6L	00A0D9008F03	Serial_1

This node is configured and responding.

`ts_config` checks the scub_ip address stored in NVRAM on the SCUB in the node. This would initially be the default address set at the factory.

If the scub_ip address is correct, the panel shown in Figure 16 is displayed and no action is required. If the node is not detected and scanned by `ccmd`, `ts_config` may ask you to try again later. The `ccmd` detection scan process should take less than a minute.

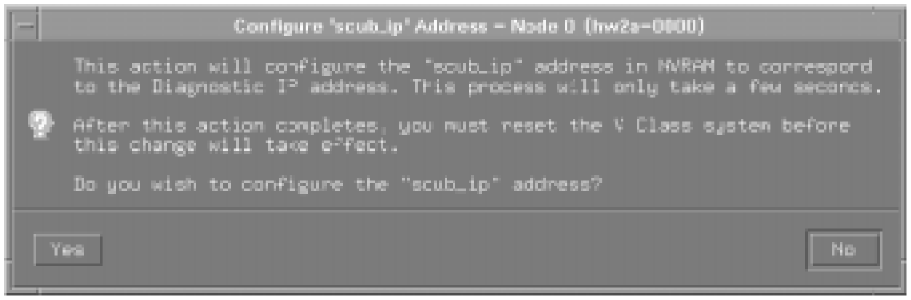
Figure 16 **`ts_config` “SCUB OK” panel**



The "scub_ip" address in NVRAM is configured correctly.

Step 3. If prompted by `ts_config` (as indicated by the panel in Figure 17), click “Yes” to correctly set the scub_ip address.

Figure 17 **`ts_config` scub_ip address configuration confirmation**



Configure 'scub_ip' Address – Node 0 (hw2a-0000)

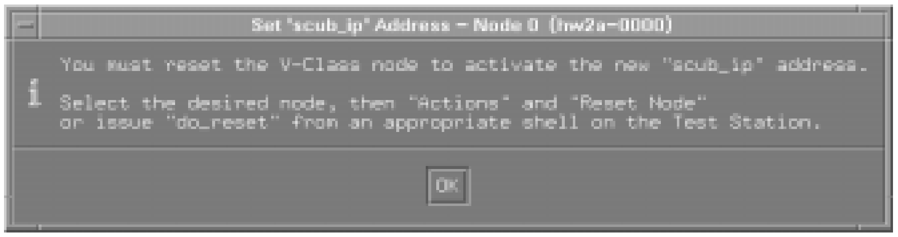
This action will configure the "scub_ip" address in NVRAM to correspond to the Diagnostic IP address. This process will only take a few seconds.

After this action completes, you must reset the V Class system before this change will take effect.

Do you wish to configure the "scub_ip" address?

Step 4. A panel as the shown in Figure 18 appears confirming that the scub_ip address is set. Click OK.

Figure 18 **`ts_config` scub_ip address set confirmation panel**



Set 'scub_ip' Address – Node 0 (hw2a-0000)

You must reset the V-Class node to activate the new "scub_ip" address.

Select the desired node, then "Actions" and "Reset Node" or issue "do_reset" from an appropriate shell on the Test Station.

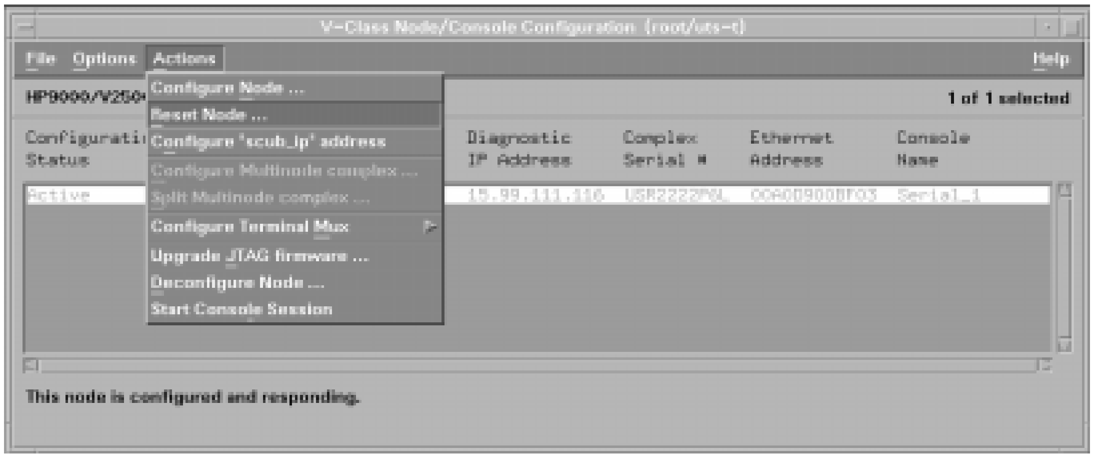
Initiate a node reset to activate the new scub_ip address.

Reset the Node

Step 1. Select the desired node from the list of available nodes.

Step 2. Select “Actions,” then “Reset Node.” This is indicated in Figure 19.

Figure 19 **`ts_config`** "Reset Node" selection


V-Class Node/Console Configuration (root/uts-t)
File
Options
Actions
Help
Configure Node ...
Reset Node ...
Configure 'scub_ip' address
Configure Multinode complex ...
Split Multinode complex ...
Configure Terminal Mux
Upgrade JTAG firmware ...
Deconfigure Node ...
Start Console Session

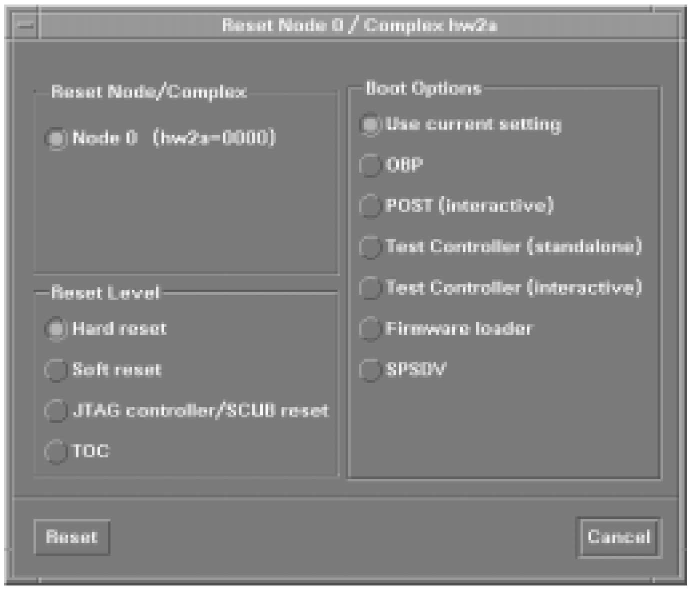
HP9000/V250 1 of 1 selected

```
Configurati  Diagnostic    Complex    Ethernet      Console
Status       IP Address    Serial #   Address       Name
Active       15.99.111.116 USR2222P8L 00A0D9000F03  Serial_1
```

This node is configured and responding.

A panel as the one shown in Figure 20 appears.

Figure 20 **`ts_config`** node reset panel



Reset Node/Complex
- [✓ Node 0 (hw2a-0000)]

Reset Level
- [✓ Hard reset]
- Soft reset
- JTAG controller/SCUB reset
- TOC

Boot Options
- [✓ Use current setting]
- OBP
- POST (interactive)
- Test Controller (standalone)
- Test Controller (interactive)
- Firmware loader
- SPSDV

Step 3. In the Node Reset panel, select the desired "Reset Level" and "Boot Options," then click Reset."

Deconfigure a Node

Deconfiguring a node removes the selected node from the SSP configuration. The SSP will no longer monitor the environmental and hard-error status of this node. Console access to the node is also be disabled.

Step 1. Select the desired node from the list of available nodes.

Step 2. Select “Actions,” then “Deconfigure Node,” then click “Yes.”

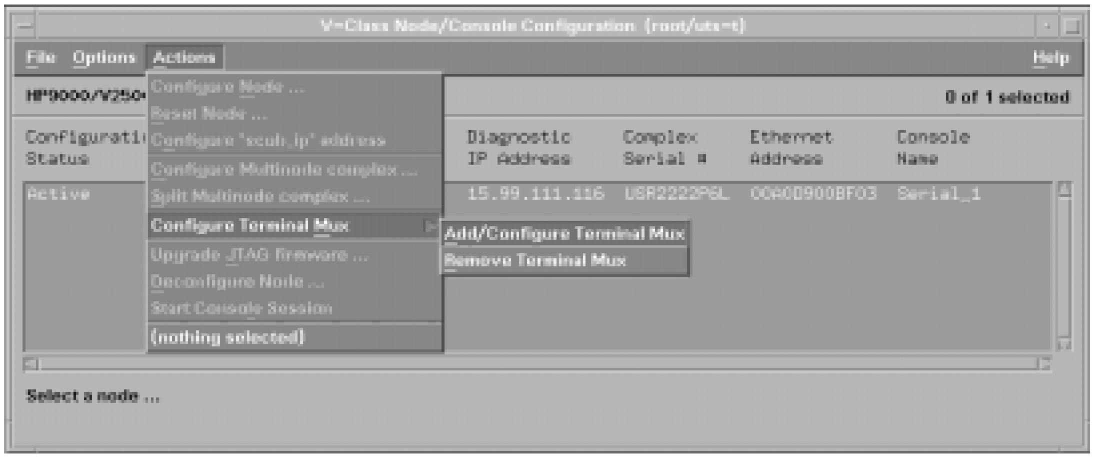
Add/Configure the Terminal Mux

To add or reconfigure the terminal mux, perform the following procedure.

Step 1. In the `ts_config` display, select “Actions,” then “Configure Terminal Mux.”

Select “Add/Configure Terminal Mux.” This is indicated in Figure 21.

Figure 21 **`ts_config` “Add/Configure Terminal Mux” selection**


V-Class Node/Console Configuration (root/uts-t)
File
Options
Actions
Help
HP9000/V250
0 of 1 selected
Configure Node ...
Reset Node ...
Configure 'scub_ip' address
Configure Multinode complex ...
Split Multinode complex ...
Configure Terminal Mux
Add/Configure Terminal Mux
Remove Terminal Mux
Upgrade JTAG firmware ...
Deconfigure Node ...
Start Console Session
(nothing selected)

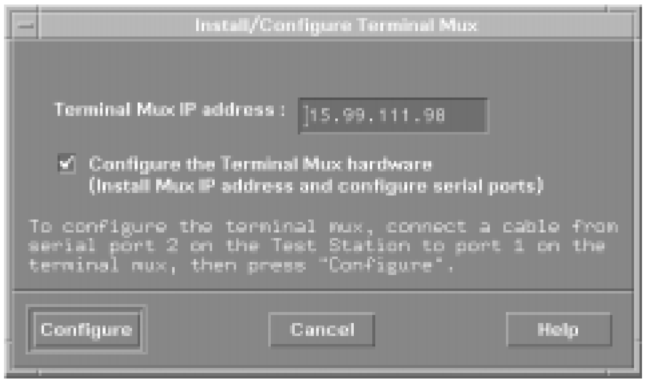
```
Configurati   Diagnostic     Complex      Ethernet      Console
Status        IP Address     Serial #     Address       Name
Active        15.99.111.116  USR2222P6L   00A0D900BF03  Serial_1
```

Select a node ...

Step 2. Connect a serial cable from serial port 2 on the SSP to port 1 on the terminal mux.

Step 3. A panel shown in Figure 22 display the IP address.

Figure 22 **Terminal mux IP address panel**



Terminal Mux IP address : 15.99.111.98

✓ Configure the Terminal Mux hardware
(Install Mux IP address and configure serial ports)

```
To configure the terminal mux, connect a cable from
serial port 2 on the Test Station to port 1 on the
terminal mux, then press "Configure".
```

Remove terminal mux

`ts_config` does not remove the terminal mux if any node consoles are assigned to terminal mux ports.

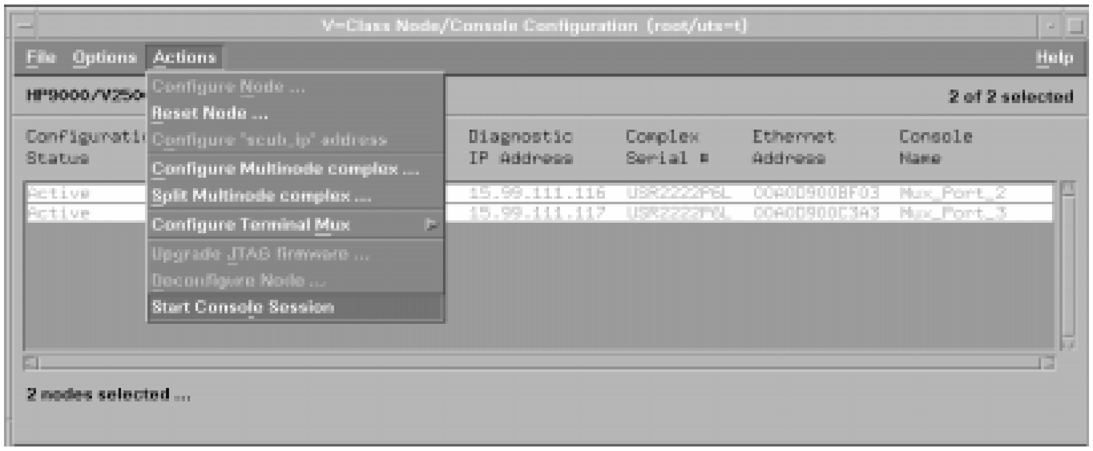
Step 1. Select “Actions,” then “Configure Terminal Mux.”

Step 2. Select “Remove Terminal Mux,” then click “Yes.”

Console sessions

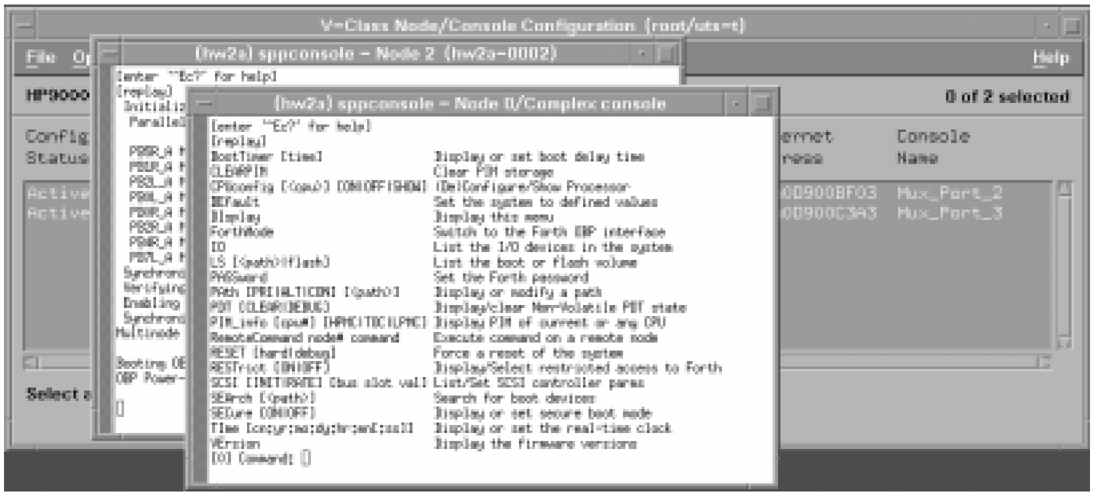
`ts_config` may also start console sessions by selecting the desired node(s) and then selecting the “Start Console Session” action as shown in Figure 23. Figure 24 shows the started console sessions.

Figure 23 “Start Console Session” selection


V-Class Node/Console Configuration (root/uts-t)
File
Options
Actions
Help
Configure Node ...
Reset Node ...
Configure 'scub_ip' address
Configure Multinode complex ...
Split Multinode complex ...
Configure Terminal Mux
Upgrade JTAG firmware ...
Deconfigure Node ...
Start Console Session

```
HP9000/V250                                                    2 of 2 selected

Configurati          Diagnostic      Complex       Ethernet       Console
Status               IP Address      Serial #      Address        Name

Active               15.99.111.116   USR2222P6L    00A0D900BF03   Mux_Port_2
Active               15.99.111.117   USR2222P6L    00A0D900C3A3   Mux_Port_3

2 nodes selected ...
```

Figure 24 Started console sessions


V-Class Node/Console Configuration (root/uts-t)
(hw2a) sppconsole - Node 2 (hw2a-0002)
(hw2a) sppconsole - Node 0/Complex console
Help
0 of 2 selected

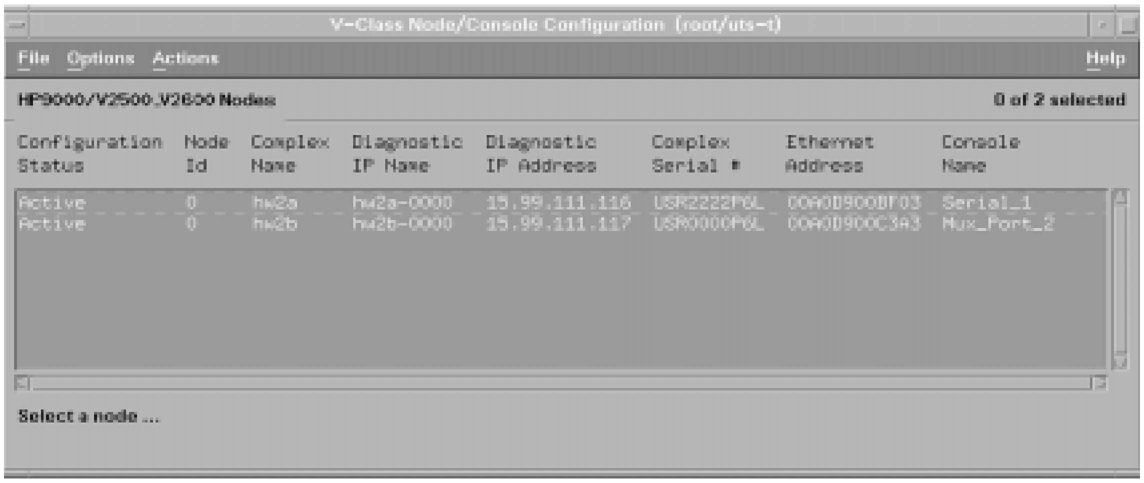
```
[enter "Ec?" for help]
[replay]
BootTimer [time]                  Display or set boot delay time
CLEARPIM                          Clear PIM storage
CPUconfig [<cpu>] [ON|OFF|SHOW]   (De)Configure/Show Processor
DEFault                           Set the system to defined values
Display                           Display this menu
ForthMode                         Switch to the Forth OBP interface
IO                                List the I/O devices in the system
LS [<path>|flash]                 List the boot or flash volume
PASSword                          Set the Forth password
PAth [PRI|ALT|CON] [<path>]       Display or modify a path
PDT [CLEAR|DEBUG]                 Display/clear Non-Volatile PDT state
PIM_info [cpu#] [HPMC|TOC|LPMC]   Display PIM of current or any CPU
RemoteCommand node# command       Execute command on a remote node
RESET [hard|debug]                Force a reset of the system
RESTrict [ON|OFF]                 Display/Select restricted access to Forth
SCSI [INIT|RATE] [bus slot val]   List/Set SCSI controller parms
SEArch [<path>]                   Search for boot devices
SECure [ON|OFF]                   Display or set secure boot mode
TIme [cn:yr:mo:dy:hr:mn[:ss]]     Display or set the real-time clock
VErsion                           Display the firmware versions
[0] Command: []
```

V2500/V2600 SCA (multinode) configuration

`ts_config` can also configure a V2500/V2600 SCA system. An example to follow describes how. The example assumes that there are two active single-node complexes. After the system has rebooted to OBP, node 0 becomes the console for the SCA complex.

To configure the two-node system in the example, start `ts_config` as described in “Starting ts_config” on page 21. Once `ts_config` has started, a window like that shown in Figure 25 is displayed.

Figure 25 **SSP supporting two single-node complexes**



HP9000/V2500_V2600 Nodes

0 of 2 selected

Configuration Status	Node Id	Complex Name	Diagnostic IP Name	Diagnostic IP Address	Complex Serial #	Ethernet Address	Console Name
Active	0	hw2a	hw2a-0000	15.99.111.116	USR2222P6L	00A0D900BF03	Serial_1
Active	0	hw2b	hw2b-0000	15.99.111.117	USR0000P6L	00A0D900C3A3	Mux_Port_2

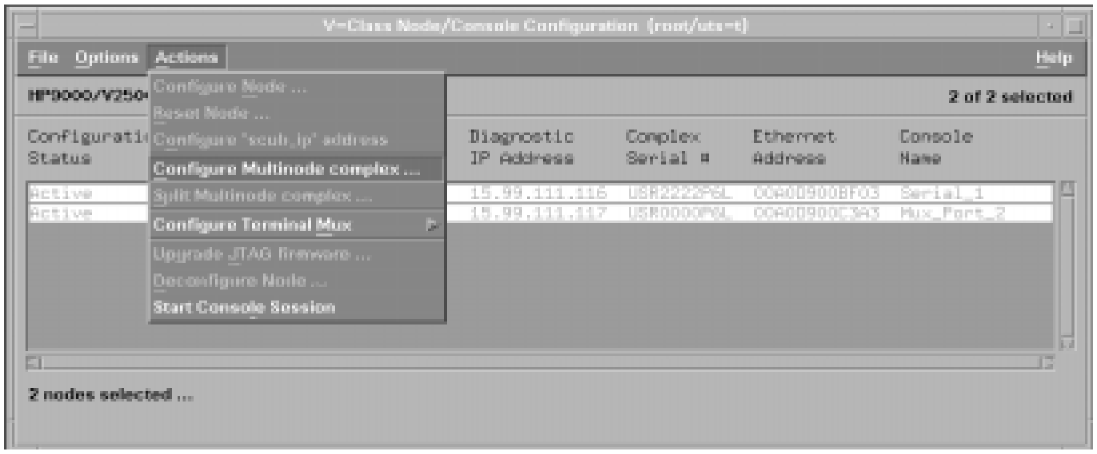
Select a node ...

The following procedure configures the two-node SCA system in the example.

Step 1. Select the nodes by clicking anywhere in the information display for each node. As the nodes are selected, they become highlighted.

Step 2. Select “Action” and then “Configure Multinode complex” as shown in Figure 26.

Figure 26 **`ts_config` Configure Multinode complex selection**


V-Class Node/Console Configuration (root/uts-t)
File Options Actions Help
Configure Node ...
Reset Node ...
Configure 'scub_ip' address
Configure Multinode complex ...
Split Multinode complex ...
Configure Terminal Mux
Upgrade JTAG firmware ...
Deconfigure Node ...
Start Console Session

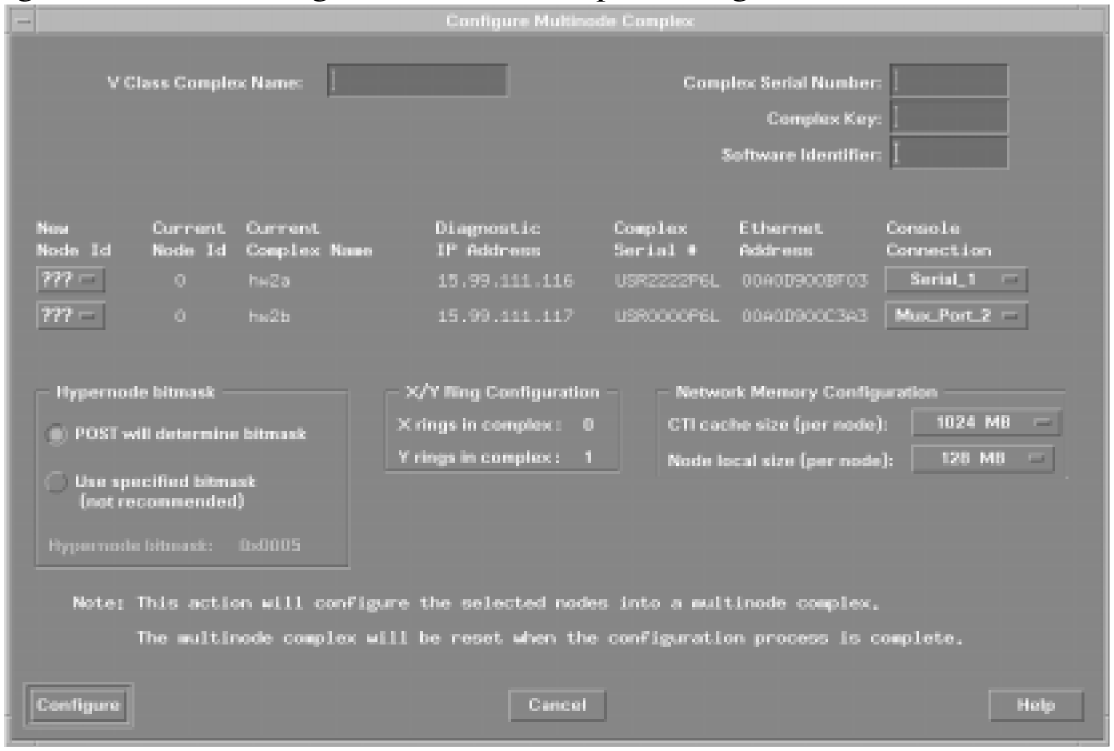
HP9000/V250 — 2 of 2 selected

Configuration Status	Diagnostic IP Address	Complex Serial #	Ethernet Address	Console Name
Active	15.99.111.116	USR2222P6L	00A0D900BF03	Serial_1
Active	15.99.111.117	USR0000P6L	00A0D900C3A3	Mux_Port_2

2 nodes selected ...

Step 3. When "Configure Multinode complex" is selected, a configuration dialog appears as shown in Figure 37.

Figure 27 **Configure Multinode Complex dialog window**



V Class Complex Name:
Complex Serial Number:
Complex Key:
Software Identifier:

New Node Id	Current Node Id	Current Complex Name	Diagnostic IP Address	Complex Serial #	Ethernet Address	Console Connection
???	0	hw2a	15.99.111.116	USR2222P6L	00A0D900BF03	Serial_1
???	0	hw2b	15.99.111.117	USR0000P6L	00A0D900C3A3	Mux_Port_2

Hypernode bitmask
[✓ POST will determine bitmask]
Use specified bitmask (not recommended)
Hypernode bitmask: 0x0005

X/Y Ring Configuration
X rings in complex : 0
Y rings in complex : 1

Network Memory Configuration
CTI cache size (per node): 1024 MB
Node local size (per node): 128 MB

Note: This action will configure the selected nodes into a multinode complex.
The multinode complex will be reset when the configuration process is complete.

Step 4. Enter the required fields into the Configure Multinode Complex dialog window.

- V-Class Complex Name—Current complex name of either node or a new complex name.
- Complex Serial Number—Unique serial number of the complex. This is not required if the nodes have the same serial numbers.
- Complex Key—Number required to enter the Complex Serial Number.

NOTE If all of the SCA system complex serial numbers are the same, no complex key is required.

Step 5. Select the desired node IDs from the “New Node ID” drop-down lists.

Step 6. If the console connection must be changed, select appropriate connection from the “Console Connection” drop-down list.

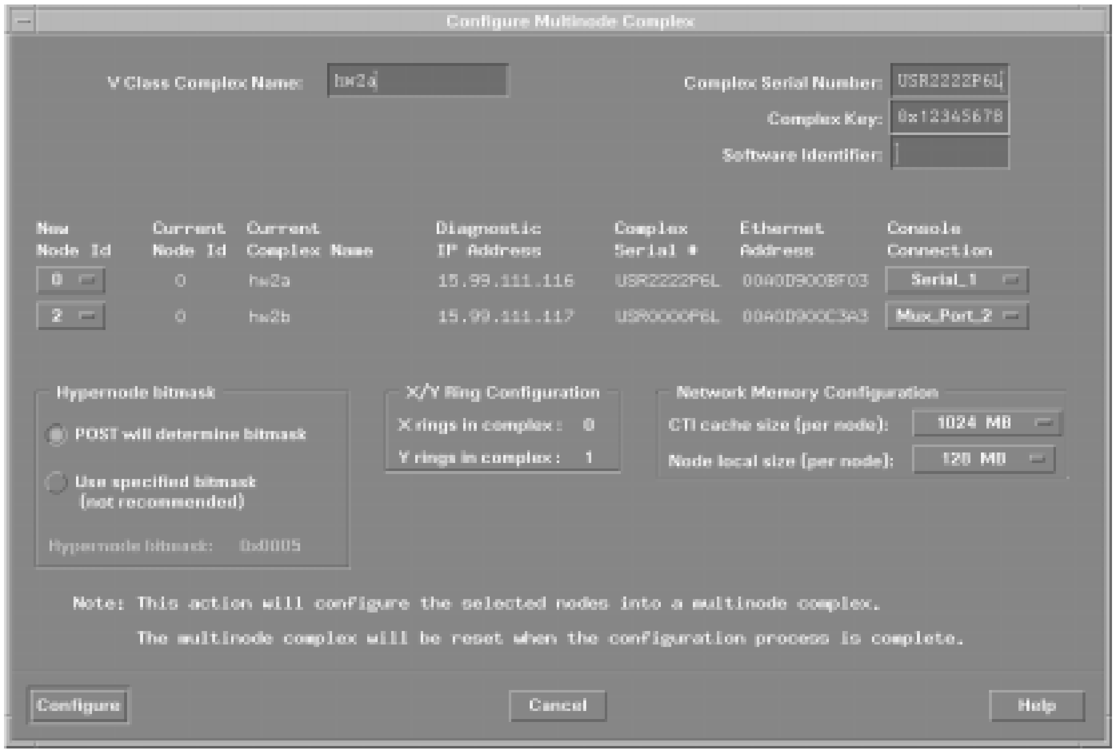
Step 7. In the “Hypernode bitmask” section select “POST will determine bitmask.”

Step 8. If necessary, select the desired CTI cache size from the “CTI cache size” drop-down list.

Step 9. If necessary, select the node-local memory size from the “Node local size” pull-down list.

NOTE The default settings for CTI cache and node-local memory are recommended.

Figure 28 **Configure Multinode Complex dialog window with appropriate values**



V Class Complex Name: hw2a

Complex Serial Number: USR2222P6L
Complex Key: 0x12345678
Software Identifier:

New Node Id	Current Node Id	Current Complex Name	Diagnostic IP Address	Complex Serial #	Ethernet Address	Console Connection
0	0	hw2a	15.99.111.116	USR2222P6L	00A0D900BF03	Serial_1
2	0	hw2b	15.99.111.117	USR0000P6L	00A0D900C3A3	Mux_Port_2

Hypernode bitmask
[✓ POST will determine bitmask]
Use specified bitmask (not recommended)
Hypernode bitmask: 0x0005

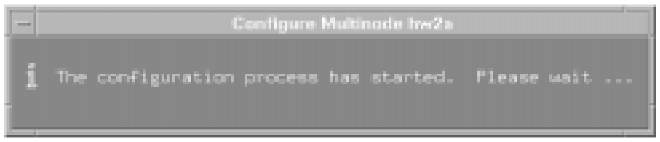
X/Y Ring Configuration
X rings in complex : 0
Y rings in complex : 1

Network Memory Configuration
CTI cache size (per node): 1024 MB
Node local size (per node): 128 MB

```
Note: This action will configure the selected nodes into a multinode complex.
      The multinode complex will be reset when the configuration process is complete.
```

Step 10. Click the “Configure” button to start the configuration. A message box appears indicating that the configuration has started.

Figure 29 **Configuration started information box**



```
The configuration process has started.  Please wait ...
```

The following activities occur during the configuration process:

- SSP files are updated based on the new complex and node names.
- Essential console server processes are started, and the now-obsolete server processes are halted.
- New node information is written to the COP chip in each node.

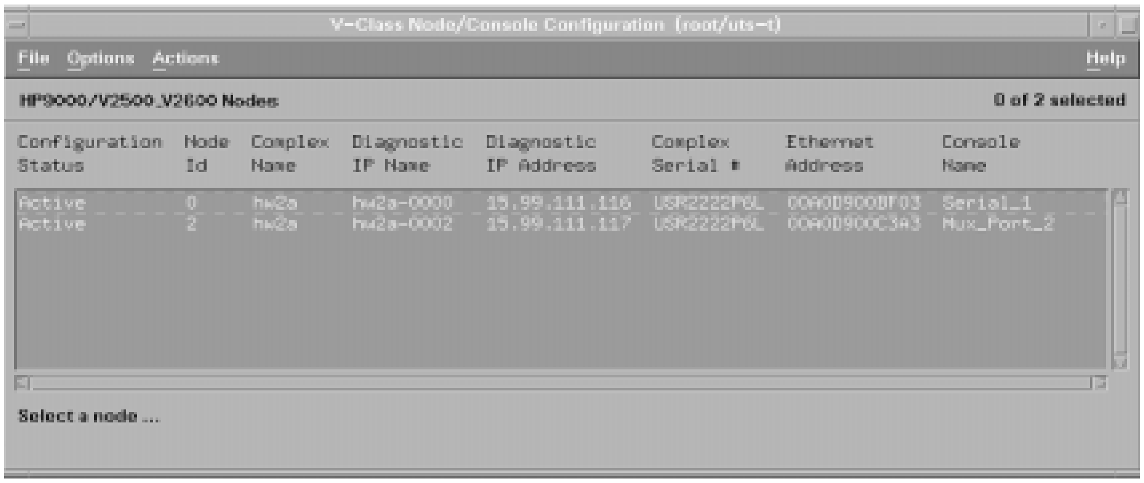
This information includes:

  - Node ID
  - Complex serial number (if it has been modified)
  - Requested or auto-generated software identifier

- Configuration Manager Daemon, `ccmd`, is notified of the new configuration.
- The shared-memory database of node information is updated.
- Multinode configuration parameters are written to NVRAM in each node. These include:
  - Hypernode bitmask
  - X- and Y-ring information
  - Node count
  - CTI cache size
  - Node-local memory size
- The boot vector of each node is set to OBP and each node is reset.

When the configuration process is complete, `ts_config` shows the new multinode complex, as in Figure 30. The restart process activates the new SSP root menu which includes customized menus for each complex. The Workspace Manager must be restarted or else the root menu will be outdated (the rest of the configuration process is complete).

Figure 30 **`ts_config`** showing newly configured complexes



HP9000/V2500_V2600 Nodes

0 of 2 selected

Configuration Status	Node Id	Complex Name	Diagnostic IP Name	Diagnostic IP Address	Complex Serial #	Ethernet Address	Console Name
Active	0	hw2a	hw2a-0000	15.99.111.116	USR2222P6L	00A0D900BF03	Serial_1
Active	2	hw2a	hw2a-0002	15.99.111.117	USR2222P6L	00A0D900C3A3	Mux_Port_2

Select a node ...

When remotely running `ts_config`, the Restart Workspace Manager step cannot be performed, because it is the SSP Workspace Manager that needs to be restarted. The Workspace Manager can be restarted at any time by clicking on the desktop background and selecting Restart Workspace Manager, then OK.

Any of the configurable parameters on the Multinode Configuration dialog may be changed by selecting each node and choosing the "Configure Multinode complex" action. Set the desired options and click Configure. During a reconfiguration, several of the required fields in the Multinode Configuration dialog are filled in by `ts_config`.

V2500/V2600 split SCA configuration

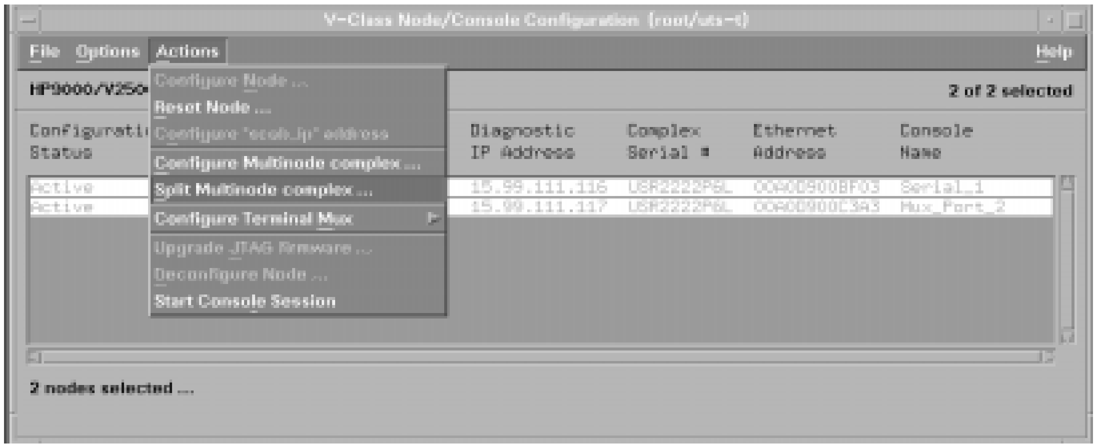
`ts_config` also provides a "Split Multinode complex" action that takes an SCA complex and logically splits it into single node systems. Each node becomes Node 0 in a new complex.

The following procedure allows the user to split the SCA system:

Step 1. In the ts_config window, select the desired nodes.

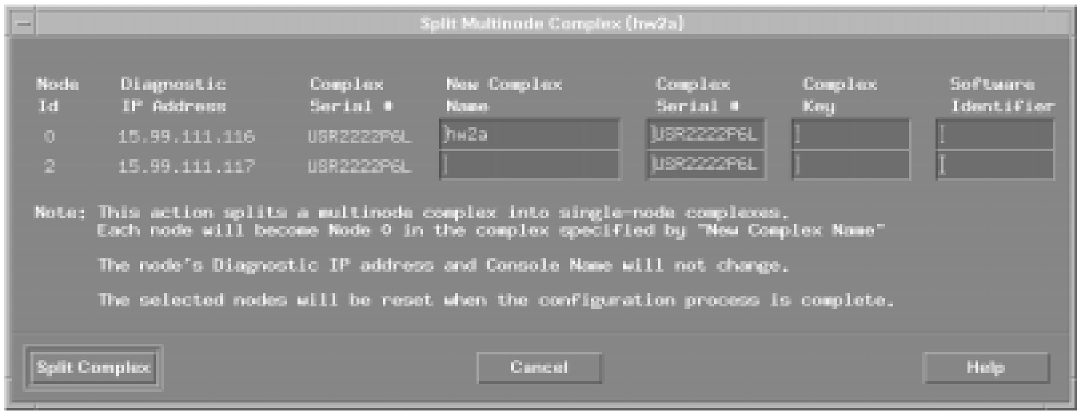
Step 2. Select every node in the desired complex, then "Actions," and then "Split Multinode complex." Figure 32 shows the `ts_config` Split Multinode complex panel.

Figure 31 **`ts_config`** Split Multinode complex operation


V-Class Node/Console Configuration (root/uts-t)
File
Options
Actions
Help
Configure Node ...
Reset Node ...
Configure "ecub_ip" address
Configure Multinode complex ...
Split Multinode complex ...
Configure Terminal Mux
Upgrade JTAG firmware ...
Deconfigure Node ...
Start Console Session

```
HP9000/V250                                            2 of 2 selected

Configurati    Diagnostic      Complex       Ethernet        Console
Status         IP Address      Serial #      Address         Name
Active         15.99.111.116   USR2222P6L    00A0D900BF03    Serial_1
Active         15.99.111.117   USR2222P6L    00A0D900C3A3    Mux_Port_2

2 nodes selected ...
```

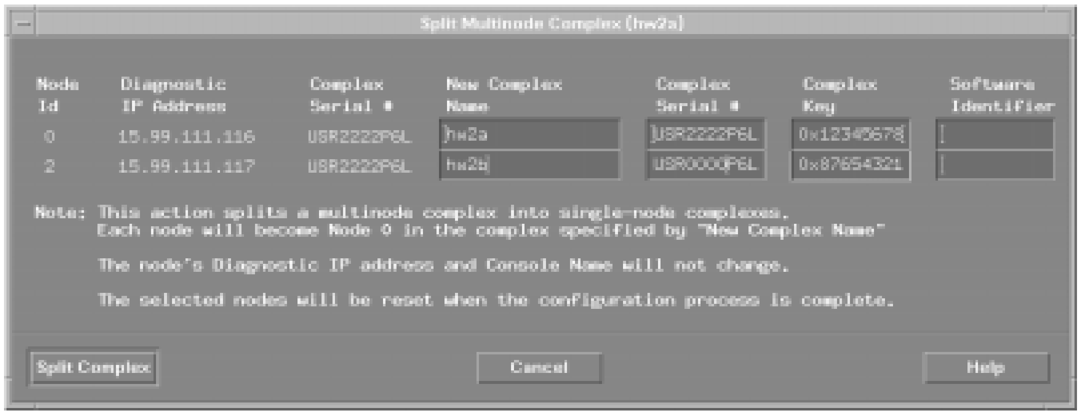
Figure 32 **`ts_config`** Split Multinode complex panel



```
Node  Diagnostic      Complex      New Complex   Complex      Complex  Software
Id    IP Address      Serial #     Name          Serial #     Key      Identifier
 0    15.99.111.116   USR2222P6L   hw2a          USR2222P6L
 2    15.99.111.117   USR2222P6L                 USR2222P6L

Note: This action splits a multinode complex into single-node complexes.
      Each node will become Node 0 in the complex specified by "New Complex Name"

      The node's Diagnostic IP address and Console Name will not change.

      The selected nodes will be reset when the configuration process is complete.
```

Step 3. Enter the complex names for each node. New complex serial numbers may be assigned. Each node becomes node 0 in a new complex. Figure 33 shows the Split Multinode panel filled in. Click the Split Complex button to initiate the configuration process.

Figure 33 **`ts_config` Split Multinode complex panel filled in**

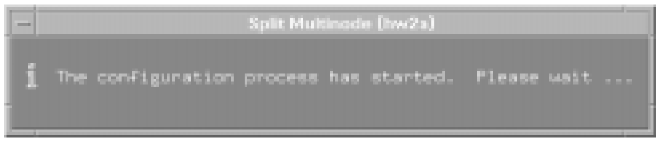


Node Id	Diagnostic IP Address	Complex Serial #	New Complex Name	Complex Serial #	Complex Key	Software Identifier
0	15.99.111.116	USR2222P6L	hw2a	USR2222P6L	0x12345678	
2	15.99.111.117	USR2222P6L	hw2b	USR0000P6L	0x87654321	

```
Note: This action splits a multinode complex into single-node complexes.
      Each node will become Node 0 in the complex specified by "New Complex Name"

      The node's Diagnostic IP address and Console Name will not change.

      The selected nodes will be reset when the configuration process is complete.
```

The message shown in Figure 34 appears indicating the configuration is taking place.

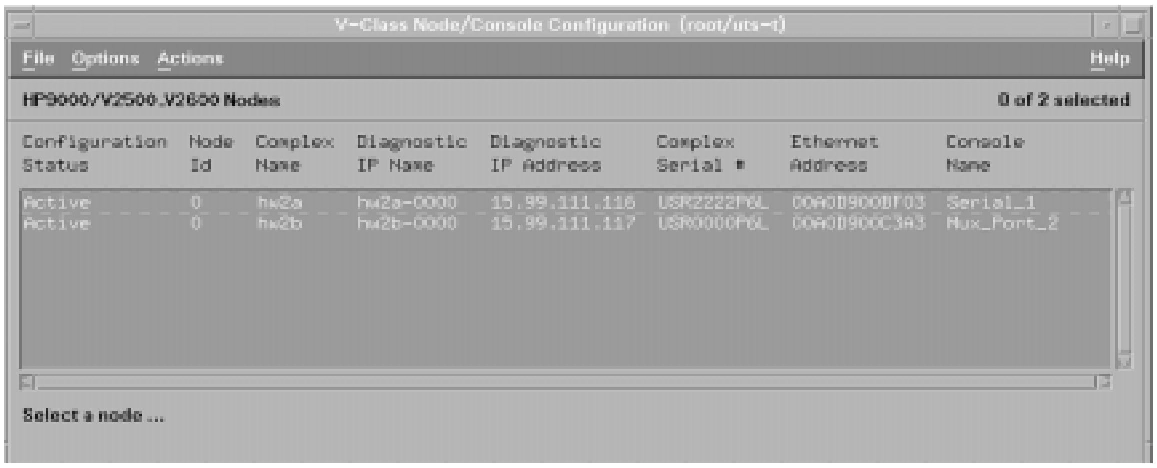
Figure 34 **Split Multinode confirmation panel**



```
The configuration process has started.  Please wait ...
```

Figure 35 shows the main `ts_config` display after the split multinode operation has completed. It shows the resulting configuration: two single node complexes (two node 0s) with names assigned in the prior step.

Figure 35 **`ts_config` Split Multinode operation complete**



HP9000/V2500,V2600 Nodes — 0 of 2 selected

Configuration Status	Node Id	Complex Name	Diagnostic IP Name	Diagnostic IP Address	Complex Serial #	Ethernet Address	Console Name
Active	0	hw2a	hw2a-0000	15.99.111.116	USR2222P6L	00A0D900BF03	Serial_1
Active	0	hw2b	hw2b-0000	15.99.111.117	USR0000P6L	00A0D900C3A3	Mux_Port_2

Select a node ...

ts_config files

`ts_config` either reads or maintains the following SSP configuration files:

`/etc/hosts`	The standard system hosts file, includes entries for the cabinet related IP addresses.
`/etc/services`	Service definitions for the console interface.
`/etc/inetd.conf`	Contains entries for starting console related processes
`/spp/data/nodes.conf`	Contains entries which define the complexes (either single cabinet or multi-cabinet) managed by the SSP. This file is maintained by the Configure Node Action of `ts_config`, but other commands can also update this file: `delete_node`, `configure_node`, and `split_multinode`.
`/spp/data/conserver.cf`	Connection definitions for the console interface.
`/spp/data/consoles.conf`	Console name to ttylink number resolution. This file is maintained by the Configure Mux Action of `ts_config`.
`/spp/data/`*<complex_name>*	For each newly configured complex, there is a complex-specific directory that contains complex-specific files, such as event and console logs. `ts_config` generates each *<complex_name>* during configuration.

The nodes.conf file contains most of the configuration management information. It defines the relationship between complex names, cabinets (nodes) within that complex and the associated host names and console port connections.

Each node has an entry in the nodes.conf file as follows:

```
NODE Complex Node ID JTAG-hostname OBP-hostname SSP-hostname Console-port
```

The variables of the entry are defined as follows:

NODE—Keyword designating a cabinet (node) entry.

Complex—Name to which the node (cabinet) is associated.

In a multi-cabinet complex all the cabinets comprise a single system (complex) and are managed by a single console (the console on cabinet 0). Each cabinet, however, has its own console that can report diagnostic information, and there is still a console configuration entry for each cabinet. These consoles, however, are not normally accessed.

Node ID—The identification of the V-Class node. This number can be 0, 2, 4, or 6.

Jtag-hostname—Host name used by the SSP to communicate with the JTAG firmware on the associated node. JTAG IP addresses are 15.99.111.116 through 15.99.111.131.

Obp-hostname—Host name used for OBP communication (not normally referenced while administering the complex). OBP IP addresses are 15.99.111.166 through 15.99.111.181.

SSP-hostname—Local host name of the private/diagnostic LAN. The default host name for this interface is tsdart-d.

Console-port—Name of the physical connection to the node RS-232 console port. The port name is linked to a ttylink service entry via the file /spp/data/consoles.conf.

The `get_node_info` program extracts information from the nodes.conf file.

IMPORTANT Most SSP configuration is carried out by `ts_config`. Sometimes problems are caused by manually editing the files maintained by `ts_config`, resulting in it not be able to properly parse the files on its own.

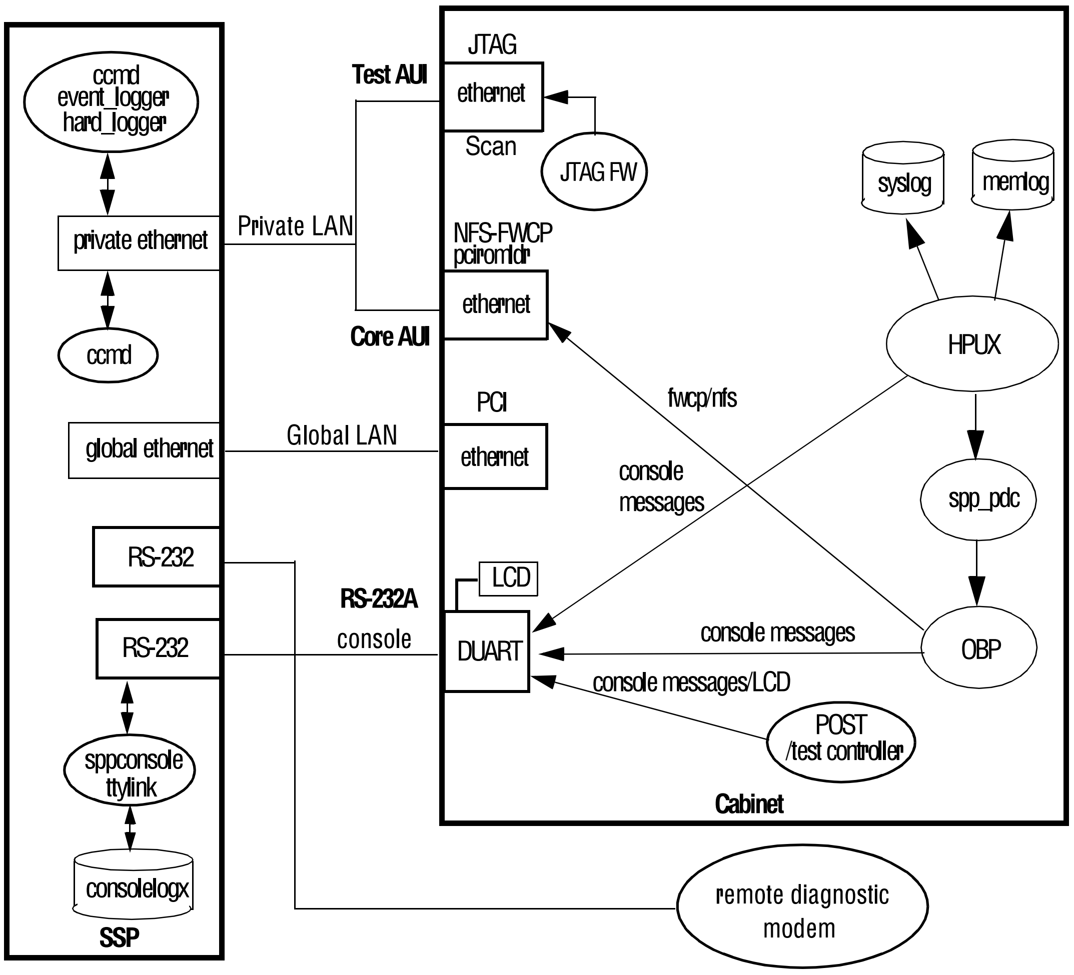
The `ts.install` script is designed not to run after initial installation to prevent inadvertent removal of changes made to the configuration files by `ts_config`. It can be forced to run, however, in order to put all the configuration files back to factory defaults. To rerun `ts.install`, execute the following command:

```
/spp/scripts/inst/ts.install
```

SSP-to-system communications

Figure 36 depicts the V-Class server to SSP communications using HP-UX.

Figure 36 SSP-to-system communications



A layer of firmware between HP-UX and OBP (Open Boot PROM) called `spp_pdc` allows the HP-UX kernel to communicate with OBP. `spp_pdc` is platform-dependent code and runs on top of OBP providing access to the devices and OBP configuration properties.

LAN communications

There are two ethernet ports located on the SCUB as shown in the diagram in the upper-left side of the node (dotted line) in Figure 36 on page 46. These comprise the "private" or diagnostic LAN. The JTAG port is used for scanning, and the NFS-FWCP port is used for downloading system firmware via `nfs` using the `fwcp` utility, via `tftp` using the `pdcfl` utility, downloading disk firmware using the `dfdutil` utility (`dfdutil` uses `tftp` for reading peripheral firmware), and loading Symbios FORTH code using the `pciromldr` utility. For more information on `dfdutil`, `tftp`, and `pciromldr`, see the appropriate man pages.

The configuration daemon, `ccmd`, which is located on the SSP obtains system configuration information over the private LAN from the JTAG port. It builds a configuration information database on the SSP. The board names and revisions, the device names and revisions, and the start-up information generated by POST are all read and stored in memory for use by other diagnostic tools.

IMPORTANT Both the B180L and the 712 workstations must have two ethernet connections: one for the private LAN and one for the global LAN. These ports are different on each model of workstation. It is important that the installer connect the LAN cables to the correct connector on the SSP.

The SSP can be placed on the customers Local Area Network using the SSP's global ethernet.

SSP host name and IP addresses

The SSP software installation process assigns the correct IP address to the appropriate LAN device.

The SCUB IP address 15.99.111.116 is the first SCUB IP address available on the SSP. If more nodes are added, they are assigned 15.99.111.117, 15.99.111.118, and so on. `ts_config` keeps track of which addresses are assigned.

The SCUB IP host name is "*complexname-000n*," where *n* is the node ID. The SSP supports multiple complexes (that is, multiple node 0s).

Serial communications

The DUART port on the SCUB provides an RS232 serial link to the SSP. Through this port HP-UX, OBP, POST (Power-On Self Test) and the Test Controller send console messages. The SSP processes these messages using the `sppconsole` and `ttylink` utilities and the `consolelogx` log file. POST and OBP also send system status to the LCD connected to the DUART. For more information on `sppconsole`, `ttylink`, and `consolelogx`, see the appropriate man pages.

NOTE The second RS-232 port on the workstations are unused and not enabled at this time.

ccmd

`ccmd` (Complex Configuration Management Daemon) is a daemon that maintains a database of information about the V2500/V2600 hardware. `ccmd` also monitors the system and reports any significant changes in system status. It supports multiple nodes, multiple complexes and nodes that have the same node number.

There are two types of related information in the database: node information (node numbers, IP addresses and scan data) and configuration data which is initialized by POST. The node information is scanned from the hardware and processed with the aid of data files at `/spp/data`. The POST configuration data is required so that certain scan based utilities can emulate various hardware functions.

`ccmd` periodically sends out a broadcast to determine what nodes are available. If `ccmd` can not talk to a node that it previously reached, it sends a response to the console and the log. If it establishes or re-establishes contact, or if a node powers up, `ccmd` reads hardware information from the node and interrogates it through scan to determine the node configuration. From this data, a complete database is built on the SSP that will be used for all scan based diagnostics.

Once running, `ccmd` checks for power-up, power-down, reset, error, and environmental conditions on regular intervals. If at any time `ccmd` detects a change in the configuration, it changes the database, updates the `/spp/data/complex.cfg` file, sets up a directory in `/spp/data` for each complex and initializes a `node_#.pwr` file for each node in the complex specific subdirectory.

If `ccmd` detects an error condition, it invokes an error analysis tool (`hard_logger`) that logs and diagnoses error conditions. After an error is investigated, `ccmd` reboots the node or complex associated with the error. The reboot operation can be avoided with the use of the `autoreset [complex]` *on*|*off*|*chk* utility in `/spp/scripts`.

`ccmd` is listed in /etc/inittab as a process that should run continuously. It may be started manually, but since it kills any previous copies at start-up, diagnostic processes that may be running will be orphaned. Only one copy of `ccmd` may run on a SSP.

If started with no options, `ccmd` disassociates itself from the terminal or window where it was started. It instead reports to the console window and the file /spp/data/ccmd_log.

If `ccmd` is sent a `SIGHUP`, it regenerates the database.

All scan-based operations require `ccmd`. If POST is unable to run, then `ccmd` is not able to read configuration data and some system information is not accessible.

`ccmd` works in co-operation with most utilities to share a common ethernet port and Diagnostic (DART) bus. In general, the scan data is sent via UDP. The DART bus should be separate from any general purpose ethernet bus. If the DART bus is improperly set-up, `ccmd` cannot run properly.

Since `ccmd` can become corrupted by bad data, it may be necessary to kill the `ccmd` process to refresh the SSPs configuration image. Killing the `ccmd` process is not always enough. If the “heart beat” LED from the SCUB is not functioning then `ccmd` is unable to communicate with the system. A `ping` command to the SCUB will not be successful either. In this case, the system or node must be powered down to reset the SCUB and re-establish communication with the SSP.

xconfig

`xconfig` is the graphical tool that can also modify the parameters initialized by POST to reconfigure a node.

The graphical interface allows the user to see the configuration state. Also the names are consistent with the hardware names, since individual configuration parameters are hidden to the user. The drawback of `xconfig` is that it can not be used as a part of script-based tests, nor can it be used for remote debug.

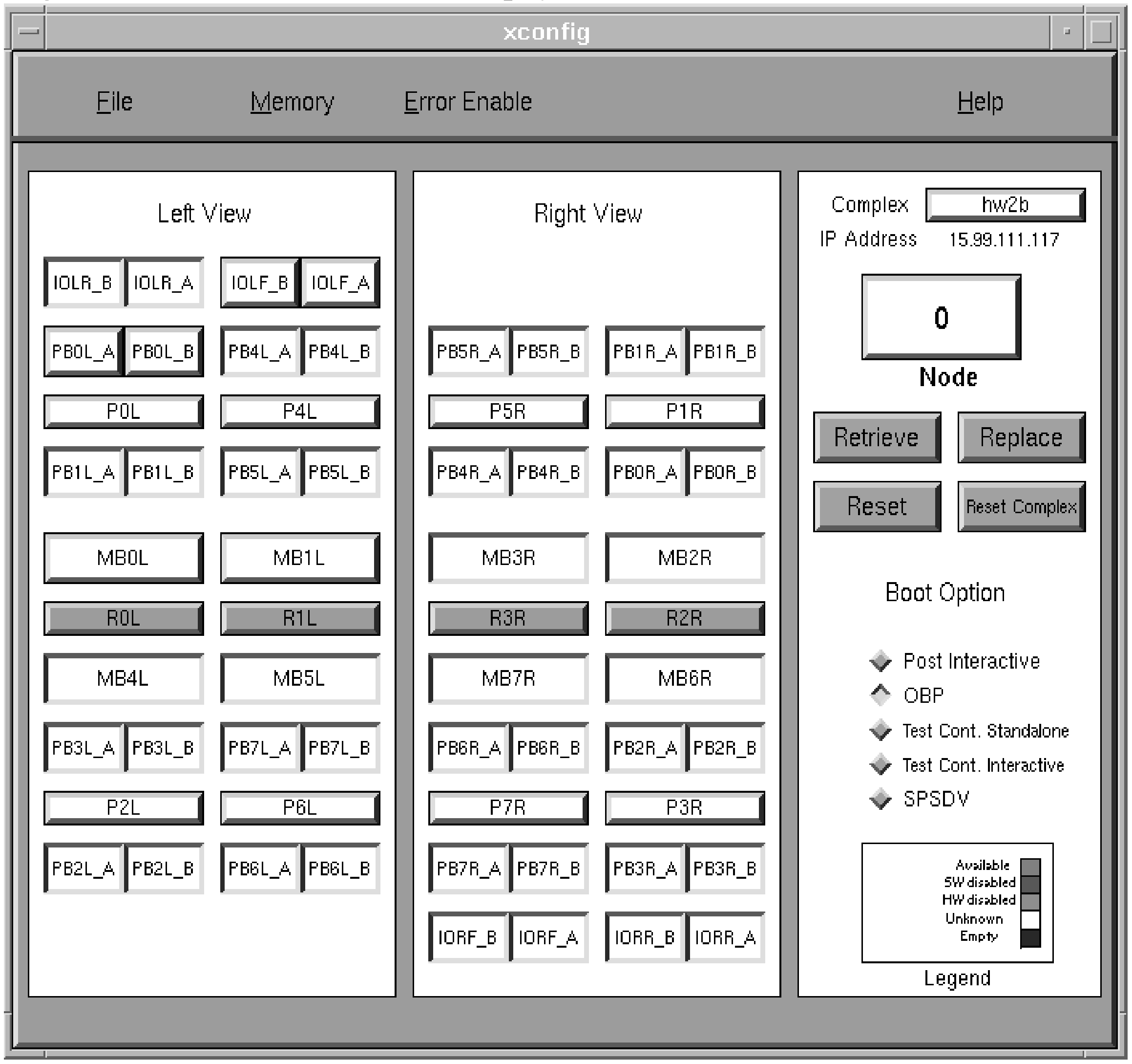
`xconfig` is started from a shell. Information on node 0 is read and interpreted to form the starting X-windows display shown in Figure 37.

The xconfig window appears on the system indicated by the environmental variable $DISPLAY. This may be overridden, however, by using the following command:

```
% xconfig -display system_name:0.0
```

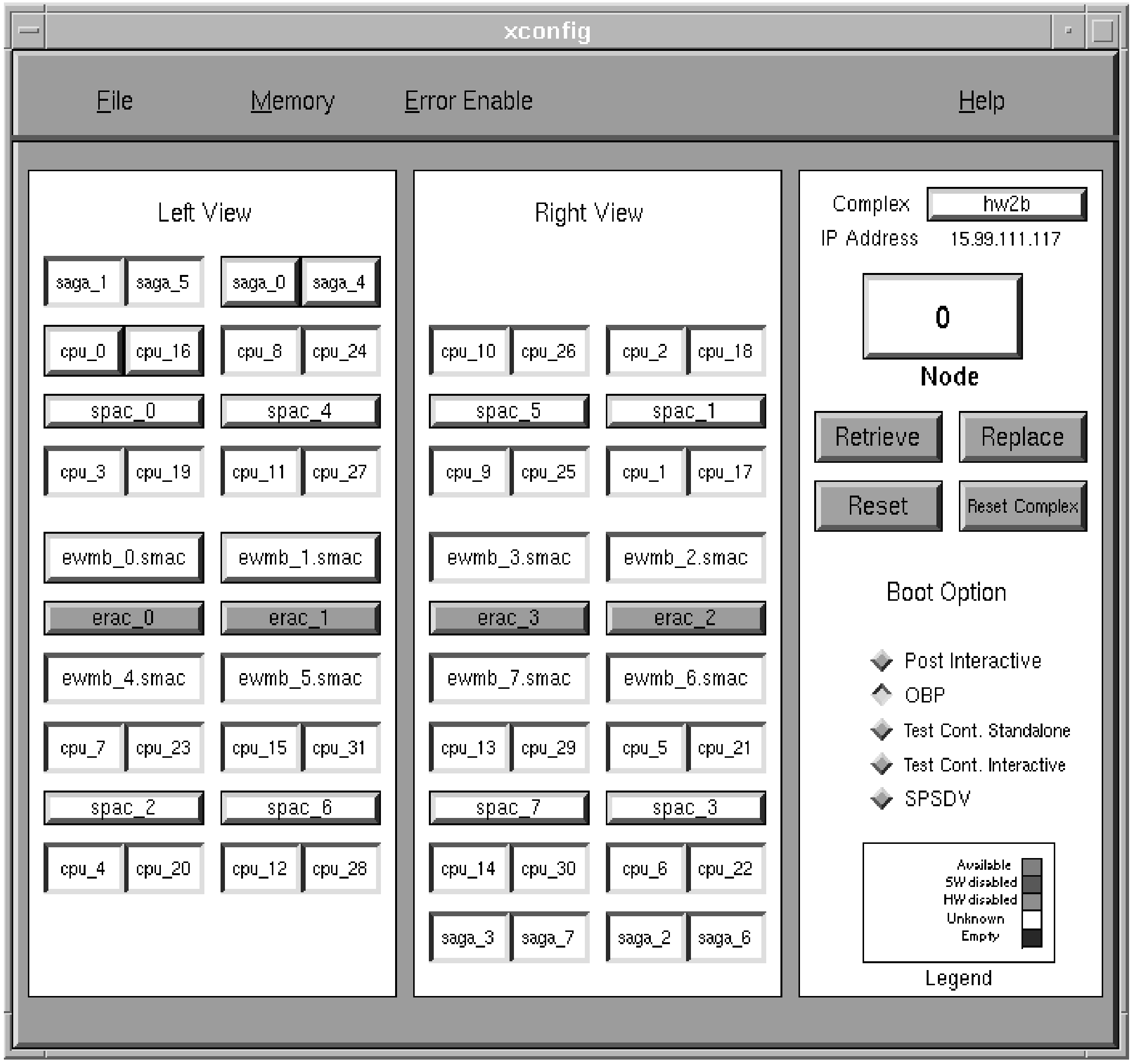
The xconfig window has two display views: one shows each component as a physical location in the server, the other shows them as logical names. Figure 37 and Figure 38 show the window in each view, respectively. To switch between views, click on the Help button in menu bar and then click the Change names option. See “Menu bar” on page 54.

Figure 37 **`xconfig`** **window—physical location names**



Complex hw2b
IP Address 15.99.111.117
0
Node

Figure 38 **`xconfig`** window—logical names



As buttons are clicked, the item selected changes state and color. There is a legend on the screen to explain the color and status. The change is recorded in the SSP's image of the node.

When the user is satisfied with the new configuration, it should be copied back into the node, and the node should be reset to enable the changes.

The main `xconfig` window has three sections:

- Menu bar—Provides additional capability and functions.
- Node configuration map—Provides the status of the node.
- Node control panel—Provides the capability to select a node and control the way data flows to it.

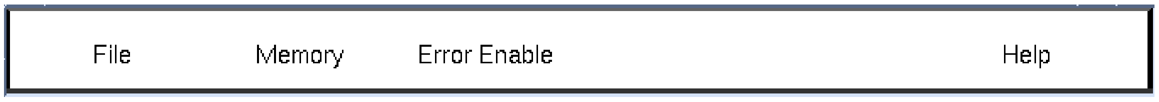
Menu bar

The menu bar appears at the top of the `xconfig` main window. It has four menus that provide additional features:

- File menu—Displays the file and exit options.
- Memory menu—Displays the main memory and CTI cache memory options.
- Error Enable menu—Displays the device menu options for error enabling and configuration.
- Help menu—Displays the help and about options.

The menu bar is shown in Figure 39.

Figure 39 **`xconfig` window menu bar**



The File menu provides the capability to save and restore node images and to exit `xconfig`.

The Memory menu provides the capability to enable or disable memory at the memory DIMM level by the total memory size and to change the network cache size on a multinode complex.

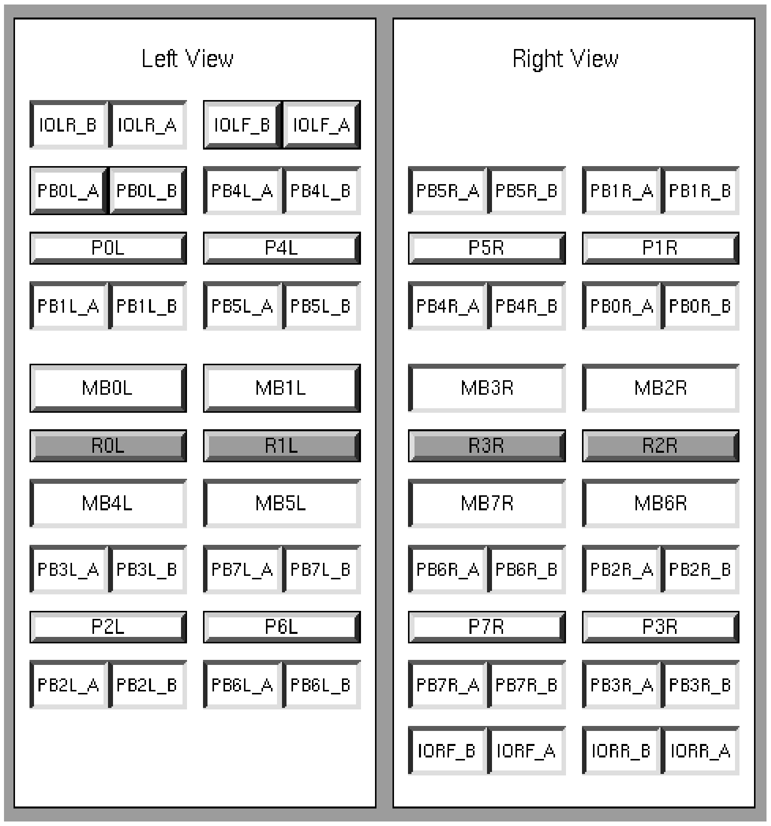
The Error Enable menu provides the capability to change a device's response to an error condition. This is normally only used for troubleshooting.

The Help menu provides a help box that acts as online documentation and also contains program revision information.

Node configuration map

The node configuration map is a representation of the left and right side views of a node as shown in Figure 40.

Figure 40 **`xconfig` window node configuration map**



The button boxes are positioned to represent the actual boards as viewed from the left and right sides. Each of the configurable components of the node is in the display. The buttons are used as follows:

- Green button—Indicates that the component is present and enabled.
- Red button—Indicates that the component is software disabled in the system.
- White button—Indicates that it is not possible to determine what the status of the component would be if POST were to be started.
- Blue box—Indicates that the component is either not present or fails the power-on self tests.
- Brown button—Indicates that POST had to hardware deconfigure this component in order to properly execute.
- Grey button—Indicates a hardware component that did not properly initialize.

The colors are shown in the legend box of the node control panel.

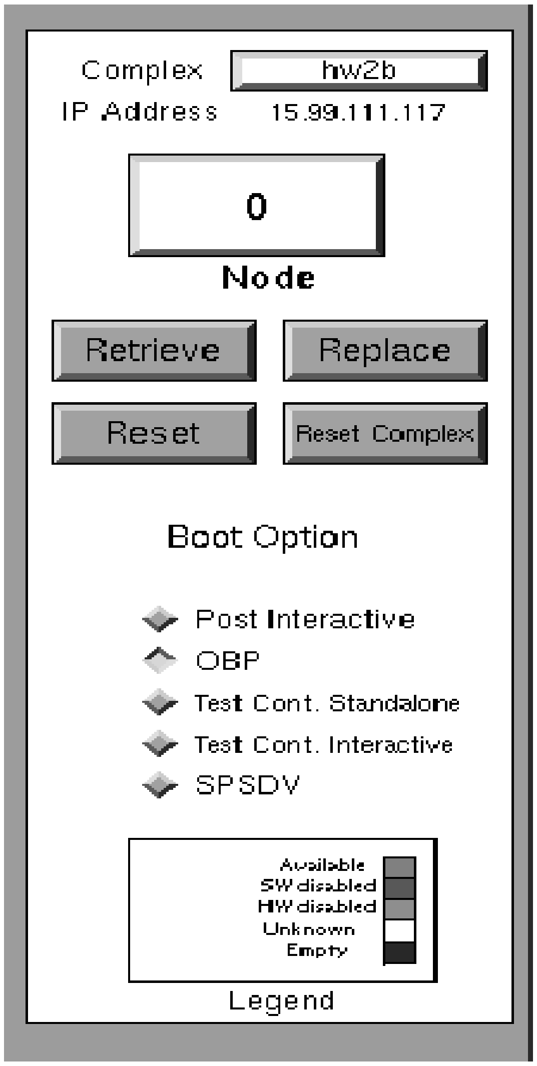
Components can change from enabled to disabled or disabled to unknown by clicking on the appropriate button with the left mouse button.

A multinode system requires an additional component on a memory board to enable the scalable coherent memory interface. This component can be viewed by right clicking the on the memory board button. The right mouse button toggles the memory board display between the memory board and the SCI device

Node control panel

The node control panel allows the user to select a node, select the stop clocks on an error function, select the boot parameters for a node and direct data flow between the node and the `xconfig` utility. It is shown in Figure 41.

Figure 41 **`xconfig`** **window node control panel**



The node number is shown in the node box. A new number can be selected by clicking on the node box and selecting the node from the pull-down menu. A new complex can be selected by clicking on the complex box and selecting it from the pull-down. A node IP address is displayed along with the node number and complex.

When a new node is selected and available, its data is automatically read and the node configuration map updated. The data image is kept on the SSP until it is rebuilt on the node using the Replace button. This is similar to the replace command on `sppdsh`.

Even though data can be rebuilt on a node, it does not become active until POST runs again and reconfigures the system. The Reset or Reset All buttons can be used to restart POST on one or all nodes of a system. A multinode system requires a reset all to properly function.

A Retrieve button is available on the node control panel to get a fresh copy of the parameters settings in the system. Clicking this button overwrites the setting local to the SSP and `xconfig`.

The Stop-on-hard button is typically used to assist in fault isolation. It stops all system clocks shortly after an error occurs. Only scan-based operations are available once system clocks have stopped.

The last group of buttons controls what happens after POST completes. The node can become idle or boot OBP, the test controller, or spsdv. The test controller and spsdv are additional diagnostic modes.

Configuration utilities

V2500/V2600 diagnostics provides utilities that assist the user with configuration management.

autoreset

`autoreset` allows the user to specify whether ccmd should automatically reset a complex after a hard error and after the hard logger error analysis software has run. `autoreset` occurs if a ccmd_reset file does not exist in the complex-specific directories

Arguments to `autoreset` arguments include `<complex_name> on` and `<complex_name> off or chk`.

The output of the `chk` option for a complex name of hw2a looks like:

```
Autoreset for hw2a is enabled.
```

or

```
Autoreset for hw2a is disabled.
```

NOTE `autoreset` determines the behavior of `ccmd` when it encounters an error condition. `ccmd` makes its decision whether to reset a complex immediately after running `hard_logger`. Enabling `autoreset` after `hard_logger` has run does not reset the complex.

est_config

`est_config` is a utility that builds the node and complex descriptions used by `est`. `est_config` reads support files at /spp/data/DB_RING_FILE, reads the electronic board identifier (COP chip) and scans to completely describe the node or complex. It also uses the hardware database created by `ccmd`. The data retrieved is organized and sorted into an appropriate node configuration file in the /spp/data/<complex name> directory.

An optional configuration directory can be specified using the `-p` argument. `est_config` works across all nodes unless a specific node or complex is requested with the `-n` option.

NOTE If there is a node_#.pwr file that is older than the node_#.cfg file, existing node configuration files do not need to be updated.

report_cfg

This utility generates a report summarizing the configuration of all nodes/complexes specified on the command line. The format of `report_cfg` is as follows:

```
report_cfg [<node id|complex> [<node id|complex> ...]]
```

`node id` may be a node number, IP name, or “all.” If no node ID is specified, the utility defaults to all nodes in the current complex.

One or more of the options in Table 6 must be specified:

Table 6 **`report_cfg` options**

`-d`	Show all details
`-s`	Show summary only
`-a`	Show summary and details (same as -d and -s)
`-A`	Show ASIC detail
`-i`	Show I/O detail
`-m`	Show memory detail
`-p`	Show processor detail

If the `report_cfg` tool detects any nodes of complexes that contain SCA DIMMS and some memory boards that are not populated with STACS, it generates a report.

Example configuration report:

```
The system inventory has determined that you’ll need to order 8 SCA Upgrade Kits
in order to connect this cabinet with other SCA cabinets. These upgrade kits are
available by additionally ordering opt. 010 of the required SCA HyperLink product
(A5518A or A5519A). You may also have to order additional memory DIMMs, memory
boards and or processor boards to meet the minimum requirements for a SCA
configuration. Refer to the HP 9000 V-Class Ordering Guide for details.
```

Effects of hardware and software deconfiguration

`report_cfg` counts all processors, STACs, SMACs, SAGAs and ERACs if POST has not marked them as empty. This results in ASICs and processors being included in the summary count even though they may have failed or have been deconfigured by software. This is necessary because POST deconfigures STACs in a single node configuration. To allow the tool to count these ASICs, it must report all ASICs that are installed, not just those enabled by POST.

`report_cfg` includes all DIMMs that POST has not marked as empty. If the user deconfigures a SMAC with software or the SMAC fails the POST selftest, POST marks the DIMMs on that SMAC as empty. If POST has written valid size information into the BCM for a DIMM, `report_cfg` reports the physical size reported by POST.

For example, if a node has both 80- and 88-bit DIMMs, POST reconfigures the 88-bit DIMMs to behave as 80-bit DIMMs, and the system logically behaves as if it has all 80-bit DIMMs. `report_cfg`, however, distinguishes (using the physical attribute in the BCM) between the 80- and 88-bit DIMMs in its reports.

Another example would be a system that contains 16 GBytes of memory but half of the DIMMs are deconfigured by software. `report_cfg` still reports that the system contains 16 Gbytes of memory.

report_cfg summary report

To obtain a system summary report, use the `-s` option for the command. The following is a sample summary report by `report_cfg`:

`report_cfg -s`

```
     Complex name    Complex serial number Node ID
  ------------------ --------------------- -------
               hw4a           USR1234567       0
               hw4a           USR1234567       2

          Cabinets:     2
        Processors:    30
  Processor boards:    30 (30 singles, 0 duals)
     Memory boards:    16
         TAC chips:    16
Enabled Memory (Mb):  8192
    88-bit 128Mbyte:   112
```

report_cfg ASIC report

To obtain a report on the ASICs in a complex, use the `-A` option. The following is a sample ASIC report by `report_cfg`:

`report_cfg -A`

```
      Complex       |Node#|        MIB COP         |          SCUB COP
====================+=====+=======================+=======================
         hw2a          0   A5074-60002 00  a 3845 A5074-60003 00  b 3830
         hw2a          2   A5074-60002 00  a 3840 A5074-60003 00  b 00XA


                                 +----- ASIC revisions ------+
     Complex        |Node| Slot  | PAC  | MAC  | TAC  | RAC  |
====================+====+=======+======+======+======+======+
         hw2a         0     0       2      2      1      2
         hw2a         0     1       2      2      1      2
         hw2a         0     2       2      2      1      2
         hw2a         0     3       2      2      1      2
         hw2a         0     4       2      2      1
         hw2a         0     5       2      2      1
         hw2a         0     6       2      2      1
         hw2a         0     7       2      2      1
         hw2a         2     0       2      2      1      2
         hw2a         2     1       2      2      1      2
         hw2a         2     2       2      2      1      2
         hw2a         2     3       2      2      1      2
         hw2a         2     4       2      2      1
         hw2a         2     5       2      2      1
         hw2a         2     6       2      2      1
         hw2a         2     7       2      2      1
```

report_cfg I/O report

To obtain an I/O report, use the `-i` option. The following is a sample I/O report by `report_cfg`:

`report_cfg -i`

```
      Complex       |Node#|        MIB COP         |          SCUB COP
====================+=====+=======================+=======================
         hw2a          0   A5074-60002 00  a 3845 A5074-60003 00  b 3830
         hw2a          2   A5074-60002 00  a 3840 A5074-60003 00  b 00XA


      Complex       |Node#|I/O board | COP
====================+=====+==========+=======================
        hw2a          0    IORF_B    A5080-60001 00  a 3821
        hw2a          0    IORF_A    A5080-60001 00  a 3821
        hw2a          2    IOLF_B    A5080-60001 00  a 3821
        hw2a          2    IOLF_A    A5080-60001 00  a 3821
```

report_cfg memory report

To obtain a report on the memory in a complex, use the `-m` option. The following is a sample memory report by `report_cfg`:

`report_cfg -m`

```
      Complex       |Node#|        MIB COP         |          SCUB COP
====================+=====+=======================+=======================
         hw2a          0   A5074-60002 00  a 3845 A5074-60003 00  b 3830
         hw2a          2   A5074-60002 00  a 3840 A5074-60003 00  b 00XA

                                               |  80-bit   |  88-bit
                                               |           |
                                               |   | 1 | 2 |   | 1 | 2
                 |Mem. |                       | 3 | 2 | 5 | 3 | 2 | 5
Complex     |Node|Board|          COP          | 2 | 8 | 6 | 2 | 8 | 6
============+====+=====+=======================+===+===+===+===+===+====
 hw4a          0   MB0L A5078-60003 01   a 00XB                        8
 hw4a          0   MB1L A5078-60003 01   a 00XB                        8
 hw4a          0   MB2R A5078-60003 01   a 00X2                        8
 hw4a          0   MB3R A5078-60003 01   a 00X2                        8
 hw4a          0   MB4L A5078-60003 01   a 00XA                        8
 hw4a          0   MB5L A5078-60003 01   a 00X2                        8
 hw4a          0   MB6R A5078-60003 01   a 00XA                        8
 hw4a          0   MB7R A5078-60003 01   a 00X2                        8
 hw4a          2   MB0L A5078-60003 01   a 00XA                        8
 hw4a          2   MB1L A5078-60003 01   a 3842                        8
 hw4a          2   MB2R A5078-60003 01   a 00XA                        8
 hw4a          2   MB3R A5078-60003 01   a 00XB                        8
 hw4a          2   MB4L A5078-60003 01   a 3843                        8
 hw4a          2   MB5L A5078-60003 01   a 00XD                        8
 hw4a          2   MB6R A5078-60003 01   a 00XB                        8
 hw4a          2   MB7R A5078-60003 01   a 00XD                        8
```

report_cfg processor report

To obtain a report on the processor in a complex, use the `-p` option. The following is a sample processor report by `report_cfg`:

`report_cfg -p`

```
      Complex       |Node#|        MIB COP         |          SCUB COP
====================+=====+=======================+=======================
         hw2a          0   A5074-60002 00  a 3845 A5074-60003 00  b 3830
         hw2a          2   A5074-60002 00  a 3840 A5074-60003 00  b 00XA

      Complex       |Node#|Processor | COP                    | CPU rev
====================+=====+==========+=======================+========
         hw2a         0    PB0L_A    A5077-60005 00  a 00XA    2.0
         hw2a         0    PB1R_A    A5492-60001 00  b 00XC    2.3
         hw2a         0    PB1L_A    A5491-60001 00  a 00XA    2.0
         hw2a         0    PB4L_A    A5492-60001 00  b 00XB    3.0
         hw2a         0    PB5L_A    A5492-60001 00  a 00XB    2.0
         hw2a         0    PB0L_B    A5077-60005 00  a 00XA    2.0
         hw2a         0    PB1L_B    A5491-60001 00  a 00XA    2.0
         hw2a         0    PB4L_B    A5492-60001 00  b 00XB    2.0
         hw2a         0    PB5L_B    A5492-60001 00  a 00XB    2.0

      Complex       |Node#|Processor | COP                    | CPU rev
====================+=====+==========+=======================+========

         hw2a         2    PB2L_A    A5491-60001 00  a 00XA    2.0
         hw2a         2    PB2R_A    A5492-60001 00  a 00XA    2.0
         hw2a         2    PB3R_A    A5491-60001 00  a 00XA    2.0
         hw2a         2    PB4L_A    A5492-60001 00  b 00XC    2.0
         hw2a         2    PB5L_A    A5491-60001 00  a 00XA    2.0
         hw2a         2    PB4L_B    A5492-60001 00  b 00XC    2.3
```

xsecure

`xsecure` is an application that helps make a V2500/V2600 class SSP secure from external sources. This tool disables modem and LAN activity to provide an extra layer of security for the V2500/V2600 system. `xsecure` may be run as a command line tool or an windows-based application.

In secure mode, all network LANs other than the tsdart bus are disabled and the optional modem on the second serial port will be disabled. When in normal mode all networks and modems are re-enabled.

If the command line `[-on | -off | -check]` options are used, `xsecure` does not use the GUI interface. These options allow the user to turn the secure mode on, off, or allow the user to check the secure mode status.

A simple button with a red or green secure mode indicator provides the user with secure mode status information. The red indicator shows that the secure mode process has begun. The label near the red button will inform the user when the SSP is secure. A green indicator and the appropriate label shows that the network is available and the SSP may be accessed through the ethernet port.

In order for `xsecure` to work properly the SSP, console cables, terminal mux and modems must be configured in specific ways. The SSP JTAG connections, OBP connections and an optional terminal mux must all be connected to the Diagnostic LAN and identified in the /etc/hosts file as tsdart-d. The sppconsole serial cable must be connected to serial port 1 and to node 0. An optional modem may be connected to serial port 2.

3 Power-On Self Test

POST is the Power On Self Test firmware for the V-Class platform. POST provides processor and system hardware initialization as well as providing basic processor selftest and utilities board SRAM pattern test capability. This chapter describes how POST initializes a node and handles power up errors.

Overview

Upon power up, all processors and hardware must be initialized before the node proceeds with booting. POST begins executing and brings up the node from an indeterminate state and then calls OBP.

None of the POST modules can be directly controlled via a user interface. Program control is provided by a set of configuration parameters (processing flags and variable definitions) stored in NVRAM by OBP, `do_reset`, or `xconfig`.

The error reporting modules display error codes for all fatal errors that occur during the POST execution. Any errors that can be recovered from, are reported to OBP. POST status is reflected on the LCD display.

POST performs the following tasks:

- Initializes and conditionally performs cache tests on each processor in the node
- Validates all shared data structures within the NVRAM.
- Initializes the core logic required to start OBP execution
- Determines node configuration
- Initializes all ASICs
- Initializes main memory
- Initializes multinode hardware
- Synchronizes nodes
- Sets up CTI cache
- Invokes OBP or the Test Controller.

Any fatal errors are reported to the user by way of the system LCD and the system console. POST passes node configuration and any options to OBP via shared data structures.

Reset

The following types of reset invoke POST:

- Power up reset— If a client had execution control before the power down condition, it invokes POST to initialize the hardware. POST initializes all hardware after a power up reset.
- Hard reset—If a client had execution control before the hard reset, it invokes POST to initialize the hardware. POST restarts execution and reinitializes all hardware.
- Soft reset—If a soft reset condition has occurred while POST is executing, POST restarts execution but does not initialize main memory.

  It invokes its interactive prompt.

POST modules

POST executes modules listed below in chronological order:

- Processor Initialization and Selftest—Each processor initializes itself on power up or reset in parallel with the other processors. Initialization includes setting values into the internal diagnostic registers, initializing the instruction and data caches, clearing a scratch ram area for stack and data storage, and enabling high-priority machine checks (HPMC), low-priority machine checks (LPMC), and transfer of control (TOC). Selftest includes instruction set tests, instruction and data cache RAM tests and TLB RAM tests.
- SCUB Hardware Initialization—POST clears any error state in the SCUB, initializes the SCUB hardware registers and DUART, and initializes and optionally tests the SRAM on the SCUB (see scuba_test_enable).
- Non-volatile Configuration Data Verification—POST verifies the checksum of all shared data regions in a battery-backed-up SRAM (NVRAM). POST verifies only the regions it shares with other modules, such as OBP, and those private to POST. If a region fails, it is rebuilt using default values.
- Hardware Configuration Determination—POST determines the ASIC installations status and verifies that each installed ASIC responds to register accesses. If one does not, it is reported as failing. POST then configures the system to utilize the maximum amount of installed hardware based on the V2500/V2600 hardware configuration rules.
- Node Hardware (ASIC) Initialization—POST sets up all available hardware with the proper operating mode(s) enabled. Routing is configured for the current hardware population.
- Node Main Memory Initialization—POST probes all installed memory boards for memory installation status. It then enables each memory board as a 2-, 4-, or 8-board configuration based on V2500/V2600 configuration rules. All remaining memory boards are configured to have the same logical memory population. It then initializes main memory in parallel, using up to eight processors using initialization hardware in the memory controllers.

- Page Deallocation Table Support—POST supports reading the page deallocation table (PDT) and remapping memory if it detects a bad page in the HPUX good-memory region. It updates all entries to reflect the new memory layout if remapping occurs. It also clears PDT if memory hardware change is detected.
- Multinode Initialization—If the system is configured as part of a multinode complex, POST initializes and configures the system for multinode operation.
- Multinode Configuration Checking—POST first verifies that the multinode configuration parameters are set properly. If a parameter is not set up properly, multinode initialization will fail.

  The V2500/V2600 only supports a subset of memory configurations in a multinode system. POST checks the current memory configuration against the list of valid multinode-capable configurations and de-allocates memory as required to establish a valid configuration.

  If any of the nodes in a multinode system contains a memory configuration that uses four bus span, then all nodes in the system must also use only four bus span. Four bus span occurs when a node is 1/4 populated with memory. To check for this condition, each node reports its local memory configuration to all other nodes in the system. If any node reports a four bus span configuration, then all other nodes deallocate memory to a 1/4 populated configuration, and booting will continue.
- Multinode Hardware Initialization—The STAC configuration registers are setup first, and then Ring Reset is asserted and deasserted. This causes the hardware to automatically perform the ring initialization sequence. Finally, POST verifies that all rings have achieved the Run state.
- CTI Cache Initialization—The SMACs are configured to enable the normal interleave CTI-cache region as specified by the `cti_cache_size` parameter. This region of memory is removed from local memory and reinitialized as CTI cache.
- Multinode Hardware Verification—POST Verifies that the next node on each ringlet has the correct node ID and STAC ID, performs a data pattern test on each cable, and then verifies coherent memory access (both reads and writes) from each node to all other nodes in the complex.

- Time Of Century Synchronization Routing—POST determines the Time Of Century monarch node. Then the internode signal routing of the Time Of Century counter is configured. POST does NOT start the Time Of Century counters.
- Client Boot—POST cleans up any residual state from POST execution and boots the client specified in `boot_module`. POST can boot clients with all processors or with just with the monarch processor leaving the other processors in an idle loop.

Interactive mode

POST provides a command line interface for configuration and debugging. The command line interface is invoked if `boot_module` is set to “interactive,” by a soft reset, or a TOC during POST execution.

If the command line interface is entered in a multinode complex, after the nodes have synchronized, the console functionality for all nodes is controlled through the console of the lowest numbered node (usually node 0). The node that receives the commands from the console is indicated by the first digit in the command prompt. Console control is switched between nodes using the node command.

Interactive mode commands

POST supports the following commands at the line prompt:

- `help`—Displays a list of supported commands and their usage.
- `banner`—Displays the POST version and build information.
- `reset [loader|post|soft]`—Causes POST to perform a reset of the node. If `loader` is specified, then the node is hard reset and executes the firmware loader PDCFL. If `post` is specified, the node is hard reset and executes POST. If `soft` is specified, the node is soft reset and executes POST.
- `node <node_number>`—Switches control of the console to the node indicated by node_number.
- `dcm`—Dumps the configuration map from NVRAM and displays the hardware status of the machine, showing the hardware that is enabled, deconfigured, or failing.

  The ASIC entries have the following values:

Table 7 **SIC entry values**

SIC entry	Value
Unknown	0xff
Pass	0x01
Fail	0x10
Deconfigured	0x20
Empty	0x30
SW deconfigured	0x40
Installed	0x50

A DIMM entry is eight bits and has the following format:

`DIMM Entry: [0|1|2|3|45|67]`

The bits are defined as follows:

- In use (bit 0)—Some part of the DIMM is being used in main memory.
- Software deconfigured (bit 1)—The DIMM has been deconfigured by the user.
- Physical bit size (bit 2)—0 = 80 bits and 1 = 88 bits).
- Logical bit size (bit 3)—0 = 80 bits and 1 = 88 bits).
- Physical RAM size (bits 4:5)—0 = 0 Mbytes, 1 = 16 Mbytes, 2 = 64 Mbytes, 3 = 128 Mbytes.
- Logical RAM size (bits 6:7)—0 = 0 Mbytes, 1 = 16 Mbytes, 2 = 64 Mbytes, 3 = 128 Mbytes.

- `setenv [parm] [value]`—Sets the configuration parameter specified by `parm` to the `value`.
- `memmap`—Displays any row that has been logically remapped due to PDT entries, failing DIMM, or software deconfigured DIMMs.
- `printenv [parm]`—Prints the value of the configuration parameter specified by `parm`. If no parameter is specified, then all are printed.

- `get_opt [asic_type [asic_number]]`—Dumps the option mode bits for the ASIC type specified by `asic_type`. If an `asic_number` is also specified, then only the values of the ASIC are printed. If an `asic_number` is not specified, then all ASICs of that `asic_type` are dumped. If no `asic_type` is specified, then all ASICs are dumped.
- `pdt`—Dumps the current Page Deallocation Table (PDT) contents.
- `clear_pdt`—Clears out all entries in the PDT.
- `bcast`—Broadcasts a command to all nodes. If `bcast` is specified before a valid command, the command is executed on the controlling node, then the command is sent to each of the other nodes, one at a time.
- `nodemap`—Prints the current Node ID and prompts for new values for the multinode parameters. If a **<CR>** is typed, the old value is retained. If the SSP parameters structure is rebuilt, all these parameters are set to the default values.

  The system connects as a two dimensional matrix of nodes. The node ID describes the node's position in the matrix. The 4-bit node ID is composed of 2-bits of X address and 2-bits of Y address.

  For example, node 2's location in the matrix is 0,2 (binary: 00 10). Node 6 is 1,2 (binary: 01 10). Nodes do not have to be numbered sequentially along the ring, in a two node configuration nodes 0 (0,0) and node 2 (0,2) are present; node 1 (0,1) is not.

  The maximum supported configuration is a four-node, two-X ring, two-Y ring system.

  The `nodemap` parameters are as follows:

  - Node ID—The ID of the current physical node. This is only reported and can not be changed with a POST command.
  - Number of nodes in complex [2]—The total number of nodes in the complex. This is set to 1 for single node. [default: 1]
  - Number of X-rings in complex\ [0]—The total number of X rings in a given dimension. [default: 0]
  - Number of Y-rings in complex\ [1]—The total number of Y rings in a given dimension. [default: 0]
  - Hypernode bit mask data [0x00000005]—A bitmask marking all nodes detected by POST during the last boot. The bitmask is little-endian notation, with the least significant bit representing node 0.

This parameter is set to a default starting value based on the other nodemap parameters each boot if generate hypernode bitmask is TRUE. Any node failing multinode initialization is removed from this bitmask in the other nodes during the boot process. If multinode initialization fails on this node, only the current node is marked in the bitmask.

If *generate hypernode bitmask* is FALSE, the value is fixed, and only the nodes marked in the current bitmask are synchronized. Any failing nodes are also left in the bitmask. Forcing a fixed value using generate hypernode bitmask is for debug only. [default: (bit representing current node ID)]

- *Generate hypernode bitmask* (0 = FALSE, 1 = TRUE) [1]—Setting this to TRUE causes POST to initialize the hypernode bitmask to a default starting value based on the other nodemap parameters. Setting it to FALSE causes POST to retain the initial value to start multinode initialization. [default: 1 (TRUE)]

Configuration parameters

The following parameters control the operation of POST:

- `ts_ip`—Specifies the SSP IP address for LAN messaging. The value should be set to the IP address of the diagnostics LAN port on the SSP. [default: 15.99.111.99]

Table 8 **Name of SSP IP address for listed utilities**

Utility	Parameter name
OBP	ts-ip#
POST	ts_ip
`sppdsh`	ts_ip

- `scub_ip`—Specifies the IP address used for LAN interface hardware on the utilities (SCUB) board. This is the IP address that POST, OBP, and the Test Controller use for LAN messaging with the SSP. [default: none]

Table 9 Name of scub IP address for listed utilities

Utility	Parameter name
OBP	obp-ip#
POST	scub-ip
`sppdsh`	scub_ip
`ts_config`	scub_ip

- `cti_cache_size`—Specifies the amount of memory, in megabytes, to reserve in the node for CTI cache. This is used only in multinode configurations. [default: 0 Mbytes]

Table 10 Name of CTI cache size IP address for listed utilities

Utility	Parameter name
OBP	cti-cache-size
POST	cti_cache_size
`sppdsh`	cti_cache_size

- `boot_module`—Specifies which client to turn execution control over to at the completion of POST execution. [default: OBP]

Table 11 Name of boot module for listed utilities

Utility	Parameter name
OBP	boot-module
POST	boot_module
`sppdsh`	boot_module

- `selftest_enable`—Enables selftest control if the processor selftest is executed during POST start-up execution. [default: true]

Table 12 **Name of selftest enable for listed utilities**

Utility	Parameter name
OBP	selftest?
POST	selftest_enable
`sppdsh`	selftest_enable

- `scuba_test_enable`—Controls whether the SCUB SRAM is tested before initialization. This affects the processor initialization, since processors test and initialize their own stack region. It also affects the SCUB initialization and core LAN SRAM initialization steps for the monarch. [default: true]

Table 13 **Name of scuba test enable for listed utilities**

Utility	Parameter name
OBP	scubatest?
POST	scuba_test_enable
`sppdsh`	scuba_test_enable

- `master_error_enable`—Determines whether POST will enable errors or not. This is used in conjunction with `use_error_overrides` to determine how errors are enabled. [default: true]

Table 14 **Name of master error enable for listed utilities**

Utility	Parameter name
OBP	master-error-enable?
POST	master_error_enable
`sppdsh`	master_error_enable

- `use_error_overrides`—Determines if POST will use the built-in defaults for errors or the user error overrides. This is only checked if `master_error_enable` is enabled. [default: false]

Table 15 **Name of use error overides for listed utilities**

Utility	Parameter name
OBP	use-error-overrides?
POST	use_error_overrides
`sppdsh`	use_error_overrides

- `force_monarch`—Determines if POST will force the monarch selection to a specific processor. The processor is specified in `monarch_number` [default: false]

Table 16 **Name of force monarch for listed utilities**

Utility	Parameter name
OBP	force-monarch?
POST	force_monarch
`sppdsh`	force_monarch

- `monarch_number`—Specifies the monarch processor when `force_monarch` is enabled. [default: 0]

Table 17 **Name of monarch number for listed utilities**

Utility	Parameter name
OBP	monarch#
POST	monarch_number
`sppdsh`	monarch_number

- `node_local_size`—Specifies the amount of physical addresses, in megabytes, starting at address 0 and ending at `node_local_size`. Addresses falling in the node local region are forwarded to the local node's memory, not to Node 0's memory. [default: 128 MB]

Table 18 **Name of local node size number for listed utilities**

Utility	Parameter name
OBP	local-node-size
POST	node_local_size
`sppdsh`	node_local_size

Messages

POST has three types of messages: LCD, console, and error. This section discusses each type.

LCD messages

The LCD messages are described in Chapter 1, “Introduction.”

Console messages

POST provides several messages that are displayed on the SSP console. This section describes these console messages.

Type-of-boot

This message reports the type of boot for the current POST execution, and the node ID and monarch processor.

Typical message:

```
POST Hard Boot on [0:PB1R_A]
```

Version and build

This message reports the version and build information for POST.

Typical message:

```
HP9000/V2500_V2600 POST Release 2.0, compiled 1998/11/04 14:33:12
```

Processor probe

This message reports where the processors are in the system. Only available processors are reported; any failing or deconfigured processors are not listed. Processors in this list may be deconfigured if they share a Runway bus with a processors that fails the probe or is deconfigured.

Typical message:

```
Probing CPUs:     PB0L_A PB0R_A PB1R_A PB1L_A PB2L_A PB2R_A PB3R_A PB3L_A
                  PB4L_A PB4R_A PB5R_A PB5L_A PB6L_A PB6R_A PB7R_A PB7L_A
                  PB0L_B PB0R_B PB1R_B PB1L_B PB2L_B PB2R_B PB3R_B PB3L_B
                  PB4L_B PB4R_B PB5R_B PB5L_B PB6L_B PB6R_B PB7R_B PB7L_B
```

Utility board initialization

This message reports that the Utilities board SRAM reserved for missing or unavailable processors is being initialized. The SRAM is tested prior to initialization if `scuba_test_enable` is true.

Typical message:

```
Completing core logic SRAM initialization.
```

Main memory initialization

This message reports that main memory initialization has started.

Typical message:

```
Starting main memory initialization.
```

Memory probe

This message reports the status of the memory boards as they are detected and probed for DIMMs

Typical message:

```
Probing memory: MB0L MB1L MB2R MB3R MB4L MB5L MB6R MB7R
```

Installed memory

This message reports the total memory installed and available, in megabytes.

Typical message:

```
Installed memory: 2048 MBs, available memory: 2048 MBs
```

Main memory initialization started

This message marks the beginning of main memory initialization.

Typical message:

```
Initializing main memory.
```

Parallel memory initialization

This message reports that main memory initialization will be done with multiple processors in parallel. Only printed if more than one processor is available for memory initialization.

Typical message:

```
Parallel memory initialization in progress.
```

Memory initialization progress

This message reports the results of the initialization, the initializing processor, and the memory board for each board available in the node. Each character indicates the physical location of the DIMM and the logical size of the DIMM. The memory information is encoded as follows:

Value	Memory Type
.	16 MBytes
:	64 MBytes
I	128 MBytes
_	Empty
#	Hardware deconfigured
$	Software (user) deconfigured

Typical message:

```
                  r0         r1         r2         r3

PB0L_A MB0L [.... ....][.... ....][____ ____][____ ____]
PB1R_A MB1L [.... ....][.... ....][____ ____][____ ____]
PB2L_A MB2R [.... ....][.... ....][____ ____][____ ____]
PB3R_A MB3R [.... ....][.... ....][____ ____][____ ____]
PB4L_A MB4L [.... ....][.... ....][____ ____][____ ____]
PB5R_A MB5L [.... ....][.... ....][____ ____][____ ____]
PB6L_A MB6R [.... ....][.... ....][____ ____][____ ____]
PB7R_A MB7R [.... ....][.... ....][____ ____][____ ____]
```

Building main memory map

This message indicates that a map in SCUB SRAM is being generated to report main memory population to OBP.

Typical message:

```
Building main memory map.
```

Main memory initialization complete

This message indicates that main memory initialization is complete.

Typical message:

```
Main memory initialization complete.
```

Multinode memory initialization

This message indicates that the node is configured in a multinode system and is starting the multinode initialization and synchronization process.

Typical message:

```
Starting multinode initialization.
```

Multinode memory configuration determination

Each node broadcasts to and receives from other nodes memory configuration information used to perform cross-node configuration checking for the system.

This message indicates that this internode configuration broadcasting is in progress.

Typical message:

```
Collecting memory configuration from nodes: 0,6,4,2
```

Memory could be deconfigured in one node, based on the configuration in one of the other nodes.

ERI ring initialization

This message indicates that multinode hardware initialization has started. POST is currently waiting for all rings to achieve the Run state.

Typical message:

```
Initializing ERI rings.
```

The following message indicates when a node has verified that all ERI Rings have achieved the Run state and broadcasted its status. The list of synchronized nodes indicates which nodes have successfully initialized their rings.

Typical message:

Synchronizing nodes: 0,4,2,6

CTI cache initialization

This message marks the beginning of CTI cache initialization.

Typical message:

```
Initializing CTI cache.
```

Parallel CTI cache initialization

This message reports that CTI cache initialization will be done with multiple processors in parallel. It is printed only if more than one processor is available for CTI cache initialization.

Typical message:

```
Parallel CTI cache initialization in progress.
```

Memory and CTI cache initialization progress

This message reports the results of the initialization, the initializing processor, and the memory board for each board available in the node. Each character indicates the logical location of the DIMM and the type (or state) of the memory region inhabiting the particular DIMM.

The cache information is encoded as follows:

Value	Memory Type
L	Set up as all local memory
C	Set up as all CTI cache
M	Set up as a mixture of local memory and CTI cache
_	Empty

#	Hardware deconfigured
$	Software (user) deconfigured

Typical message:

```
                 r0          r1          r2          r3
PB0L_A MB0L [LLLL ____][MMMM ____][____ ____][____ ____]
PB1R_A MB1L [LLLL ____][MMMM ____][____ ____][____ ____]
PB2L_A MB2R [LLLL ____][MMMM ____][____ ____][____ ____]
PB3R_A MB3R [LLLL ____][MMMM ____][____ ____][____ ____]
PB4L_A MB4L [LLLL ____][MMMM ____][____ ____][____ ____]
PB5R_A MB5L [LLLL ____][MMMM ____][____ ____][____ ____]
PB6L_A MB6R [LLLL ____][MMMM ____][____ ____][____ ____]
PB7R_A MB7R [LLLL ____][MMMM ____][____ ____][____ ____]
```

Remote memory testing

This message indicates the start of the remote memory test using coherent reads and writes.

Typical message:

```
Verifying remote memory access.
```

The following message indicates when each node listed has successfully completed the remote memory access test.

Typical message:

```
Synchronizing nodes: 0,6,4,2
```

TOC routing

This message indicates the start of Time Of Century routing procedure.

Typical message:

```
Enabling Time of Century synchronization routing.
```

The following message indicates when each node listed has successfully completed the TOC routing test.

Typical message:

```
Synchronizing nodes: 0,6,4,2
```

System control to boot client

This message indicates that system control is being handed off to the specified boot client.

For example, one of the following:

```
Booting OBP
Booting DIAG
Booting SPSDV
Booting RDR dumper
Booting Boombox
```

Interactive boot

This message indicates that POST is entering its interactive mode. POST provides a console interface for system configuration and debug.

Typical message:

```
Booting Interactive
```

The following is the POST interactive prompt and is only seen if `boot_module` is set to interactive.

Typical message:

```
[0:PB0L_A] POST>
```

Chassis codes

The processor initialization and selftest functions in POST report status and error information with chassis codes. These chassis codes are shared with `cpu3000` and are documented in the man page with the exception of the following POST-specific codes:

0x6103C	The processor is executing it's processor initialization code
0x22025	The processor encountered a data error while loading the processor Icache
0x22026	The processor encountered a tag error while loading the processor Icache.

Error messages

POST provides error messages that are printed to the console. This section describes these error messages.

SSP parameters failure

This message reports the that SSP parameters structure failed the checksum and was rebuilt to the default structure.

Typical message:

```
Teststation Parameters checksum FAILED, rebuilding...
```

This node may be forced with the `sppdsh reboot <node>` default command.

Configuration map failure

This message indicates that the configuration map structure failed the checksum and was rebuilt to defaults. Any user deconfigured hardware state is lost.

Typical message:

```
Configuration Map checksum FAILED, rebuilding...
```

Configuration parameters failure

This message indicates that the configuration parameters structure failed the checksum and was rebuilt to the default structure. Any user overrides from the default value, for parameters that have a default, is lost. Some parameters have no default and retain the value in NVRAM. Since NVRAM could be corrupt, these values could be invalid.

Typical message:

```
Configuration Parameters checksum FAILED, rebuilding...
```

ASIC probe failure

This message indicates that the specified ASIC failed the probe. The status of any components that must be accessed through this component are unknown, and they are not available if installed.

Typical message:

```
Failed probe of P1R
Unable to determine status of PB1R_A PB1L_A PB1L_B PB1L_A IOLR_B
```

Memory board deconfiguration

This message indicates that the specified memory board is deconfigured. This can be due to a memory board being found on one side of memory without a corresponding pair, since boards must be used in pairs of even/odd boards. This can also occur when a memory board has no usable memory.

Typical message:

```
Deconfiguring: MB5L
```

Illegal memory board configuration

This message indicates that there is an unallowed memory board configuration. Memory boards can only be used in two-, four-, or eight-board configurations. In the following example, a six-board configuration was detected, and two boards will be deconfigured.

Typical message:

```
Illegal 6 memory board configuration.
```

Processor initialization failure

This message indicates that the specified processor failed to perform the step described during parallel main memory initialization. The monarch processor completes the initialization assigned the failing processor.

Typical message:

```
PB1R_A timed out during encache memory init code
PB1R_A timed out during memory initialization
PB1R_A timed out during idle request after memory init
PB0L_B failed to go idle after memory init
Unable to force CPU PB2L_A into idle loop
```

Monarch completing memory initialization

This message indicates that the monarch processor is completing the memory initialization assigned to the specified processor.

Typical message:

```
Using Monarch to initialize memory assigned to PB2L_A
```

PDT checksum failure

This message indicates that the page deallocation table structure failed the checksum and was rebuilt to defaults. All bad page information is lost.

Typical message:

```
Page Deallocation Table (PDT) checksum FAILED, rebuilding...
```

Memory hardware change detected

This message indicates that POST detected a change in memory hardware and cleared all entries in the PDT.

Typical message:

```
Detected a hardware change, clearing the Page Deallocation Table (PDT).
```

Memory remapped

This message indicates that POST remapped memory to achieve HP-UX good memory region. This occurs when a bad page is marked within the good memory region.

Typical message:

```
Memory was re-mapped to achieve HP/UX good memory region.
```

Contiguous memory block not found

This message indicates that POST could not find a block of contiguous memory to place at address zero to achieve good memory. POST will report no main memory to the OBP for this failure.

Typical message:

```
HP/UX good memory region could not be achieved.
```

Processor not reported

This message indicates that a processor failed to mark itself in the system report register.

Reporting happens early in the sequence, and this failure usually indicates the processor has failed to execute any instructions.

Typical message:

```
Failed probe of PB1R_B, CPU failed to report in.
```

Processor initialization/selftest failure

This message indicates that a processor failed at some point during initialization or selftest. The chassis code for the module that failed is reported.

Typical message:

```
Failed probe of PB1R_B
chassis code 0x6103C
```

Processor not responding to interrupt

This message indicates that a processor properly initialized itself but did not respond to an external interrupt

Typical message:

```
Failed probe of PB1R_B
cpu PB1R_B did not respond to an interrupt
```

Shared Runway bus failure

This message indicates that an available processor has been deconfigured because it shares a Runway bus with a processor that failed to probe

Typical message:

```
cpu PB1R_A deconfigured due to PB1R_B shutdown.
```

New monarch processor selected

This message indicates that the previous monarch processor was deconfigured and a new one was selected. The new monarch continues the initialization of the rest of the system

Typical message:

```
INFO: New monarch selected: PB0R_A
```

New monarch processor not found

This message indicates that the other processor on the Runway bus with the monarch processor was deconfigured or failed and another suitable processor could not be found to replace the monarch.

Typical message:

```
WARNING: The monarch shares a Runway bus with a failed
cpu.
```

Multinode console error messages

The following multinode error messages could be printed to the console as POST executes the multinode initialization procedure. The field values shown are for example purposes only.

80-bit DIMM mode set

This message indicates that there are one or more 80-Bit DIMMs in the node. 80-Bit DIMMs can not be used in a multinode system because there are too few bits to hold the multinode tag data. 88-Bits DIMMs are needed.

Use `dcm -d all` <node> to locate and replace these DIMMs.

Typical message:

```
80-bit DIMM mode is set.
```

Invalid CTI cache state

The parameter `cti_cache_size` is set to an invalid value. Use `setenv` to correct this problem.

Typical message:

```
CTI cache is disabled
```

Invalid local node size

The parameter `node_local_size` is set to an invalid value. Use `setenv` to correct this problem.

Typical message:

```
Force node ID region size is disabled
```

Corrupt multinode parameter

This message indicates that one of the parameters vital to multinode initialization is corrupt. Use `nodemap` to correct this problem.

Typical message:

```
Invalid multinode configuration database entry (0x02)
```

POST may still attempt to boot multinode, in which case the following message appears:

```
Forcing to maximum multinode configuration.
```

```
POST is enabling all possible nodes in a complex and will
deconfigure non-existent or non-working nodes as the
timeout is reached.
```

Invalid memory configurations

The following statements refer to possible problems with the current memory configuration(s).

Typical message:

```
No main memory available
```

```
Invalid multi-node memory configuration
```

```
Re-configuring memory to a valid multinode configuration
```

```
Unable to achieve a valid multinode memory configuration
```

POST may attempt to deconfigure a node's memory so that it can work with the other nodes in its complex. If POST is successful, the new memory configuration is used and multinode initialization proceeds normally. If not, the `dcm` command may need to be invoked to assess the current state of memory on each node. A good rule of thumb is to identically populate all memory boards in the entire complex.

STAC deconfigured

The following message indicates that the STAC chip has been deconfigured either by hardware or software.

Typical message:

```
The STAC on MB0l is deconfigured
```

STAC initialization failure

The following message indicates that the STAC chip did not pass ASIC probing or initialization.

Typical message:

```
MB01 does not have a working STAC installed
```

CTI cache initialization failure

The following message indicates that the current memory configuration is too small to allocate any of it for CTI cache.

Typical message:

```
Unable to configure CTI cache for current memory configuration
```

Memory board mismatch

The following message indicates that the memory board configuration of the indicated node is not identical to the other nodes. This is not a supported configuration. Valid multinode memory configurations are available in the Configuration Guide.

Typical message:

```
Memory board config mismatch on node 2
```

Invalid CTI cache size

The following message indicates that the parameter `cti_cache_size` is set to an invalid value for the current memory configuration. The largest valid size less than the value specified will be used, and the `cti_cache_size` parameter will be changed in NVRAM.

The CTI cache size cannot exceed 1/2 of the total memory in each node.

Typical message:

```
INFO: Invalid CTI cache size for current memory configuration (4096MB), resetting
cti_cache_size to 1024 MB
```

Interleave mismatch

The following message indicates that POST discovered an interleave mismatch or that a DIMM did not pass the memory probe. A good rule of thumb is to identically populate all memory boards in the entire complex. It may be necessary to use the `dcm` command to assess the current state of memory on each node.

Typical message:

```
Found mixed 4 and 8 bus interleave span
Found mixed 8 and 4 bus interleave span
Found mixed 2 and 4 bank interleave span
Found mixed 4 and 2 bank interleave span
```

ERI ring failures

Several typical error messages relating to the ERI rings are presented in this section.

The following messages are typical of those received for a particular failure mode of an ERI ring. The bit missing in the "act:" field indicates the failing ring on this node. The bits are ordered from right to left from 0 to 7.

Typical messages:

```
Unable to RESET rings: (exp: x=0x00 y=0xff, act: x=0x00 y=0xfd)
Unable to clear RESET: (exp: x=0x00 y=0xff, act: x=0x00 y=0xfd)
Unable to achieve RUN state: (exp: x=0x00 y=0xff, act: x=0x00 y=0xfd)
Unable to enable rings: (exp: x=0x00 y=0xff, act: x=0x00 y=0xfd)
Unable to disable rings: (exp: x=0x0 y=0xff, act: x=0x00 y=0xfd)
Unable to enable global errors: (exp: x=0x00 y=0xff, act: x=0x00 y=0xfd)
```

In these examples, the failure is on ring 1 in the Y direction.

The following message indicates that an HPMC was trapped while performing the actions indicated in the example on an off-node CSR or memory location indicated.

Typical messages:

```
MB0l cable pattern test failed: HPMC while writing 0x000000fc.00050000
MB0l cable pattern test failed: HPMC while reading 0x000000fc.00050000
Store/flush/read test failed: HPMC while reading address 0x00000010.00000020
Store/flush/read test failed: HPMC while clearing address 0x00000010.00000020

Store/flush/read test failed: address 0x10.00000020, expected 0xaa55aa01, actual
0xaa51aa01
```

POST cannot continue booting multinode. The ERI cables may be misrouted or broken or one (or more) of the System Configuration CSRs is corrupt. Another possibility is that the tag state for this memory line is corrupt.

The following message indicates that one of the ERI cables connected to MB0l is broken. It is impossible to know which one, because the error could have occurred during the write (output port) or the read (input port).

Typical messages:

```
MB0l cable pattern test failed: expected 0xaaaaaaaa, actual 0xaa2aaaaa
```

The following message indicates that MB0l cable connection test failed: the test expected MB0l, actual received MB4l.

Typical messages:

```
MB0l cable connection test failed: expected node 0, actual node 4
```

The cable routing is probably incorrect. Check that all ERI cables are connected to and from their designated ports.

Ethernet packet error

The following message indicates that an ethernet packet error has occurred. As a result, nodes in the complex may not synchronize properly.

Typical message:

```
Ethernet packet error
```

Communications time-out

The following message indicates that the nodes listed are not responding. These nodes will be removed from the expected node mask. Multinode initialization will continue normally if possible.

Typical message:

```
Node communication time-out for nodes: 4 6
```

4 Test Controller

The Test Controller is an EEPROM-based utility that provides the environment for executing the off-line diagnostic tests. It is controlled through parameters stored in the NVRAM on the Utilities board. The Test Controller reads these parameters to determine its execution mode, the number processors to test, which SMACs, SPACs, and SAGAs to include in the testing, which subtests to run, and other diagnostic test-specific information.

Test Controller modes

There are three basic operational modes for this utility:

- Stand-alone mode
- Interactive mode
- I/O Utility mode

In stand-alone mode, `cxtest` invokes the Test Controller. The Test Controller reads test parameters from NVRAM (these parameters are written into NVRAM by `cxtest` before it invokes the Test Controller), executes the test and subtests specified in NVRAM, and sets a completion bit in NVRAM when the test and subtests are finished. `cxtest` is described in Chapter 5.

In interactive mode, a user interface allows the user to select the processors to test, to select the subtests to run, and to examine error information. The user interface is a set of menus described in this chapter. The Test Controller loops waiting for the start command. Prior to issuing the start command, any global and/or processor-specific parameters can be modified. When all tests have completed, the Test Controller waits for the next start command. Any combination of parameter and tests may be modified and executed.

In I/O Utility mode, the Test Controller loads and subsequently executes a firmware utility module from the SSP. The SSP utility `tc_ioutil` identifies the utility module to be loaded. `tc_ioutil` updates an NVRAM location with the file name of the utility module to be loaded. See `tc_ioutil` and `dfdutil` in Chapter 12, “Utilities,” for more details.

User interface

The Test Controller provides for the control of off-line diagnostic test execution. It utilizes a set of parameters to control its operation. The parameters consist of the following:

- Global set that controls the overall operation of the Test Controller
- Test set (one per test) that controls how the tests are executed by the Test Controller
- CPU parameters (one per processor) that contain status information about the tests executing on each processor

All these parameters are in NVRAM.

The user interface allows the user to modify parameters that reside in NVRAM, thereby controlling the operation of the Test Controller. It also allows the user to select which subtests are executed on each of the processors and modify the test parameters, as well as any other test information.

The Test Controller user interface consists of two basic menus. The first is the main menu that gives the user the following capabilities:

- Modify the POST boot selection
- Control operation of the Test Controller
- Display the current global parameter selections
- Display processor summary
- Switch processors
- Go to the Test Configuration menu

The second menu is the processor Test Control menu that provides the following capabilities:

- Select classes of subtests to execute
- Select subtests to execute
- Specify pause enables
- Specify whether or not to loop
- Specify the test and/or subtest error counts

- Read and write the 128 words of test specific information
- Select the hardware to test
- Display the current parameter selections

Main menu

Test Controller Main Menu

```
MAIN Menu commands

     0=Quit Test Controller
     1=Begin Test Controller Execution
     2=Halt Test Controller Execution
     3=Resume Test Controller Execution
     4=Switch CPU
     5=POST Boot Selection
     6=Execution Mode Selection
     7=Global Parameter Display
     8=CPU Summary Display
     9=Display CPU Errors
     A=Test Selection Menu
     B=Test Configuration Menu
     C=Debugging Menu
     D=Display revision

Enter Command:
```

Each main menu selection is defined as follows:

- 0=Quit Test Controller—Terminates the Test Controller utility and either reboots the system (to POST and then to the selected program) or halts the system depending on the current value of the POST Boot Selection flag.
- 1=Begin Test Controller Execution—Starts the Test Controller utility executing the specified subtests on the selected processors. The entire system is started from the beginning.
- 2=Halt Test Controller Execution—Suspends temporarily the operation of the Test Controller. This command may be entered at any time. Only the Test Controller is halted; subtests on other processors continue to execute.

- 3=Resume Test Controller Execution—Continues execution from the point of interruption.
- 4=Switch CPU—Allows the user to start the Test Controller on the specified processor. The previously used processor starts executing the command wait loop code. The user is prompted for the processor as follows:

  Enter CPU (0-1f):
- 5=POST Boot Selection—Prompts the user for the new value with the following prompt:

```
  Boot Option (1=OBP, 2=TC Interactive, 3=TC Standalone 4=Loader, 5=SPSDV, 6=RDR
Dumper, 7=Boombox, 8=POST Interactive):
```

  For all values, POST boots to ESPDV. The Test Controller performs a hard reset to POST when the Test Controller terminates.
- 6=Execution Mode Selection—Allows the user to select the mode for executing the subtests. The two options are serial and parallel. The following prompt queries for the selection:

  ```
  Execution Mode Selection (0=serial, 1=parallel):
  ```
- 7=Global Parameter Display—Displays the available hardware components, the current “POST Boot Selection” value, and the current “Execution Mode Selection” value. The display is shown below:

Example Global Parameter display

```
Enter command: 7

MAIN Menu - Global Parameters Display
    CPUs:                 0* 1* 2* 3* 4* 5* 6* 7* 8* 9* A* B* C* D* E* F*

                         10*11*12*13*14*15 16*17*18*19*1A*1B*1C*1D 1E*1F
    SPACs:                0* 1* 2* 3* 4* 5* 6* 7*
    SMACs:                0* 1* 2  3  4  5  6* 7*
    STACs:                0  1  2  3  4  5  6  7
    SAGAs:                0  1* 2  3  4  5* 6  7
    Execution Mode Selection:  Serial  Parallel*
    POST Boot Selection:       TC Interactive
```

The asterisks denote the component has passed POST processing and is available for diagnostic testing (see processors 0-3 and SMACs 0, 2, 4, and 6 in the display).

- 8=CPU Summary display—Displays a summary of the current processor and testing information. An example of the display is shown below:

Example CPU summary display

```
MAIN Menu - CPU Summary Display
    Total Failures = 0
  Configuration Map
  =================
  CPUs :  0   1   2   3*  4   5   6   7   8   9  10  11  12  13  14  15
  CPUs : 16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
 SPACs :  0*  1*  2*  3*  4*  5*  6*  7*
 SMACs :  0*  1*  2   3   4   5   6   7
 STACs :  0   1   2   3   4   5   6   7
 SAGAs :  0*  1   2   3   4*  5   6   7

                      FAIL
  CPU STATE           COUNT SUBTEST TEST NAME
  === =====           ===== ======= =========
    0 Not Available   n/a   n/a     n/a
    1 Not Available   n/a   n/a     n/a
    2 Not Available   n/a   n/a     n/a
    3 Idle            n/a   n/a     n/a
    4 Not Available   n/a   n/a     n/a
    5 Not Available   n/a   n/a     n/a
    6 Not Available   n/a   n/a     n/a
    7 Not Available   n/a   n/a     n/a
    8 Not Available   n/a   n/a     n/a
    9 Not Available   n/a   n/a     n/a
   10 Not Available   n/a   n/a     n/a
   11 Not Available   n/a   n/a     n/a
   12 Not Available   n/a   n/a     n/a
   13 Not Available   n/a   n/a     n/a
   14 Not Available   n/a   n/a     n/a
   15 Not Available   n/a   n/a     n/a
   16 Not Available   n/a   n/a     n/a
   17 Not Available   n/a   n/a     n/a
   18 Not Available   n/a   n/a     n/a
   19 Not Available   n/a   n/a     n/a
   20 Not Available   n/a   n/a     n/a
   21 Not Available   n/a   n/a     n/a
   22 Not Available   n/a   n/a     n/a
   23 Idle            n/a   n/a     n/a
   24 Idle            n/a   n/a     n/a
   25 Not Available   n/a   n/a     n/a
   26 Not Available   n/a   n/a     n/a
   27 Not Available   n/a   n/a     n/a
   28 Not Available   n/a   n/a     n/a
   29 Not Available   n/a   n/a     n/a
   30 Not Available   n/a   n/a     n/a
   31 Not Available   n/a   n/a     n/a
Hit <ENTER> key to return to the MAIN Menu:
```

Each available hardware component is marked with an asterisk just to the right of its number (see processors 0-3 and SMACs 0, 2, 4, and 6 in the display).

The possible states in the CPU Summary Display are described in Table 19.

Table 19 **Processor States**

CPU State	Description
Not Available	Denotes processor is not available for testing.
Running	Denotes a test is currently running on this processor.
Idle	Denotes that no test is running on this processor.
Ready	Denotes last subtest completed and ready for next subtest.
Test Completed	Denotes test completed execution on this processor.
Error Detected	Denotes test halted due to an error condition on this processor.
Test Timeout	Denotes a timeout detected during test execution on this processor; the test is halted.
HW Reqs Not Met	Denotes the hardware selected does not meet the minimum hardware required for executing the test.
User Halted	Denotes user halted test.
Unexpected HPMC	Denotes running test caused an HPMC; the test is halted.
SW Deconfigured	Denotes test automatically halted testing on this processor, because of a software restriction.

- 9=Display CPU Errors—Displays the errors for the currently selected processor. When selected, the user is prompted for the processor as follows:

  ```
  Enter CPU [0-1f]:
  ```

Example Test Parameters display.

```
Test Configuration Menu - Test Parameters Display
  CPUs:  ( 1)  0  1  2  3* 4  5  6  7  8  9  A  B  C  D  E  F
              10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
  SPACs: ( 1)  0* 1* 2* 3* 4* 5* 6* 7*
  SMACs: ( 0)  0* 1* 2  3  4  5  6  7
  STACs: ( 0)  0  1  2  3  4  5  6  7
  SAGAs: ( 0)  0* 1  2  3  4* 5  6  7
  Nodes: ( 1)
  Loop Enable:           ON   OFF*
  Loop Count:            00
  Test Error Count:      01
  Pause Test Start:      ON   OFF*
  Pause Test End:        ON   OFF*
  Pause Subtest Start:   ON   OFF*
  Pause Subtest End:     ON   OFF*
  Pause On Fail:         ON*  OFF
```

There are four fields:

- Date/Time—Date and time the error was logged.
- Subtest—Failing subtest number.
- Event Code—32-bit event code.
- Error Message—40-character error message

- A=Test Selection Menu—Invokes the menu. This menu allows the user to select which tests to execute. An asterisk before the test name denotes that it has been selected. Only the memory test is selected. Selecting options 1 through A toggles the current state for that particular test. Selecting option 0 returns the user to the main menu shown below:

Test Selection display

```
MAIN Menu - Test Selection Display

0=Return to Main Menu
1=*Memory test
2=not available
3=not available
4=ERI test
5=I/O test
6=CPU selftests
7=not available
8=not available
9=not available
A=not available
Please enter number of test:
```

- B=Test Configuration Menu—Switches the user to the Configuration menu shown below for the specified test.
- C=Debugging Menu—Invokes the Debugging Menu shown below that allows the user to read or write to any memory location on the node and dump various data.

Example Debugging menu

```
MAIN Menu - Debugging Menu

0=Return to Main Menu
1=Read 32-bit Memory Location
2=Write 32-bit Memory Location
3=Read 64-bit Memory Location
4=Write 64-bit Memory Location
5=Read ECC from memory line
6=Read Tag From Memory Line
7=Print Test Revision
8=Dump last HPMC Info
9=Clear All Error Info
A=Dump TC CPU Info Structure
B=Dump TC Test Info Structure
C=Reset SONIC Interface
D=Dump SONIC Registers
Enter number of activity:
```

- Selection 1 queries for the 40-bit address to read as follows:

  ```
  Enter 40-bit address:
  ```

- Selection 2 queries for the 40-bit address and then for the 32-bits of data to write:

  ```
  Enter 32-bit data:
  ```

- Selection 3 queries for the 40-bit address to read.
- Selection 4 queries for the 40-bit address to write, and then for the 64-bits of data to write as follows:

  ```
  Enter Upper 32-bits:
  Enter Lower 32-bits:
  ```

- Selections 5 and 6 query for the 40-bit address for which to display the ECC or tag of that memory line.
- Selection 7 queries the user for the test index with the following prompt:

  ```
  Enter test index [1-A]:
  ```

- Selection 8 prints information for the last HPMC that occurred on the specified processor.
- Selection 9 clears all stored error information.

NOTE Use caution using selection 9 as there is no undo function.

- Selection A queries the user for the processor index as follows:

  ```
  Enter cpu index [0-1f]:
  ```

- Selection B queries the user for the test index with the prompt shown.

  ```
  Enter test index [1-A]:
  ```

Test Configuration menu

The Test Configuration menu is shown below:

Test Configuration menu

```
Test Configuration Menu

0=Return to Main Menu A=Hardware Selection Menu
1=Display ClassesB=Loop Enable
2=Display SubtestsC=Loop Count
3=Select ClassesD=Test Error Count
4=Select SubtestsE=Pause at Test Start
5=Read All Test ParametersF=Pause at Test End
6=Read One Test ParameterG=Pause at Subtest Start
7=Write Test ParameterH=Pause at Subtest End
8=Reset ParametersI=Pause On Failure
9=Display Test Configuration

Enter command:
```

Each Test Configuration menu selection is defined as follows:

- 0=Return to Main Menu—Returns the user to the Main menu.
- 1=Display Classes—Displays the current class definitions for this diagnostic. An example of the display is shown below:

Test Configuration menu - Class display

```
Test Configuration Menu - Class Display

Class  Description
0         class 0  description
1         class 1  description
.             .
.             .
n         class n description
```

- 2=Display Subtests—Displays the current subtest definitions for this diagnostic. An example of the display is shown in the example below.

Test Configuration menu - Subtest display

```
Test Configuration Menu - Subtest Display

Subtest     Description
0           subtest 0 description
1           subtest 1 description
.            .
.            .
n*         subtest n description
```

An asterisk following the subtest number denotes that it is selected for execution. For example, see the “n subtest n description” line.

- 3=Select Classes—Allows the user to specify which classes of subtests to execute. These selections are in addition to any subtests selected. The following prompt is displayed:

  `Enter class number:`

  The user must enter one of the following:

  - An optional operator followed by a class number, for example 2, +2 or -2.
  - Multiple class numbers (class, +class, -class are allowed) separated by commas or spaces, for example 1,2,3.

    The class numbers are decimal.

- 4=Select Subtests—Allows the user to specify which subtests to execute. These selections are in addition to any classes selected. The following prompt is displayed:

  `Enter subtest number or subtest range:`

  The user may enter one of the following:

  - A single subtest number.
  - A subtest range which consists of two numbers separated with a dash and is inclusive. An example of a valid range entry is 100-200.
  - An optional operator followed by a subtest selection, for example 2, +2, -2, +100-200, or -100-200.

  The subtest numbers are decimal values only.

- 5=Read All Test Parameters—Reads all 128 words that make up the test parameter set and displays this information. These test parameters reside in NVRAM and are defined by the particular test. An example of the display is shown in the example below:

Test Configuration menu - Test Parameters

```
Test Configuration Menu - Test Parameters

Word     Value          Word     Value
----     -----          ----     -----
 0       a5a5a5a5        10      00000000
 1       a5a5a5a5        11      00000000
 2       a5a5a5a5        12      00000000
 3       a5a5a5a5        13      00000000
 4       00000002        14      00000000
 5       00000000        15      00000000
 6       88888888        16      00000000
 7       88888888        17      00000000
 8       00000000        18      00000000
 9       00000000        19      00000000

Hit <ENTER> key to continue; ‘q’ to quit :
```

If enter is pressed, the next 20 parameters are displayed in the same format as above.

- 6=Read One Test Parameter—Allows the user to read a single test parameter.
- 7=Write Test Parameter—Allows the user to modify any one of the test parameter words. The user is first requested to specify which word is to be modified:

```
Specify Test Parameter word [0-127]:
```

After specifying the word, the user is then prompted for the new value:

```
New value for Test Parameter word xx:
```

- 8=Reset Parameters—Resets some of the parameters to their default values. Table 20 lists the affected parameters and their default values.

Table 20 **Parameter Defaults**

Parameter	Default value
Loop Enable	0
Loop Count	0
Test Error Count	1
Pause At Test Start	0
Pause At Test End	0
Pause At Subtest Start	0
Pause At Subtest End	0

- 9=Display Test Configuration—Displays the current values of the processor parameters. An example of the display is shown in the example below. An asterisk denotes the current selections. For Example, processor 0 is selected.

  This minimum hardware requirements information is enclosed in parentheses after the hardware type label and denotes the number of that type required. In the example below, 1 processor, 1 SPAC (the one associated with the selected processor), and 1 SMAC are needed.

  The Test Controller compares the selected hardware components versus these minimum requirements to determine if the test can be executed. Unless these minimum requirements are met, the test cannot be executed.

Test Configuration menu - Test Parameters display

```
Test Configuration Menu - Test Parameters Display
  CPUs:  ( 1)  0  1  2  3* 4  5  6  7  8  9  A  B  C  D  E  F
              10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
  SPACs: ( 1)  0* 1* 2* 3* 4* 5* 6* 7*
  SMACs: ( 0)  0* 1* 2  3  4  5  6  7
  STACs: ( 0)  0  1  2  3  4  5  6  7
  SAGAs: ( 0)  0* 1  2  3  4* 5  6  7
  Nodes: ( 1)
  Loop Enable:           ON   OFF*
  Loop Count:            00
  Test Error Count:      01
  Pause Test Start:      ON   OFF*
  Pause Test End:        ON   OFF*
  Pause Subtest Start:   ON   OFF*
  Pause Subtest End:     ON   OFF*
  Pause On Fail:         ON*  OFF
```

A=Hardware Selection menu—Invokes Hardware Selection menu shown in the example below:

Test Configuration menu - Hardware Selection menu

```
Test Configuration Menu - Hardware Selection Display

0=Return to Test Configuration Menu
1=CPU Selection
2=SPAC Selection
3=SMAC Selection
4=STAC Selection
5=SAGA Selection
6=Node Selection
Enter Command:
```

The selections of the Hardware Selection menu are defined as follows:

- 1-5=<hardware> Selection—Selects the appropriate controller. The following prompt is displayed:

  `Select <hardware>:`

  The user must enter one of the following:

  - An optional operator followed by a hardware component number, for example 2, +2 or -2.

    - Multiple hardware component numbers separated by commas or spaces, for example 1,+2,-3.

    The format 2, or +2, denotes to use this hardware component in testing. The format -2 denotes not to use this hardware component in testing. The 1 and +1 formats are equivalent, and leaving a hardware component out of the list is equivalent to the -n format.

    As an example, to select all the even processors the user would enter:

    0,2,4,6,8,a,c,e, 10, 12, 14, 16, 18, 1a, 1c, 1e

  - 6=Node Selection—Allows the user to enter 7-bit node ids. The format is similar to that used for processors, i.e. an optional operator can be used and multiple entries are allowed. The following prompt is used:

    ```
    Enter Node Ids:
    ```

- B=Loop Enable—Allows the user to modify the value of the loop enable flag, which causes the Test Controller utility to loop on all selected subtests when the last subtest is executed. The user is prompted for the new value as follows:

  ```
  Loop Enable (0=disabled, 1=enabled):
  ```

- C=Loop Count—Allows the user to specify the number of times to loop through the selected subtests. It is only used if the “Loop Enable” flag is set. The user is prompted for the new value (a decimal number) as follows:

  ```
  Loop Count:
  ```

- D=Test Error Count—Allows the user to modify the maximum number of test errors that can occur before the Test Controller utility terminates execution of subtests on this processor. The user is prompted for the new value (a decimal number) as follows:

  ```
  Test Error Count value (1-127, N=no limit):
  ```

  A value of N means that there is no limit to the number of errors that can occur.

- E=Pause at Test Start—Allows the user to modify the pause at test start flag. This flag results in the Test Controller pausing the testing on this processor just prior to starting the process of determining the first subtest to execute. The user is prompted for the new value as follows:

```
Pause at Test Start (0=disabled, 1=enabled):
```

- F=Pause at Test End—Allows the user to modify the pause at test end flag. This flag results in the Test Controller pausing the testing on this processor after last subtest has completed execution and all cleanup is complete. The user is prompted for the new value as follows:

```
Pause at Test End (0=disabled, 1=enabled):
```

- G=Pause at Subtest Start—Allows the user to modify the pause at subtest start flag. This flag results in the Test Controller pausing the testing on this processor just prior to starting the execution of the current subtest. The user is prompted for the new value as follows:

```
Pause at Subtest Start (0=disabled, 1=enabled):
```

- H=Pause at Subtest End—Allows the user to modify the pause at subtest end flag. This flag results in the Test Controller pausing the testing on this processor just after detecting that the current subtest has completed execution. The user is prompted for the new value as follows:

```
Pause at Subtest End (0=disabled, 1=enabled):
```

- I=Pause on Failure—Allows the user to specify whether testing should halt when the specified number of failures is detected. The default is to halt. The user is prompted for the new value as follows:

```
Pause in Failure (0=disabled, 1=enabled)
```

Example of running diagnostics from Test Controller command line

This example shows how to run `mem3000` from the Test Controller command line within the following scenario:

- Configure `mem3000` to run on a system with four memory boards installed.
- Set the classes and subtests to be executed.
- Run the tests.
- View the results.

Configuration

To execute the scenario, perform the procedures in this and the following sections:

Step 1. From the Test Controller main menu shown below, select the Test Selection Menu, option A:

Test Controller main menu

```
MAIN Menu commands

0=Quit Test Controller
1=Begin Test Controller Execution
2=Halt Test Controller Execution
3=Resume Test Controller Execution
4=Switch CPU
5=POST Boot Selection
6=Execution Mode Selection
7=Global Parameter Display
8=CPU Summary Display
9=Display CPU Errors
A=Test Selection Menu
B=Test Configuration Menu
C=Debugging Menu
D=Display revision
```

Step 2. From the Test Selection menu shown below, select Memory test, option 1.

Test Controller Test Selection menu

```
MAIN Menu - Test Selection Display

    0= Return to Main Menu
    1= Memory test
    2= not available
    3= not available
    4= ERI test
    5= I/O test
    6= CPU Selftests
    7= not available
    8= not available
    9= not available
    A= not available
Please enter number of test:
```

Step 3. Select option 0 to return to the Main Menu

Step 4. Select option B, Test Configuration Menu, from the Main Menu.

Step 5. From the menu, select Memory test, option 1.

This opens the Test Configuration menu shown below:

Test menu

```
1=*Memory test
2= not available
3= not available
4= ERI test
5= I/O test
6= CPU Selftests
7= not available
8= not available
9= not available
A= not available
Please enter number of test:
```

Step 6. From the Test Configuration menu shown below, select the Hardware Selection menu, option A.

Test Controller Test Selection menu

```
Test Configuration Menu

     0=Return to Main Menu         A=Hardware Selection Menu
     1=Display Classes             B=Loop Enable
     2=Display Subtests            C=Loop Count
     3=Select Classes              D=Test Error Count
     4=Select Subtests             E=Pause at Test Start
     5=Read All Test Parameters    F=Pause at Test End
     6=Read One Test Parameter     G=Pause at Subtest Start
     7=Write Test Parameter        H=Pause at Subtest End
     8=Reset Parameters            I=Pause On Failure
     9=Display Test Configuration

Enter command:
```

Step 7. From the Hardware Selection menu shown below, select CPUs, option 1.

Selecting CPUs from Hardware Selection menu

```
Test Configuration Menu - Hardware Selection Display
   0=Return to Test Configuration Menu
   1=CPU Selection
   2=SPAC Selection
   3=SMAC Selection
   4=STAC Selection
   5=SAGA Selection
   6=Node Selection
```

Step 8. At the following prompt:

```
Select CPUs: 0 2
```

Select the number of processors (CPUs).

After the number of processors is chosen, the Hardware Selection menu reappears.

Step 9. From this menu select Return to Test Configuration menu, option 0.

Step 10. From the Test Configuration menu, the user can select option 9 to view the changes.

Selecting classes and subtests

To select test classes and subtest, perform the following procedure:

Step 1. From the Test Configuration menu, select Select Classes, option 3.

Step 2. From the following prompt, select the test classes:

```
Enter class number:
```

Step 3. From the Test Configuration menu, select Display Subtests, option 2.

The subtest menu shown below lists all available subtests:

`mem3000` Subtests menu

```
100*   Diagnostic CSR Read/Write Test
101*   Other SMAC CSR Read/Write Test
110*   Memory Data Read/Write Test
120*   Memory ECC Read/Write Test
130*   Memory Tag Read/Write Test
140*   Memory Initialization Test
150*   First 32 Memory Lines Test
190*   DIMM Probe Test
200*   Tag Bank Test
210*   Tag Addressing Test
211*   Tag Byte Uniqueness Pattern Test
230*   Tag March-C Pattern #1 Test
231*   Tag March-C Pattern #2 Test
232*   Tag March-C Pattern #3 Test
233*   Tag March-C Pattern #4 Test
234*   Tag March-C Pattern #5 Test
235*   Tag March-C Pattern #6 Test
236*   Tag March-C Pattern #7 Test
237*   Tag March-C Pattern #8 Test
238*   Tag March-C User Data Pattern Test
300*   Memory Bank Test
310*   Memory Addressing Test
311*   Byte Uniqueness Pattern Test
330*   Memory March-C Pattern #1 Test
331*   Memory March-C Pattern #2 Test
332*   Memory March-C Pattern #3 Test
333*   Memory March-C Pattern #4 Test
334*   Memory March-C Pattern #5 Test
335*   Memory March-C Pattern #6 Test
336*   Memory March-C Pattern #7 Test
337*   Memory March-C Pattern #8 Test
338*   Memory March-C User Data Pattern Test
400*   Memory Load/Store Test
410*   Memory Data Flush Transaction Test
420*   Memory Semaphore Transaction Test
500*   TAG ECC Single Error Correction Test
501    DATA ECC Single Error Correction Test
502    ECC Single Error Correction Test
510*   ECC Double Error Detection (coherent) test
520*   ECC Double Error Detection (non-coherent) test
530*   ECC Disable Test
600*   Memory Access Protection Test
610*   Memory Tag Test I
640*   80 vs. 88 Bit DIMM Test
```

Step 4. Select all appropriate subtests. Table 21 lists the test patterns for subtests 230 through 238.

Table 21 Test patterns for subtests 230-238 and 330-338

Subtest	Patterns

230, 330	`0x7f7f7f7f7f7f7f7f`
	`0x8080808080808080`

231, 331	`0xbfbfbfbfbfbfbfbf`
	`0x4040404040404040`

232, 332	`0xdfdfdfdfdfdfdfdf`
	`0x2020202020202020`

233, 333	`0xefefefefefefefef`
	`0x1010101010101010`

234, 334	`0xf7f7f7f7f7f7f7f7`
	`0x0808080808080808`

235, 335	`0xfbfbfbfbfbfbfbfb`
	`0x0404040404040404`

236, 336	`0xfdfdfdfdfdfdfdfd`
	`0x0202020202020202`

237, 337	`0xfefefefefefefefe`
	`0x0101010101010101`

238, 338	`0xa5a5a5a5a5a5a5a5`
	`0x5a5a5a5a5a5a5a5a`

Selecting Display Subtests, option 2, from the Test Configuration Menu reflects the changes.

Starting tests

To run the tests selected from the Test Controller main menu, select Begin Test Controller Execution, option 1. The output is shown in the example below:

Example of `mem3000` execution

```
% Enter command: 1

 Execution Starting MEM3000: Subtest 234
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
...........................................................

 Execution Completed - No errors detected
```

Viewing the results

To review the results of the test, select CPU Summary Display, option 8, from the Main menu.

An example of the results is shown below:

Example CPU Summary display

```
  Configuration Map
  =================
  CPUs :  0   1   2   3*  4   5   6   7   8   9  10  11  12  13  14  15
  CPUs : 16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
 SPACs :  0*  1*  2*  3*  4*  5*  6*  7*
 SMACs :  0*  1*  2   3   4   5   6   7
 STACs :  0   1   2   3   4   5   6   7
 SAGAs :  0*  1   2   3   4*  5   6   7
                     FAIL
  CPU STATE          COUNT SUBTEST           TEST NAME
  === =====          ===== =======           =========
  0  Test Completed   0     150     MEM3000 - EEPROM based memory tests
  1  Idle             n/a   n/a    n/a
  2  Test Completed   0     150     MEM3000 - EEPROM based memory tests
```

5 cxtest

The cxtest program is a graphical front end and a command line interpreter for the test controller. It is a standalone program that runs independently of any diagnostic tests loaded in the EEPROM on the Utilities board.

Overview

The `cxtest` program runs on the SSP and communicates with the test controller via the NVRAM configuration parameters on the Utilities board. Depending on the command line, `cxtest` either starts the graphics display or runs as a command line interpreter.

The GUI provides an easy and flexible way to select and run tests. The main screen has six drop down menus. The six menus are, File Menu, Test Menu, Global Parm Menu, Command Menu, System Configuration Menu, and Help Menu. The menus contents are present in the sections below.

The advantage of using the command line over the graphics user interface is that there is a somewhat greater control over the order in which tests are run and the ability to run the tests from a script. The disadvantage is that the user must be more aware of what tests are installed and how to run them.

The test controller must be in the stand-alone mode in order for `cxtest` to be able to communicate with it. To run the test controller in stand-alone mode, run the following command:

```
do_reset <node> tc_standalone
```

For example:

```
do_reset 0 tc_standalone
```

When `cxtest` is invoked, it first retrieves system information from NVRAM and EEPROM on the Utilities board. This information includes:

- Tests loaded
- Parameters required for those tests
- Hardware configuration

The `cxtest` program works with the test controller to execute tests based on the options selected by the user. It performs the following functions:

- Looping
- Dispatching tests
- Configuring hardware

- Retrieving error information from the test controller

The test controller operates in the standalone mode when running in conjunction with `cxtest`. This is true whether one is using the command line version of `cxtest` or the graphics interface.

Graphics interface

To start the `cxtest` graphics interface, specify the `-d` option on the command line as follows:

```
% cxtest -d
```

This causes `cxtest` to open a window on the display. Where the window is displayed is set by the environment variable $DISPLAY. This cannot be changed on the command line.

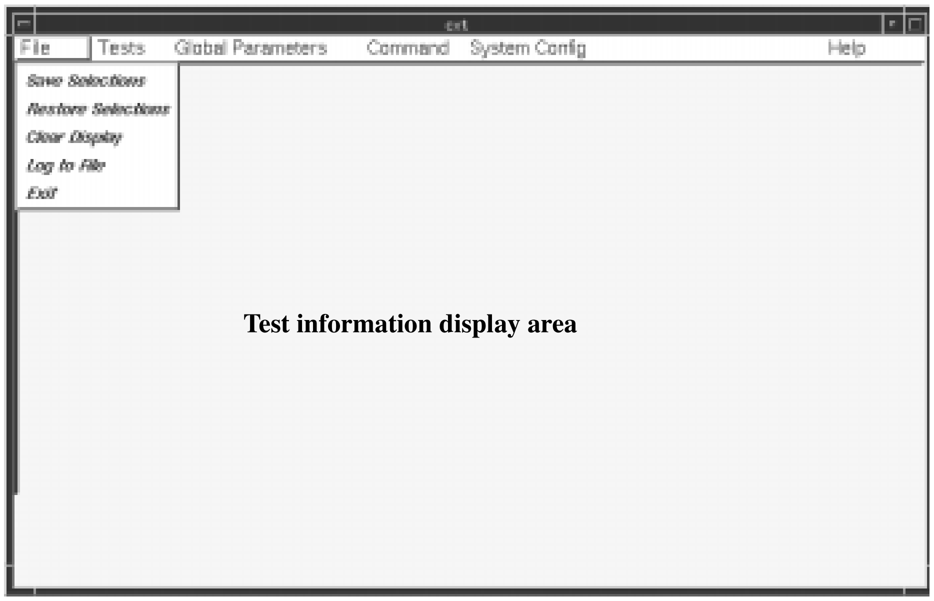
The window has two areas of importance:

- Menu selections
- Test information display

Menus

There are six main menus in `cxtest`. Figure 42 shows the cxtest menu bar.

Figure 42 **`cxtest` menu**



File menu

The File menu has the following options:

- Save Selections
- Restore Selections
- Log to File/Close Log File
- Clear Log
- Exit

Save Selections

The Save Selections option saves specific tests or configurations.

Restore Selections

With the Restore Selections option, the user runs specific tests without having to click on many buttons.

Clear Display

This option clears the browser of all text. It does not clear the log file.

Log to File/Close Log File

This option starts logging the information to the file /spp/data/<COMPLEX_NAME>/cxtest.log. The previous file is not saved. No information present on the screen prior to this option being enabled is saved.

Exit

The Exit option closes `cxtest`.

When exiting `cxtest`, the state of the Boot option is set to what is displayed in the System Configuration menu. The default is to return to OBP, so if the user intends to return to `cxtest`, make sure the test controller stand-alone option is checked. See Figure 45.

Test menu

Selecting a test from the Test menu opens a window like the one in Figure 43. The Test menu varies depending on the tests loaded in EEPROM. If there is only one test loaded in EEPROM, then only one test appears in this window. The test names are presented as they appear in the EEPROM. If a test is not present in EEPROM, there is a "-" in the menu.

The selections presented are based on whether the Test Controller has built a Subtest table and Class table in its `tc_test_info_struct` structure.

Class menus

Selecting a test opens a window that displays all classes for the test. See Figure 43. Down the left hand side of the window are a column of round buttons, and down the right hand side of the window are two columns of buttons.

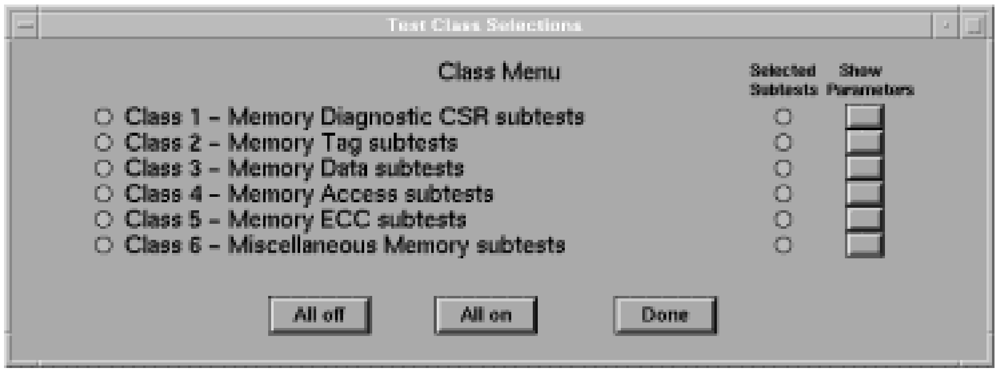
Subtest menus

The left column of buttons allows the user to select all the subtests in the class with a single click. The button will turn yellow when selected.

The round buttons on the right hand side select subtests (opening another window for theses choices) within that class. If only some of the subtests in a class are selected, then both the round buttons for that class turn yellow.

Consider the round button on the left as an indication that tests within the class have been selected and the round button on the right as an indication that only some of the tests have been selected, as opposed to all the tests of that class.

Figure 43 **Test Class Selection menu**


Test Class Selections
Class Menu
Selected Subtests
Show Parameters
Class 1 – Memory Diagnostic CSR subtests
Class 2 – Memory Tag subtests
Class 3 – Memory Data subtests
Class 4 – Memory Access subtests
Class 5 – Memory ECC subtests
Class 6 – Miscellaneous Memory subtests
All off
All on
Done

The last button on the right hand side associates parameters with the test class. Clicking this button opens another window with the parameters for that class. If there are more parameters than will fit on the screen, a scroll bar allows the user to scroll the list.

Also above and below each of the scroll bars are two buttons that allow the user to page up or down through the parameter list.

The Defaults button installs test default values into all the parameters.

If a class of tests has no parameters associated with it, the right most button (the square one) is not shown.

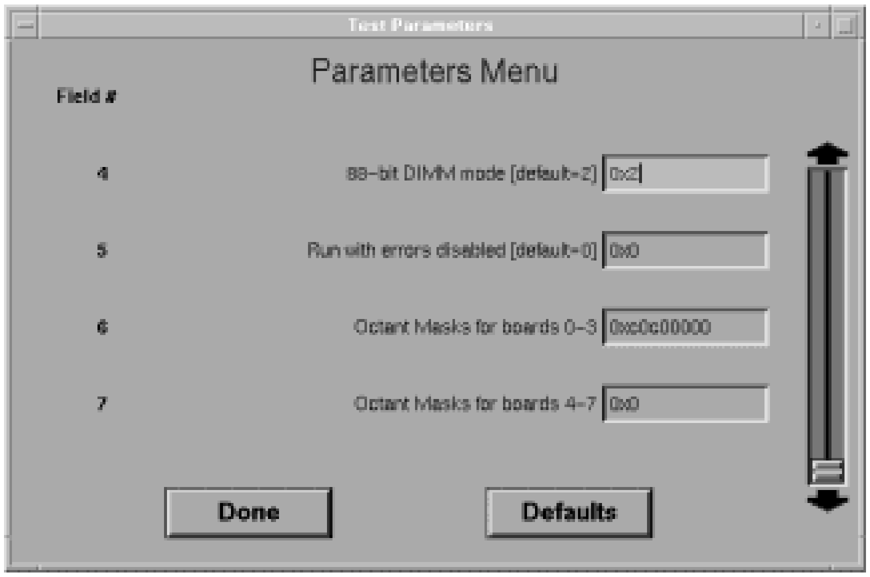
Global Test Parameters menu

`cxtest` provides the ability to loop on a number of tests by setting the Loop Enable count. The looping parameter is applied on a per test basis and is applied to all the tests. As an example it the user selects two subtests from the cpu test and one class from the memory test and sets the loop parameter to 3, then the two subtests from the cpu test are repeated three times in. Then the class of tests from the memory test are repeated three times.

A number of different pausing options are selectable. The pause can be selected before a class, after a class, before a subtest and after a subtest. Setting any of these options will cause them to be in effect for all the selected tests.

Clicking this menu option opens the window shown in Figure 44.

Figure 44 **`cxtest` Global Test Parameters menu**



Field #		
4	88-bit DIMM mode [default=2]	0x2
5	Run with errors disabled [default=0]	0x0
6	Octant Masks for boards 0-3	0xc0c00000
7	Octant Masks for boards 4-7	0x0

Command menu

The Command menu is used to perform actions on the node or complex being tested. These actions include:

- Go
- Reset Machine
- Read Boot Config Map

The Go selection starts the subtests. The subtests are sent to the test controller one at a time so that the application can detect the completion of each subtest. While running, an Abort button appears at the bottom of the screen. Clicking on this while the test is in progress terminates all test selections, not just the subtest currently running.

The Reset Machine option resets the system.

The Reading Boot Configuration Map must be used if the physical location of the boards being tested changes. It allows the user to keep `cxtest` running while the node is powered off and boards are moved.

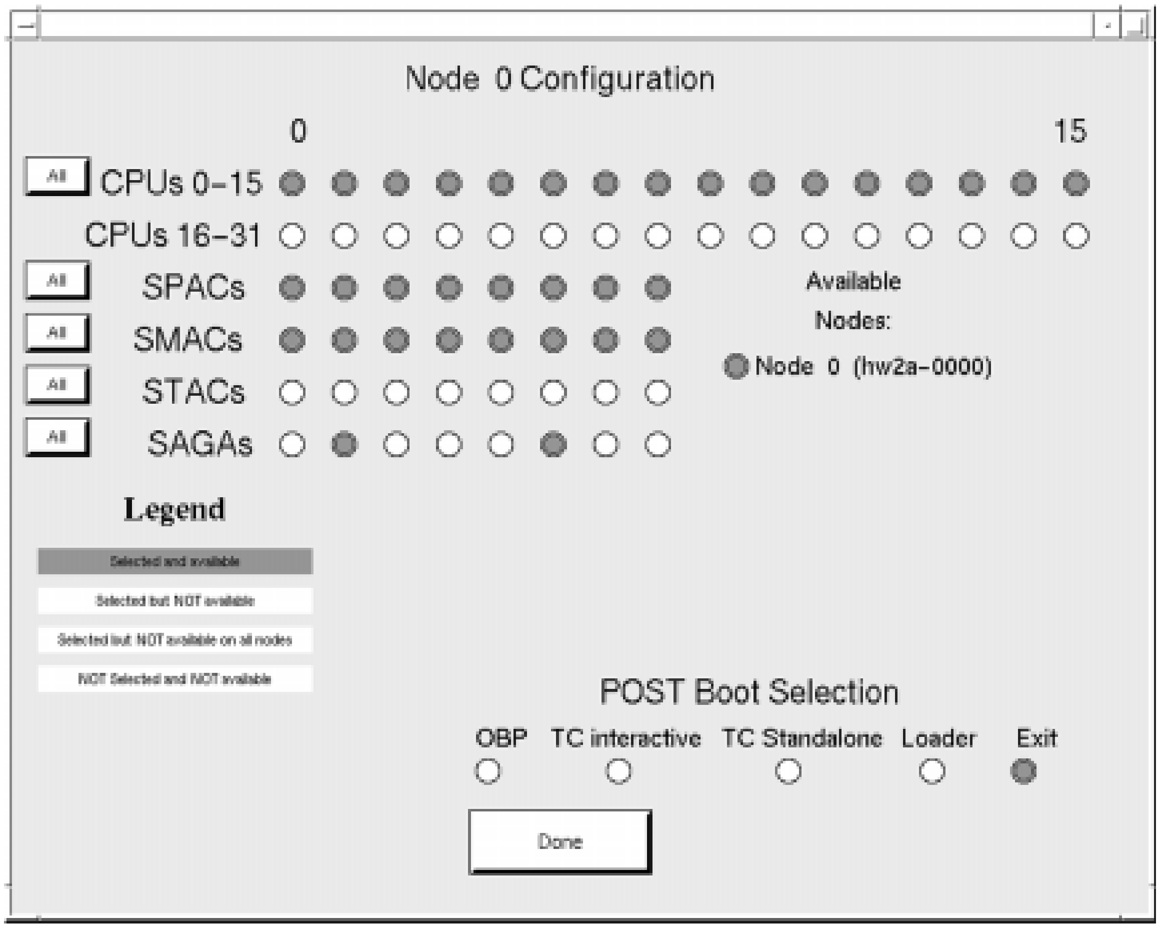
The Abort button will disappear while control is returned.

System Configuration menu

The System Configuration menu displays all nodes that were on-line at the time `cxtest` started. Clicking one of the menu entries opens a node configuration window (Node x Configuration window) that allows the user to select the hardware to test, excluding I/O-specific devices such as disk drives or PCI adapters. See Figure 45.

Any hardware selections made on this screen apply to all tests to be run. For example, to run test A with hardware configuration A and then test B with hardware configuration B, the configuration must be changed manually after test A is completed. The information contained in the node configuration window is extracted from the boot configuration map. If this map changes, it can be reread using the Command menu.

Figure 45 System configuration window



Help menu

The Help menu has two entries: About and Contents. The About selection displays the version number of `cxtest` running and the Contents selection opens a browser that can scroll through the help file.

Display area

The display area shows the output of the tests. This output consists of messages that indicate when the tests start, the amount of time that the test has be running, and any error information. The user can not cut and paste from this area. To record what transpires in test session, use the Log to File option.

When a test is started, a large ABORT button appears. Use this to escape out of a long test sequence or to stop a test not configured correctly. After using the ABORT button, reset the node from the Command menu. Once the abort command has been sent to the test controller, the button will disappear. The abort button also disappears when the test completes successfully.

Powering down the system

When using `cxtest` in a troubleshooting environment, it is not necessary to exit and enter `cxtest` each time the power is cycled.

To remove power to the system (for example, to move a board), power the system down leaving `cxtest` running. Make sure that no tests are actively running.

Once power is restored, POST returns control to the test controller in the stand-alone mode. The user must also wait for the `ccmd` routine to regenerate the database. A message will appear on the SSP console stating the database generation is complete.

After the database is regenerated, the Boot Configuration map must be read using the Command menu. If this is not done, `cxtest` will not have the correct hardware configuration information.

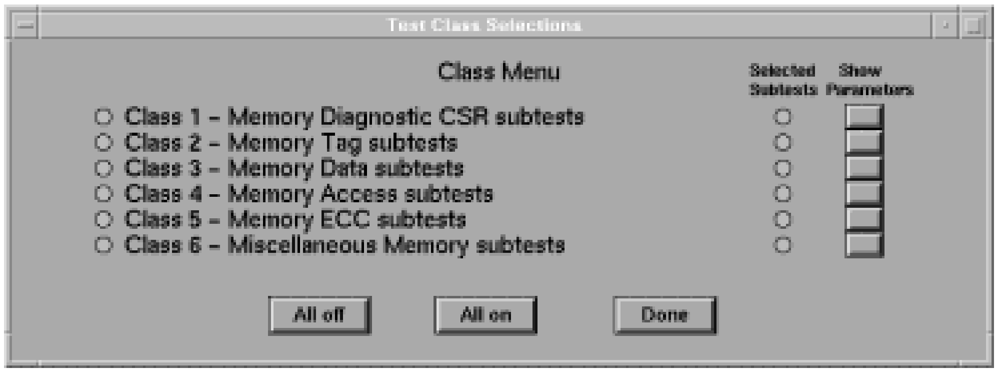
Example of running diagnostics from `cxtest` window

The following example procedure shows the user how to use `mem3000` from `cxtest`. It assumes that the node configuration has been set up using the main `cxtest` window.

Step 1. From the `cxtest` main menu Tests option, select MEM3000 - EEPROM based memory tests.

This opens the class selection window shown in Figure 46.

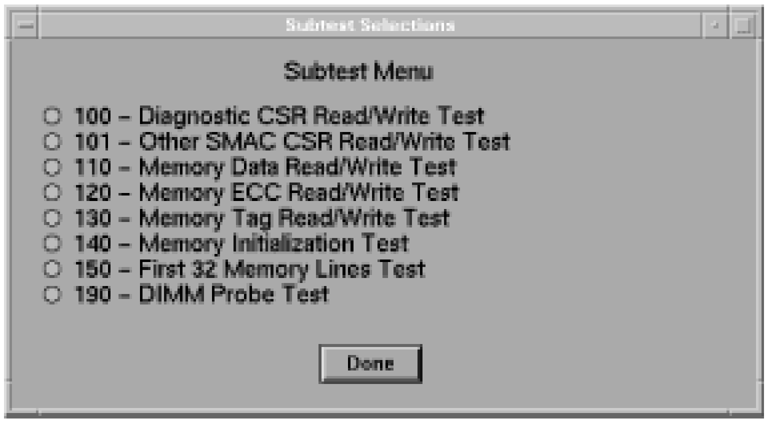
Figure 46 **`mem3000` Test Class Selection window**


Test Class Selections
Class Menu
Selected Subtests
Show Parameters
Class 1 – Memory Diagnostic CSR subtests
Class 2 – Memory Tag subtests
Class 3 – Memory Data subtests
Class 4 – Memory Access subtests
Class 5 – Memory ECC subtests
Class 6 – Miscellaneous Memory subtests
All off
All on
Done

Step 2. In the Test Class Selections window, click on the round buttons to the left of the classes to select which class of test to execute. Any combination of classes may be selected.

Step 3. Click on a Selected Subtests button to select which associated subtests to run for each class. A Subtest Selections window opens for each Selected Subtests button clicked. The Class 1 Subtest Selections window is shown in Figure 47.

Figure 47 **`mem3000` Class 1 Subtest Selections window**



Subtest Selections

Subtest Menu

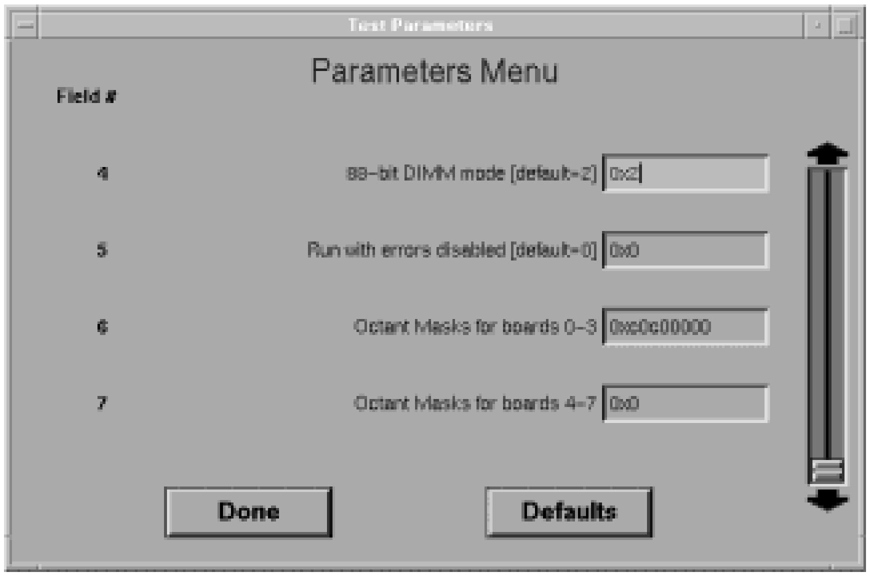
- ○ 100 – Diagnostic CSR Read/Write Test
- ○ 101 – Other SMAC CSR Read/Write Test
- ○ 110 – Memory Data Read/Write Test
- ○ 120 – Memory ECC Read/Write Test
- ○ 130 – Memory Tag Read/Write Test
- ○ 140 – Memory Initialization Test
- ○ 150 – First 32 Memory Lines Test
- ○ 190 – DIMM Probe Test

Done

Step 4. In the Subtest Selections window for each class, click the button for subtest to be executed. Any combination of subtests may be executed.

Step 5. To set the parameters for each class of test, click the appropriate Show Parameters button in Test Class Selections window. This opens the Class Parameters window. Figure 48 shows the `mem3000` Class 1 Test Parameters window.

Figure 48 **`mem3000` Test Parameters window**



Test Parameters

Parameters Menu

Field #		
4	88-bit DIMM mode [default=2]	0x2
5	Run with errors disabled [default=0]	0x0
6	Octant Masks for boards 0–3	0xc0c00000
7	Octant Masks for boards 4–7	0x0

Done Defaults

Step 6. To start the selected tests and subtests, click the Go option in the Command menu in the `cxtest` main window.

Step 7. View the results in the lower window pane of the `cxtest` main window.

Command line interface

`cxtest` is a utility that allows the user to run tests loaded into the Test Controller. Tests can be specified on the command line or a Graphic User Interface can be started to simplify test selection. `cxtest` allows use of the Test Controller without being at the system console.

The advantage of using the command line over the graphics user interface is that there is a little greater control over the order tests are run and the ability to run the tests from a script. The disadvantage is that the user must be more aware of what tests are installed and how to run them.

NOTE The -d option must be used on the command line to start the GUI interface to `cxtest`.

By default, `cxtest` tries to load the test information needed from a file. The name of the file is cxtest.load located in the same directory as the test. A different file can be specified by using the `-f` option on the command line.

Command line options

When using the command line interface to `cxtest`, the command line should be built around the following example model.

`cxtest` command line model

```
cxtest [loading options] <test> [parameters]
[looping/pausing/control] [class/subtest selections] <test>[parameters]
[looping/pausing/control] [class/subtest
selections]...etc.
```

Table 22 **Command line loading options.**

Loading options	Description
`-f <load_file_name>`	Use the <load_file_name> to read what tests are to be loaded in to `cxtest`.
`default`	If no load options are specified, the file /spp/data/cxtest.load is read to know what tests to load.

Command line test selections

The command line interface deciphers the following switches to select tests.

- `-mem`—Memory diagnostic.
- `-io`—I/O diagnostic.
- `-cpu`—processor diagnostic.
- `-eri`—multinode diagnostic

All the arguments between two test selections apply only to first test specified as in the following example:

Example `cxtest` command line

```
cxtest -mem -lt 3 -c 4 -io -c 2
```

The looping specification only applies to the memory test which runs the class-4 tests three times. The I/O test run the class-2 test only once.

Command line looping and pausing

A number of different pausing options are available. Tests can be paused at the beginning of a subtest, end of a subtest, beginning of a class and at the end of a class. Table 23 shows `cxtest` looping and pausing options.

Table 23 **Looping, pause, and control options**

Looping and Pause Controls	Description
`-pe <ON-OFF>`	Pause at end of subtest
`-pb <ON-OFF>`	Pause at beginning of subtest
`-ps <ON-OFF>`	Pause at beginning of class
`-pt <ON-OFF>`	Pause at end of class
`-ls <number>`	Execute <number> of loops of the test that follows
`-lt <number>`	Execute <number> of loops of all the tests that follow
`-t <number>`	Switches the test controller to run on processor <number>. (range 0-15)
`-mt <number>`	Allows specification of error count

To set the number of times a test is looped on use the `-lt <number>` option.

Example of `cxtest -lt` option

```
cxtest -mem -lt 3 -c 4 -io -c 2
```

The looping specification only applies to the memory test which runs the class-4 tests three times. The I/O test run the class-2 test only once.

Command line error counts

The error count allows the test to proceed after an error has occurred. The error count must be set for the test to run after a failure. To set the error count use the `-mt <number>` option.

Command line class Selections

To select an entire class of subtests, use the `-c <number>` option. The user can specify a range of classes by using a hyphen between the numbers.

As an example, **`-c 2-4`**, runs classes 2, 3, and 4.

To specify a list of classes, place a comma between the numbers. An example would be **`-c 5,7,2`**. This runs class 5 then class 7 and finally class 2.

Command line subtest selections

To select a subtest use the `-s <number>` option. To specify a range of subtests, use a hyphen between the numbers. As an example, **`-s 100-150`**, runs subtests 100 through 150.

To specify a list of subtests. place a comma between the numbers. As an example, **`-s 100,150,140`**, runs subtest 100, then subtest 150, and finally subtest 140.

Command line parameter specifications

To specify the value of a parameter for a test, use the `-pa# <val>` option. These options must be placed before the tests that uses them on the command line as in the following example:

Example of `cxtest -pa` option

```
cxtest -mem -c 1 -pa4 4 -pa5 2 -c 2
```

This runs class one, changes the value of the parameters 4 and 5, and then run class two. The parameters only have effect for the test specified. That is, if the memory test had been followed by `-io -c 1`, the value of parameters 4 and 5 would be the defaults for the I/O tests.

There are 128 parameters in all, `-pa0` through `-pa127`.

The user can also specify node-specific parameters from the command line. The syntax is:

```
-pa#_n# <val>
```

where the first # is the parameter number and the second # is the node number to which the parameter is applied.

As an example,

```
cxtest -mem -pa6_n0 8 -pa10_n2 16 -c 3
```

would set parameter 6 on node 0 to 8 and parameter 10 on node 2 to 16 and then run class 3. As with the “global” test parameters, the node-specific parameters apply only for the test specified and must be specified in the command line before the tests which will use them.

The underscore (_) separating the parameter number from the node specification can be replaced with a colon (:). The command shown below is equivalent to the prior example.

```
cxtest -mem -pa6:n0 8 -pa10:n2 16 -c 3
```

Changing test controller

The `-t <proc_num>` option changes the processor that is running the test controller. This parameter must be used before the test selections (i.e. `-c xx` or `-s yy`). There is a 10-second delay to invoke the change. `<proc_num>` must be a valid processor (0-31), and it must be present and available in the system.

Test output

Test progress and error information is displayed just as in the graphics interface, with the exception that the information is displayed on the terminal the test was started from. There is no logging to a file with this interface, so the invocation of `cxtest` should capture standard out and standard error into a file if the log is desired.

6 Processor-dependent code firmware loader

The processor-dependent code firmware loader (`pdcfl`) is a firmware module that loads other firmware modules into FLASH. It is intended to speed up download of POST and OBP on newly manufactured or malfunctioning utility boards. If the target system can successfully boot OBP, OBP should be used to download firmware in favor of `pdcfl`. `pdcfl` can be loaded into FLASH using `load_eprom` as a stand-alone, portable module.

pdcfl loading, booting, and setup

NOTE This step should not be necessary under normal circumstances.

`pdcfl` is loaded on all Utilities boards at the factory. If the Utilities board FLASH contents have been erased, pdcfl may be loaded into the Utilities board using `load_eprom`. `load_eprom` supports a `-f` option for loading `pdcfl` to the appropriate sector in FLASH memory.

As an example:

```
load_eprom -n <node IP number|node IP name> -f /spp/firmware/pdcfl.fw
```

Once `pdcfl` has been loaded, it can be started by issuing a `do_reset` with a loader option:

Example of `do_reset` with loader option

```
do_reset <node id> loader
```

NVRAM setup

NOTE This step should not be necessary under normal circumstances.

If the NVRAM contents have been corrupted, there are two parameters that must be initialized: `ts_ip` and `scub_ip`. The usual values are:

```
ts_ip      15.99.111.99
scub_ip    15.99.111.116
```

`scub_ip` may vary based on the number of nodes connected to the SSP. Use `ts_config` to initialize the `scub_ip`. If `ts_ip` does not match the SSP IP, use `pdcfl setenv`.

SSP setup

When installing SSP software, the install scripts automatically set up the Service Support Processor to support `pdcfl`.

NOTE If `pdcfl` is unable to access firmware files on the SSP, correct the SSP configuration.

The SSP needs to be setup to act as a `tftp` server for loading the desired files into FLASH memory.

This requires making these entries to the following files:

To /etc/services make the following entry:

```
tftp        69/udp          Trivial File Transfer Protocol
```

To /etc/inetd.conf make the following entry:

```
tftp       dgram  udp wait   root /usr/lbin/tftpd    tftpd -T 90
```

Also send an HUP to inetd.

To /etc/passwd make the following entry:

```
tftp:*:510:20::/spp/firmware:/usr/bin/false
```

Files for loading to FLASH can then be placed in the /spp/firmware directory.

To /var/adm/inetd.sec, make the following entry:

```
tftp allow 15.99.111.*
```

NOTE All of the above modifications are automatically performed by `ts_install` when the SSP software is installed.

pdcfl commands

In a multinode complex, all nodes are synchronized and controlled through the console of the lowest numbered node (usually node 0). The node that receives the commands from the console is indicated by the first digit in the command prompt.

From the `pdcfl` prompt, the following commands are supported:

- `node <number>`—Switches console control to the node indicated.
- `printenv <variable>`—Prints configuration variables from NVRAM.
- `setenv <variable value>`—Allows setting configuration variables in NVRAM.
- `lifls`—Prints a listing of the LIF volume in the FLASH EEPROMs. The listing includes the name of the module, the FLASH address at which the module starts, the size in LIF units, the date the module was last written, and the sectors included by the module.

An example of the `lifls` command

```
PDCFL> lifls

LIF Volume FLASH4

Name         Addr           Size    Date           Sectors
----------------------------------------------------------
POST         0xF0020000     0x400   04/09/97       4-5
TC           0xF0140000     0x300   04/09/97       16-17
CPU3000      0xF0170000     0x300   04/09/97       17-18
DIODC        0xF01A0000     0x300   04/09/97       19-20
MEM3000      0xF01D0000     0x2F0   04/09/97       20-21
RDR_DUMPE    0xF01FF000     0x10    04/09/97       21
IO3000       0xF0260000     0x500   04/09/97       25-27
ERI3000      0xF02B0000     0x300   04/09/97       27-28
PDCFL        0xF02E0000     0x200   04/09/97       29-30
```

- `ver`—Prints the listing of the version strings for modules loaded into flash. The `ver` command can not detect OBP, `PDC_entry`, and `spp_pdc` version strings.
- `fload <file> <location>`—Loads a `file` from the SSP `tftp` directory to the address in FLASH specified in the LIF directory by name. `location` can also be a specific address given in hex to allow loading files that have not yet been entered in the LIF directory. If this form is used, the LIF directory will not be updated.

An example of the `fload` command

```
[0:PB4L_A] PDCFL> fload mem3000.fw mem3000
TFTP server     : 15.99.111.99
CUB IP          : 15.99.111.166
Reading         : mem3000.fw
Writing         : mem3000                    (each '.' represents 4K copied)


Reading sector 0xF01C0000
Writing sector:    ................................
Reading sector 0xF01E0000
Writing sector:    ................................
```

- `reset [post|OBP]`—Resets the node, optionally changing the boot vector to point to the POST module.
- `bcast <command>`—Broadcasts the `command` to all nodes in a complex. The node that has control executes the `command` first, then all other nodes execute it one at a time.
- `nodemap`—Allows configuration of the parameters used in multinode initialization

7 cpu3000

`cpu3000` runs via the test controller and provides a basic test of the functionality of the processor. It requires a minimum of one processor with its associated SPAC and two EWMBs. Included in the testing are most of the instruction set, the ALU, general/space/control registers, external interrupts, the Diagnostic registers, TLB, the instruction cache, the data cache, and the floating point unit. The tests are grouped together in five classes beginning with verification of the most basic functionality and progressing toward more complex functionality. Each class has a set of subtests that target specific functionality.

cpu3000 classes and subtests

`cpu3000` consists of a series of tests grouped together in classes beginning with verification of the most basic functionality and progressing toward more complex functionality. Each class has subtests which target specific functionality.

When a failure is encountered, the chassis code is available through the test controller along with the progress value.

cpu3000 classes

`cpu3000` has five classes of tests shown in Table 24.

Table 24 **Classes of `cpu3000` tests**

Class	Name
1	Basic CPU tests
2	Instruction cache RAM test
3	Data cache RAM tests
4	TLB RAM tests
5	Functional tests requiring main memory

cpu3000 subtests

The `cpu3000` subtests are listed in Table 25 through Table 28.

Table 25 **cpu3000** Class 1 subtests

Subtest	Name	Description
100	Processor basic	Verifies the majority of registers and a basic set of instructions. Chassis code: 0x41020.
101	Processor-ALU	Verifies the processor and arithmetic Logic unit (ALU) functionality. Chassis code: 0x41021.
102	Processor branch	Verifies the branch instructions. Chassis code: 0x41022.
103	Processor-arithmetic condition	Verifies the arithmetic conditions of the unit, extract/deposit and carry/ borrow instructions.Chassis code: 0x41023
104	Processor bit operations	Verifies the processor's bit operations. Chassis code: 0x41024
105	Space and control registers	Verifies the space and control registers. Chassis code: 0x41025.
110	External interrupts	Executes sixty three external interrupts, one for each EIR_VAL position excluding Itimer. Chassis code: 0x41026.
111	Interval timer	Verifies the interval timer trap. its masking capability, associated control process, and timer rollover. Chassis code: 0x41027
120	Multimedia	Verifies the functional operation of the multimedia units. Chassis code: 0x41028.
130	Shadow	Verifies the shadow registers. Chassis code: 0x41029.

Subtest	Name	Description
140	Diagnostic register	Verifies the local Diagnose Registers. Chassis code: 0x4102a.
141	Remote diagnostics registers	Verifies the remote Diagnose Registers. Chassis code: 0x4102b.
150	Register bypass	Verifies the register bypass functionality of the processor. It tests three different types of bypassing that can occur between the two integer queues. Chassis code: 0x4102c.

Table 26 **cpu3000** Class 2 subtests

Subtest	Name	Description
210	Icache RAM	This routine pattern tests the icache ram. Chassis code: 0x42020.

Table 27 **cpu3000** Class 3 subtests

Subtest	Name	Description
310	Dcache RAM	Verifies the data cache rams. Chassis code: 0x42070.

Table 28 **cpu3000** Class 4 subtests

Subtest	Name	Description
400	TLB RAM	Verifies the TLB ram arrays with a pseudo random pattern. Chassis code: 0x410b1.

Table 29 **cpu3000** Class 5 subtests

Subtest	Name	Description
500	Late-early self test (LST-EST)	Runs subtests 100, 101, 102, 103, 104, 105, 120, 130, and 150, first in main memory and then in the Icache. This test has the following chassis codes: LST test - 0x44020 processor basic - 0x44021 processor ALU- 0x44022 processor branch - 0x44023 processor arithmetic condition - 0x44024 processor bit ops - 0x44025, space and control registers - 0x44026 multimedia - 0x44029 shadow - 0x4402a register bypass - 0x4402d.
510	Cache-byte	Verifies the instructions that store bytes, halfwords, and words.Chassis code: 0x44030.
520	Cache flush	Verifies the instructions that flush the Icache and Dcache. Chassis code: 0x44040.
530	Icache miss	Verifies that instructions can be encached from coherent memory. 0x44050.
540	Dcache miss	Verifies that data can be encached from coherent memory. Chassis code: 0x44060.
560	TLB transfer	Verifies TLB hits and misses, as well as access rights and protection ID validation. Chassis code: 0x410b2.

Subtest	Name	Description
570	Floating point unit	Verifies the floating point unit. It consists of several groups of tests that include testing of the FPU registers, instruction tests, trap handling, and access rights and ID validation. This test has the following chassis codes: FPU functionality - 0x410a0 FPU registers - 0x410a1 FPU instruction - 0x410a2 FPU traps - 0x410a3 FPU miscellaneous tests- 0x410a4 FPU bypass - 0x410a5.

cpu3000 errors

When a failure occurs, the chassis code and the progress value are available through the test controller. The progress value indicates what portion of the subtest encountered the error. To match chassis codes on other platforms, the three most significant bits of the chassis code are modified to indicate a fault detected by the test. Therefore, a failure in the FPU Traps test, chassis code 0x410a3, for example, would be reported as a chassis code of 0x210a3.

8 eri3000

`eri3000` verifies the functionality of the V2500/V2600 multinode and SCA systems. `eri3000` validates the following functionality:

- METABUS—Connection between SMAC and STAC
- STAC registers
- ERI Rings—Connections between STACs of different nodes
- STAC ERI ring state machine
- Time of Century Synchronization—Signal sent from an SPAC and propagated along a specific wire of the ERI rings
- Remote Memory Access
- CTI Cache

`eri3000` is executed by the test controller. Class 1 tests require one node with a minimum of one processor and its associated SPAC and two memory boards with associated SMACs and STACs. Classes 2 through 4 require at least two nodes with a minimum of one processor and its associated SPAC and two memory boards with associated SMACs and STACs each. These nodes must be connected by ERI cables and properly configured using the multinode configuration rules. Also, each node must be connected on the diagnostics LAN. A provided test verifies diagnostic LAN functionality and test synchronization capability.

eri3000 classes and subtests

The `eri3000` tests are grouped together in classes beginning with verification of the most basic functionality and progressing toward more complex functionality. Each class is divided into subtests which target specific functionality

When a failure occurs, the chassis code is available through the test controller along with the progress value.

eri3000 classes

`eri3000` has three classes of tests shown in Table 30.

Table 30 Classes of **`eri3000`** tests

Class	Name
1	Verifies Local STAC register Access and Metabus Connections
2	Verifies ERI Ring Initialization and Configuration
3	Verifies STAC-STAC ERI Data Path

eri3000 subtests

The `eri3000` subtests are listed in Table 31 through.

Table 31 **`eri3000` Class 1 subtests**

Subtest	Name	Description
100	CSR Read/Write	Verifies the ability to read/write the Ring Configuration, Chip Configuration, and error registers in the STAC.
110	Metabus Walking Ones	Uses the Error Address register to perform a walking ones pattern test of the metabus.
120	Metabus Walking Zeros	Uses the Error Address register to perform a walking zeros pattern test of the metabus.
130	Metabus SMAC-to-STAC Pattern	Uses the Error Address register to perform data pattern tests that check for pin-to-pin shorts at both the SMAC and the STAC for the MT (SMAC-to-STAC) Metabus path.
140	Metabus STAC-to-SMAC Pattern	Uses the Error Address register to perform data pattern tests that check for pin-to-pin shorts at both the SMAC and the STAC for the TM (STAC-to-SMAC) Metabus path.
150	Metabus User Data Pattern	Uses the Error Address register to perform data pattern test for the Metabus using user-specified patterns.

Table 32 **`eri3000`** Class 2 subtests

Subtest	Name	Description
200	LAN Synchronization	Verifies various synchronization messages used to coordinate tests across nodes.
201	Ring DEAD State	Verifies that all STACs in the node can be put into the DEAD state.
210	Ring RESET State	Verifies that all STACS in the node can be put into the RESET state.
220	Ring INIT State	Verifies that all STACs in all nodes in the complex can get to the INIT state. This requires node-to-node communications. Each node, in turn, performs the following: Set all local rings into RESET Tell all other nodes to go to RESET Release RESET Check local node for INIT
230	Ring RUN State	Verifies that all STACs in all nodes in the complex can get to the RUN state. This requires node-to-node communications. Each node, in turn, performs the following: Set all local rings into RESET Tell all other nodes to go to RESET Release RESET Tell all other nodes to release RESET Check local node for RUN

Subtest	Name	Description
240	Ring CLEAR State	Verifies that all STACs in all nodes in the complex can get to the CLEAR state.
250	Ring WAIT State	Verifies that all STACs in all nodes in the complex can get to the WAIT state. This requires node-to-node communications. Each node, in turn, performs the following: Set all local rings into CLEAR Tell all other nodes to go to CLEAR Release CLEAR Check local node for WAIT
260	Ring Initialization	Verifies that all STACs in all nodes can negotiate from reset state to run state. Each node, in turn, performs the following: Goto RESET and sync Deassert RESET Poll until run state achieved Verify run state and sync Verify scrubber or distance from scrubber field

Table 33 **`eri3000`** **Class 3 subtests**

Subtest	Name	Description
300	Remote CSR Read/Write	Verifies that remote STAC and SPAC registers can be accessed from each node to all other nodes in the complex.
310	ERI Y-ring Walking Ones	Performs a walking ones pattern test on the ERI from each node to the next connected node, using the STAC Error Address register as the data repository.
320	ERI Y-ring Walking Zeros	Performs a walking zeros pattern test on the ERI from each node to the next connected node, using the STAC Error Address register as the data repository.
330	ERI Y-ring User Data Pattern	Performs a data pattern test on the ERI from each node to the next connected node. Each node uses a local user-specified data value. The STAC Error Address register is used as the data repository.
340	ERI X-ring Walking Ones	Performs a walking ones pattern test on the ERI from each node to the next connected node, using the STAC Error Address register as the data repository.
350	ERI X-ring Walking Zeros	Performs a walking zeros pattern test on the ERI from each node to the next connected node, using the STAC Error Address register as the data repository.
360	ERI X-ring User Data Pattern	Performs a data pattern test on the ERI from each node to the next connected node. Each node uses a local user-specified data value. The STAC Error Address register is used as the data repository.

Table 34 `eri3000` Class 4 subtests

Subtest	Name	Description
400	Scrubber Determination Test	Verifies that the scrubber was determined correctly for each ring, and that all other nodes determined their length from the scrubber.
410	Time of Century Synchronization Test	Verifies the time TOC synchronization logic for all TACs. Each node performs the following: turns off all SPACs TOC, sets up all SPACs for 1 msec, turns on master SPAC (now all SPACs are engaged), polls until desired TOC value achieved, checks all SPACs for TOC, turns off all non-master SPACs, turn off master SPAC
420	Remote Memory Store/Flush Test	Verifies each node's ability to perform a store, then flush, into each other node's memory.
430	Network Cache Rollout Test	Verifies the network cache will rollout a line when another line is accessed that would use the same cache line.

User parameter definitions

The Test Controller environment allows eri3000 to have 20 user parameters. eri3000 has defined these parameters:

Table 35 User parameter definitions

Words	Name	Definition
0/1	64-bit user pattern #0	Used in subtests 330 and 360 (defaults=0x3c3c3c3c/0x3c3c3c3c)
2/3	64-bit user pattern #1	Used in subtests 330 and 360 (defaults=0xc3c3c3c3/0xc3c3c3c3)
4	node mask override	Specifies which nodes to use in testing. (defaults to use all nodes that synchronized at reset)

eri3000 error messages

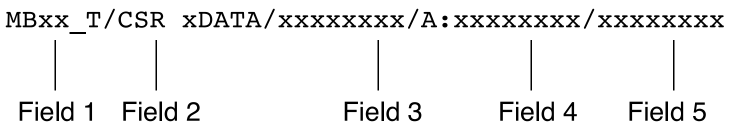
When a failure occurs an event code is sent along with an error message. The least significant 12 bits of the event code contain the error code.

Error messages can occur in the following formats.

Type 1—CSR mismatch

Many subtests use the type 1 message format to indicate that the actual value of a CSR is not equal to its expected value.

Figure 49 **Type 1 error message format**



```
MBxx_T/CSR xDATA/xxxxxxxx/A:xxxxxxxx/xxxxxxxx
```

There are five fields separated by / symbols. The meaning of each field is as follows:

- Field 1—Specifies the STAC on which the failure was detected
- Field 2—Specifies CSR address where mismatch was found
- Field 3—Specifies the value U for errors found in the upper portion (bits 0:31) and L for errors found in the lower portion (bits 32:63)
- Field 4—Specifies the actual 32-bits of data (A: corresponds to "actual")
- Field 5—Specifies the expected 32-bits of data

Type 2—ERI mismatch

The class 4 subtesst use this format to indicate a failure in remote access. For errors in the upper portion (bits 0-31):

Figure 50 **Type 2 error message format**

```
  x/SN:x/DN:x/@xx xxxxxxx/A:xxxxxxxx/xxxxxxxx
  |    |    |        |          |       |
Field 1 Field 2 Field 3   Field 4   Field 5  Field 6
```

There are six fields separated by / symbols. The meaning of each field is as follows:

- Field 1—Specifies the value U for errors found in the upper portion (bits 0:31) and L for errors found in the lower portion (bits 32:63)
- Field 2—Specifies source node (originator of failing transaction)
- Field 3—Specifies destination node (node on which error was found)
- Field 4—Specifies 40-bit physical address
- Field 5—Specifies the actual 32-bits of data (A: is reminder)
- Field 6—Specifies the expected 32-bits of data

Type 3—ERI ring state error

All subtests use this format to indicate that not all ERI rings were able to achieve a certain state.

Figure 51 **Type 3 error message format**

```
INCORRECT RING STATE/AMxxxx/EMxxxx/ESxxxxx
                      |      |      |
                   Field 1 Field 2 Field 3
```

There are three fields separated by / symbols. The meaning of each field is as follows:

- Field 1—Specifies actual ring mask. Bitmask of rings which achieved desired state. Bits 0:7 denote X-rings and 8:15 denote Y-rings. Bits missing in this field indicate failing rings.
- Field 2—Specifies expected ring mask. Bitmask of rings being tested. Bits 0:7 denote X-rings and 8:15 denote Y-rings.
- Field 3—Specifies expected ring state. Bits 0:4 denote X-ring state and 12:15 denote Y-ring state

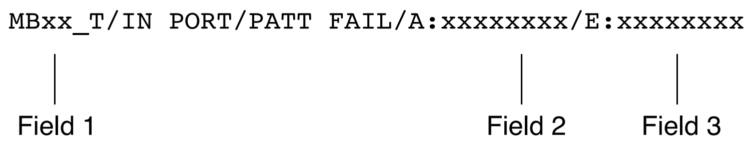
Table 36 **ERI ring states**

Ring state	X-Ring	Y-Ring
DEAD	00000	00000
RESET	10000	00001
INIT	20000	00002
RUN	30000	00003
CLEAR	40000	00004
WAIT	50000	00005

Type 4—ERI cable pattern failure

Class 3 subtests use this format to indicate that the cable connected to the input port of the STAC has failed a pattern test.

Figure 52 **Type 4 error message format**



```
MBxx_T/IN PORT/PATT FAIL/A:xxxxxxxx/E:xxxxxxxx
```

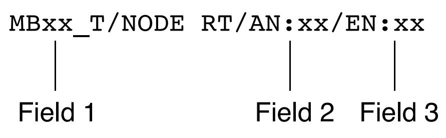
There are three fields separated by / symbols. The meaning of each field is as follows:

- Field 1—Specifies the STAC which has the failing cable connected to its input port.)
- Field 2—Specifies the actual 32-bits of data (A: is reminder)
- Field 3—Specifies the expected 32-bits of data (E: is reminder)

Type 5—ERI node routing failure

Subtest 300 uses this format to indicate that the Node_Id found in the TAC System Configuration CSR in the destination node is not correct. This could mean that the nodes are not properly configured, the cables are not connected correctly, or the Node_Id for a node is wrong. In rare cases, it could mean there is a bad cable at some point on this ring (all cables on the ring are suspect).

Figure 53 **Type 5 error message format**



```
MBxx_T/NODE RT/AN:xx/EN:xx
```

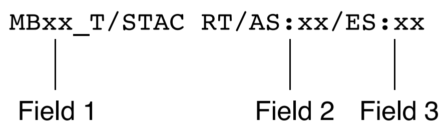
There are three fields separated by / symbols. The meaning of each field is as follows:

- Field 1—Specifies the source TAC.
- Field 2—Specifies the actual 32-bits of data (A: is reminder)
- Field 3—Specifies the expected 32-bits of data (E: is reminder)

Type 6—ERI TAC routing failure

Subtest 300 uses this format to indicate that the STAC_Id found in the STAC Chip Configuration CSR in the destination node is not correct. This could mean that the nodes are not properly configured or the cables are not connected correctly. In rare cases, this could mean there is a bad cable at some point on this ring (all cables on the ring are suspect).

Figure 54 **Type 6 error message format**



```
MBxx_T/STAC RT/AS:xx/ES:xx
```

There are three fields separated by / symbols. The meaning of each field is as follows:

- Field 1—Specifies the TAC which has the failing cable connected to its input port.)
- Field 2—Specifies the actual STAC_Id reported.
- Field 3—Specifies the expected STAC_Id.

Type 7—ERI synchronization failure

All subtests use this format. The diagnostic LAN broadcasts synchronization packets to all nodes. These packets coordinate subtest actions. They start up, perform a barrier-sync, report test status, and report results of specific actions. A sync failure can occur for one of the following reasons.

- Pattern failed because results from different nodes did not match.
- Pattern failed because results from different nodes did not match. One node got an HPMC and could not send the sync packet. In this case, one node sync failure may occur while the failing node halts or continues to run “off in the weeds.”
- SONIC chip or diagnostic LAN cables are malfunctioning.

Synchronization errors only occur if something is definitely wrong. The volume of sync packet traffic is often the first indication that something has gone wrong.

Figure 55 Type 7 error message formats

```
SYNC FAIL/PATT: xxxxxxxx/ANM:xxxx/ENM:xxxx
 SYNC FAIL/PATT:  NONE  /ANM:xxxx/ENM:xxxx
 SYNC FAIL/  AT_xxxxxx  /ANM:xxxx/ENM:xxxx
SF:B4 SEND/PATT:xxxxxxxx/ANM:xxxx/ENM:xxxx
SF:B4 READ/PATT:xxxxxxxx/ANM:xxxx/ENM:xxxx
```

There are three fields separated by / symbols. The meaning of each field is as follows:

- Field 1—Specifies the pattern (PATT:) used in synchronization or the attempted (failing) ring state (AT_).

  The pattern may be the actual pattern being used or (more likely) a cryptic representation of where the test was when the sync failed. Two interesting patterns are `0x1XXX and 0x2XXX`. These syncs involve starting and ending a subtest, respectively. (ex. PATT:00001310 indicates failure to start subtest 310)

The state shown in `AT_xxxxxx` indicates the last ring state the node achieved. The node has broadcasted its state and was waiting for the rest of the nodes to sync at the same state before moving on. The possibilities for this field are AT_DEAD, AT_RESET, AT_INIT, AT_RUN, AT_CLEAR, AT_WAIT.

- Field 2—Specifies actual node mask. This is a bitmask of nodes which correctly responded to the synchronization request. Bits missing in this field indicate failing nodes.
- Field 3—Specifies expected nodes mask. Bit 0 (which denotes node 0) is located at the far right.

Type 8—ERI Time of Century (TOC) Sync failure

Subtest 410 uses this format to indicate a failure in configuring, starting, or maintaining TOC synchronization on all nodes. All values are decimal.

Figure 56 **Type 8 error message formats**

```
TOC  CFG FAIL PAC:x/MASTER N:x/P:x/T:x
TOC SYNC FAIL PAC:x/MASTER N:x/P:x/T:x
                  |      |      |    |
                  |      |      |    |
               Field 1 Field 2 Field 3 Field 4
```

There are four fields separated by / symbols. The meaning of each field is as follows:

- Field 1—Specifies the failing SPAC on this node
- Field 2—Specifies the source node for the TOC signal
- Field 3—Specifies the source SPAC for the TOC signal.
- Field 3—Specifies the source TAC for transmission of the TOC signal across nodes

Type 9—Error(s) Detected by Another Node

This format informs the user that the original error was detected by another node in the complex. When a node detects a failure, it sends a special synchronization packet to inform all other nodes that an error has occurred. At this point, these nodes should exit the subtest gracefully with this as their failure mode. This type of error communication does not occur on class 1 subtests because LAN packets are not verified until subtest 200.

Type 10—ERI System Error

This format informs the user that the system is not properly configured for a certain subtest. Many subtests require that more than one node be present and synchronized for proper execution. This subtest requires a valid multinode system.

Type 11—Default Error Message

This subtest is disabled at this time.

Event codes

The following defines the meaning of the lower 12-bits of the event code. It only indicates the type of error/event that occurred. For more detailed information, the error/event message is needed.

Table 37 **Event codes**

Code	Definition	Error Occurrence
1	INCORRECT_RING_STATE	ERI Ring State Error
2	SYNC_FAILURE	ERI Sync Failure
3	CABLE_PATT_FAIL	ERI Cable Pattern Failure
4	CABLE_NODE_RT_FAIL	ERI Node Routing Failure
5	CABLE_STAC_RT_FAIL	ERI STAC Routing Failure
6	TOC_CFG_ERROR_TYPE	ERI Time of Century Configuration Failure
7	TOC_SYNC_ERROR_TYPE	ERI Time of Century Sync Failure

Code	Definition	Error Occurrence
8	CSR_UDATA_MISMATCH_TYPE	CSR Mismatch (upper)
9	CSR_LDATA_MISMATCH_TYPE	CSR Mismatch (lower)
A	ERI_UDATA_MISMATCH_TYPE	ERI Mismatch (upper)
B	ERI_LDATA_MISMATCH_TYPE	ERI Mismatch (lower)
C	ERI_SYS_ERROR_TYPE	ERI System Error
D	OTHER_NODE_FAILURE_TYPE	Other Node Detected Error

9 io3000

The I/O diagnostic supports Symbios 875 HVD SCSI controllers, Symbios 895 LVD SCSI controllers, and Tachyon Fibre Channel controllers.

`io3000` requires a node with a minimum of one processor, one SIOB with associated SPACs, and two EWMBs with associated SMACs. To exercise peripherals, either a Symbios SCSI or a Tachyon Fibre Channel card is required.

io3000 classes and subtests

`io3000` consists of a series of tests grouped together in classes beginning with verification of the most basic functionality and progressing toward more complex functionality. Each class is broken down into subtests which target specific functionality.

The following sections describe the classes and individual subtests.

io3000 classes

`io3000` has 10 classes of tests shown in Table 38.

Table 38 **Classes of `io3000` test**

Class	Name	Description
1	SAGA CSR Test	Verifies successful writes and reads of SAGA CSRs.
2	SAGA Memory Test	Verifies the functionality of SAGA context/ shared memory and prefetch memory.
5	SCSI Disk Interface Test	Verifies the ability to successfully issue SCSI commands to every selected disk or Fibre Channel target.
6	Channel Mode Test	Verifies the ability to successfully build and use SAGA channels in all the supported modes. Also, every channel can be verified to be usable.
7	DMA Boundary Conditions Test	Verifies that various DMA conditions and every possible interrupt vector work correctly. Also verifies every possible interrupt vector.
8	MultiDisk Concurrency Test	Queues up all selected disks for simultaneous transfers. In this test, all disks operate in a parallel fashion.

Class	Name	Description
11	SAGA SCSI Tape Interface Test	Verifies the ability to successfully issue SCSI commands to every selected tape drive.
12	Symbios Test	Verifies the basic functionality of the Symbios SCSI controller.
15	CDROM SCSI Access Test	Verifies basic SCSI bus access.
16	Tachyon SAGA PCI Access Test	Verifies the SAGA PCI interface to all selected Tachyon controllers
17	Tachlite PCI Loopback	Verifies the PCI interface and simple loopback operations to the selected controllers.
18	Fibre Channel SCSI Mux	Verifies the node's ability to communicate with the FC SCSI Mux

io3000 subtests

The `io3000` subtests are listed in Table 39 through Table 48.

Table 39 **`io3000`** **Class 1 subtests**

Subtest	Name	Description
100	CSR reset	Verifies that each SAGA CSR has a defined state and contains the proper value after the SAGA reset is completed.
105	CSR read/ write	Verify writes and reads for each SAGA CSR using a bitwise March C- test.
110	Error CSR	Verifies that each individual error type in the ErrorCause register can be set and is capable of generating an interrupt.

Table 40 **`io3000`** Class 2 subtests

Subtest	Name	Description
200	Context/ shared memory read/ write	Writes to the first 64-bit location of each context SRAM and reads them to verify that they can be uniquely accessed.
205	Context/ shared memory access width	Verifies that all supported access widths of context SRAM function properly by writing and reading the first 64-bit location.
210	Context/ shared memory march C-	Verifies that coverage for full march C- will increase from approximately 99% to 100% of targeted fault using a bitwise march C- algorithm.
215	Context/ shared memory pattern	Writes and reads random pattern to all of context/shared memory. The random pattern can be modified by changing the random seed option. Also, a user-specified pattern can be used by setting the user pattern options.
220	Context/ shared memory parity detection	Verifies the ability of SAGA to detect parity errors on reads from context/ shared memory. This test uses the FCC bit (force parity error to context SRAM) to write a parity error into context SRAM. The bad parity is read out, causing a parity error to be detected and logged in the SAGAs Error Cause CSR.
225	Prefetch memory read/ write	Verifies that the first 64-bit location of each prefetch SRAM uniquely accessed.
230	Prefetch memory access width	Verifies that all supported access widths of prefetch SRAM function properly by writing and reading the first 64-bit location.

Subtest	Name	Description
235	Prefetch memory march C-	Verify writes and reads to all of prefetch memory using a bitwise march C- algorithm. The default option does a shortened version of the march C- algorithm by using a limited pattern set. The march C- complete enable can be set to do a full march C- test. The test time increases by a factor of approximately four. The fault coverage for full march C- increases from approximately 99% to 100% of targeted faults.
240	Prefetch memory pattern	Writes and reads random pattern to all of prefetch memory. The random pattern can be modified by changing the random seed option. Also, a user-specified pattern can be used by setting the user pattern options.
245	Prefetch memory parity detection	Verifies the ability of SAGA to detect parity errors on reads from prefetch memory. This test uses the FPR bit (force parity error to prefetch SRAM) to write a parity error into context SRAM. Then the bad parity is read out, causing a parity error to be detected and logged in the SAGA's Error Cause CSR.

Table 41 **`io3000` Class 5 subtests**

Subtest	Name	Description
500	SCSI disk test unit ready	A SCSI `test unit ready` command is issued to all selected devices at least twice. This first time, it should return with a SCSI check condition (not reported to the user) since the SCSI bus has been reset. The command is retried after approximately one second. If the second test unit ready fails, an error is reported. The `test unit ready` command does not cause a SCSI data phase to occur.
505	SCSI disk inquiry	A SCSI `inquiry` command is executed on every selected device. This test verifies that the device type field in the inquiry return data is a direct access (disk). A SCSI data in phase will occur.
510	SCSI disk read capacity	A SCSI `read capacity` command is issued to every selected device.
515	SCSI disk read	A SCSI `read` command is issued to every selected device. No data verification is performed.
520	SCSI disk write	A SCSI `write` command is issued to every selected device. No data verification is performed. This test only writes to the disk if the write enable option is turned on. The default is to not allow writes to the device.

Table 42 `io3000` Class 6 subtests

Subtest	Name	Description
600	Channel init	Subtests 600-645 create channels by writing to the SAGA channel builder CSR. The method of channel creation and the specific mode (ATPR setting) is specified in the subtest's one line description. Each test will write data to the disk and read it back and verify it. Each disk's write enable option must be set for the writes and data verification to be allowed. ATPR = 0x0
605	Channel build	ATPR = 0x0
610	Channel init, data prefetch	ATPR = 0x2
615	Channel init, write TLB	ATPR = 0x8
620	Channel build	ATPR = 0x2
625	Channel init, write TLB, data prefetch	ATPR = 0xa
630	Channel init, TLB prefetch	ATPR = 0xc
635	Channel build	ATPR = 0xc

Subtest	Name	Description
640	Channel init, TLB and data prefetch	ATPR = 0xe
645	Channel build	ATPR = 0xe
650	Channel context access	Verifies selected SAGA channels in virtual mode. After each channel is built, the test checks the context SRAM. If the full channel disable option is set, Channels 0, 1007, and power of 2 channels greater than 31 are tested. Otherwise all channels greater than channel number 31 are tested. Channels 1-21 are reserved for controller DMA access. (default).

Table 43 `io3000` Class 7 subtests

Subtest	Name	Description
700	External interrupt	Verifies all possible external interrupt vectors. A separate DMA is executed for each external interrupt vector.
705	DMA across page and channel	Verifies writes and reads of DMAs that cross page and channel boundaries.
710	Jump forward within a page	Verifies writes and reads of DMAs that jump forward within a page.

Subtest	Name	Description
715	Jump backward within a page	Verifies writes and reads of DMAs that jump backward within a page.
720	Jump outside of a page (TLB encached)	Verifies a DMA jump outside of a page. The TLB for the destination page is encached in context SRAM for both writes and reads.
725	Jump outside of a page (TLB not encached)	Verifies a DMA jump outside of a page. The TLB for the destination page is not encached in context SRAM. This means that SAGA must fetch a new TLB before the transfer can continue. This is done for both writes and reads.
730	Jump outside of a channel	Verifies a DMA jump outside of the current channel. This is done for both writes and reads.
735	Non contiguous TLBs	Sets up a translation table for scattered system page mappings (noncontiguous). Then a DMA is set up to use this table. This causes the SAGA to access pages noncontiguously throughout the DMA.

Table 44 **io3000** Class 8 subtests

Subtest	Name	Description
800	Multidisk nonmixed traffic	Issues all selected devices simultaneous SCSI writes and then SCSI reads. The channels are programmed in virtual mode, with data and TLB prefetch turned on.
805	Multidisk mixed traffic	All selected devices transfer data simultaneously. Some devices are performing SCSI reads, while others are performing SCSI writes, thereby causing mixed or bidirectional traffic on the SCSI and PCI busses. The channels are programmed in virtual mode, with data and TLB prefetch turned on. Refetch is turned off. ATPR = 0xe
810	Multidisk mixed traffic	This is the same subtest as 805 but with refetch turned on. ATPR = 0xf

Table 45 **`io3000`** Class 11 subtests

Subtest	Name	Description
1100	SCSI tape test unit ready	Issues a SCSI test unit ready command to all selected devices at least three times. This first time the SCSI bus will have been reset. This is normal. The command is retried after approximately one second. The command is issued again to allow for a check condition due to the medium being changed. Many tape drives require a tape to be installed in the drive, causing the second test unit ready to respond with a medium changed sense status. If the third test unit ready fails, an error is reported. The test unit ready command does not cause a SCSI data phase error to occur.
1105	SCSI tape inquiry	Executes a SCSI inquiry command on every selected device. It verifies the device type field in the inquiry return data to be sequential (tape). A SCSI data in phase error does occur.
1110	SCSI tape rewind	Executes a SCSI rewind command on every selected device and waits for it to complete. The rewind command will not cause a SCSI data phase to occur.
1115	SCSI tape read	A SCSI read command is executed on every selected device. On fixed-block drives, one block is read. On variable-block drives, 252 bytes are read. A SCSI data in phase will occur.

Table 46 **`io3000`** Class 12 subtests

Subtest	Name	Description
1200	Symbios PCI configuration space	Verifies the ability of the SAGA to access the Symbios SCSI controller by way of the PCI configuration space. Verifies the PCI vendor ID and device ID fields to be 0x1000 and 0x000f, respectively. Also verifies the base address registers to be writable and readable.
1205	Symbios SCSI PCI I/O and Memory space	Maps the Symbios SCSI controller through PCI configuration space so that the controller's CSRs may be accessed by way of both PCI I/O and memory space. The test writes a pattern to a scratch register (SCRATCHA) in the Symbios chip. The register is then read back to verify the previous write succeeded.

Subtest	Name	Description
1230	Symbios SCSI Scripts RAM	Performs a simple `data equals address` pattern test of the SCRIPT RAM.
1240	Symbios SCSI Interrupt	Copies a simple `SCRIPTS` instruction to SCRIPTS RAM on the Symbios controller. The SCRIPTS instruction is a simple `INT` opcode which, when executed by the Symbios chip, should cause a DMA interrupt to be logged. The DSP register of the Symbios chip is set to point to the instruction and the ISTAT register is polled until the interrupt is detected or the allotted time has elapsed.
1250	Symbios SCSI DMA engine	Writes a simple `SCRIPT` to Symbios SCRIPTS RAM which contains a `MEM MOVE` opcode. `SCRIPT` copies 256 bytes from one section of SCRIPTS RAM to another SCRIPTS RAM area. Once the SCRIPT has completed, the test verifies that the original block of data was copied to the destination area.

Table 47 io3000 Class 15 subtests

Subtest	Name	Description
1500	SCSI CDROM test unit ready	Issues a SCSI `test unit ready` command to all selected devices at least twice. The response to first command should return a SCSI “check condition” (not reported to the user) since the SCSI bus will have been reset. After approximately one second, the command is sent again. If the second test unit ready fails, an error is reported. The test unit ready command will not cause a SCSI data phase to occur.
1505	SCSI CDROM inquiry	Executes a SCSI `inquiry` command on every selected device. Verifies the device type field in the inquiry return data to indicate a CDROM.
1510	SCSI CDROM read capacity	Issues a SCSI `read capacity` command to every selected device.
1515	SCSI CDROM read	Issues a SCSI `read` command to every selected device. No data verification is performed.

NOTE Class 15 subtests also test DVD drives.

Table 48 **io3000 Class 16 subtests**

Subtest	Name	Description
1600	Tachyon PCI configuration space	Verifies the ability of the SAGA to access the Tachyon Fibre Channel controller by way of the PCI configuration space. Verifies the PCI vendor ID and device ID fields to be 0x107e and 0x0004, respectively. Also verifies the base address registers to be writable and readable.
1605	Tachyon PCI I/O and Memory space	Maps the Tachyon Fibre Channel controller through PCI configuration space so that the controller's CSRs may be accessed by way of PC memory space. The test writes a pattern to the world-wide name Hi (www_hi) in the Tachyon chip. The register is then read back to verify the previous write succeeded.

Table 49 **io3000 Class 17 subtests**

Subtest	Name	Description
1700	Tachlite PCI read	Verifies the processor's ability to read the PCI registers on the Tachlite card and verifies the vendor and device ID.

Subtest	Name	Description
1705	Tachlite PCI register reset	Confirms that the PCI registers reset with the proper values.
1710	Tachlite loopback	Resets the controller and then places it in internal loopback mode. it verifies that the controller.
1715	Tachlite loopback/DMA verify	Resets the controller, places it in internal loopback mode, verifies that loopback mode is set, performs a DMA data transfer through the controller, and then verifies the data. The data patterns are: zeros followed by all ones, a marching left one, alternating ones and zeros followed by other recognizable patterns. This tests the full data bus (64 bits on the SAGA) for connectivity errors.

Table 50 **io3000 Class 18 subtests**

Subtest	Name	Description
1800	Fibre Channel Mux Test Unit ready	Issues a Test Unit Ready to the alpa specified in the FC test parameter.
1805	Fibre Channel Mux Inquiry/ Device verify	Issues a Test Unit Ready and Inquiry for the alpa specified in the FC test parameter. The 64-bit lun values in the specified parameter words must be zero to direct the request to the Mux. The vendor ID and device type are verified to be the HP FC SCSI Mux.

User parameters

The test controller provides `io3000` with up to 37 user parameter words. Current parameters are defined in Table 51.

Table 51 **`io3000`** test parameters

Words	Description
0	See Table 52.
1	Device write enable mask—Each bit in the mask corresponds with a device. Bit 0 (MSB or left most bit in the parameter word) corresponds to device 0, bit 29 corresponds to the last (29th) device. Device 0 is the first device parameter location in user parameter word 8 (see Words 8-19 Device specification below). A binary '0' in a device's bit field means that SCSI writes (to that disk) are not enabled. Any test that does SCSI writes will not do so when the disk's corresponding write enable is turned off (binary '0'). The subtest will not be completely disabled though. This means that SCSI reads will still take place, but data verification will not be performed. The default setting for all disks is SCSI writes are disabled.
2	Transfer length (class 8 only)
3	Pattern (upper 32 bits)
4	Pattern (lower 32 bits)
5	Random seed
6	Not used.
7	Not used.
8-37	Device specification. See Figure 57 on page 192.

Table 52 **`io3000` user test parameter word 0 bit definition**

Bit	Description
0-23	Unused
24	Force code copy enable—Setting this bit causes all subtests that use encached routines to copy the code segment from flash into main memory. The copy will be performed even if the previous subtest already performed the copy. This feature should not be needed unless the code in main memory is being corrupted in a manner that cannot be easily detected.
25	Force SCSI firmware copy enable—Setting this bit causes all subtests that use the SCSI firmware to copy the firmware from flash into main memory. The copy is performed even if the previous subtest already performed the copy. This feature should not be needed unless the SCSI firmware in main memory is being corrupted in a manner that cannot be easily detected.
26	Full channel test disable (subtest 650)— Setting this bit causes subtest 650 to only test channels that are a power of 2, with the addition of channel 1007 (default). This reduces the run time significantly on this subtest.
27	Custom SCSI firmware enable—Setting this bit allows a non-supported version of the SCSI controller's firmware. If this option is set, user parameter word 6 must contain the length of the firmware file in half-words. Also, the custom firmware file must be loaded into flash at sector 6.
28	Multidisk enable (classes 6-7)—Setting this bit causes all specified disks to be tested in classes 6-7. The default is to only test the first disk (as specified in the user parameters) on each controller. Since classes 6-7 do disk transfers serially, little additional coverage is gained by running the tests on all the disks.

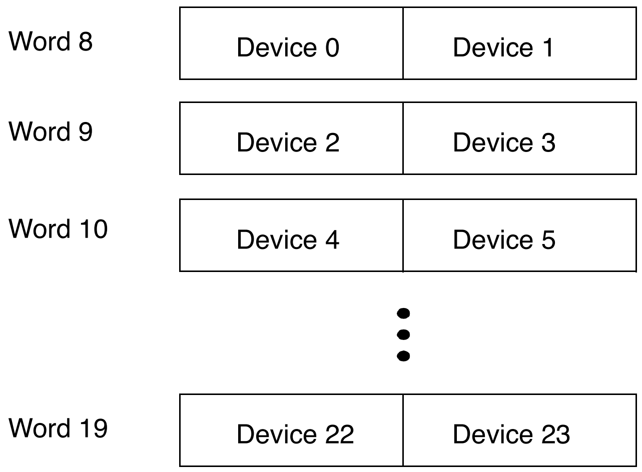
Bit	Description
29	User pattern enable (subtests 215, 240, classes 6-8)—Setting this bit causes each of the above specified tests to use the patterns as specified in user parameter 3 and 4, rather than a hard coded default pattern.
30	Random pattern enable (subtests 215, 240, classes 6-8)—Setting this bit causes each of the above specified tests to use subtest specific random patterns, rather than a hard coded default pattern.
31	March C complete (subtests 210, 235)—Setting this bit causes each of the above specified tests to do a complete bitwise march C test on the SRAM. The default is to do a quick version which takes about 25 percent of the time with about 99 percent of the coverage.

Device specification

Due to Core Logic SRAM space limitations, only 20 devices per SAGA can be tested at a time. Up to 24 SCSI devices can be specified using parameter words 8-19. Each of these parameter words contains two device specifications, as shown in Figure 57. Word 8 contains device specification 0 and 1. Word 9 contains 2 and 3, and so on.

Up to six Fibre Channel devices can be specified in parameter words 20-37. Each device requires three parameter words as shown in Figure 58 and Table 54.

Figure 57 **`io3000` test parameter device specification for directly attached SCSI targets (words 8-19)**

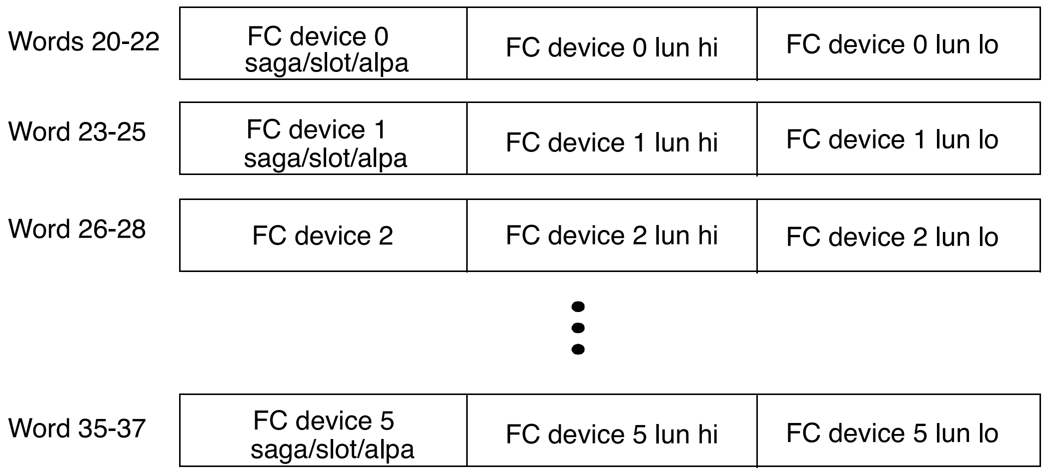


Fields within each parameter word specify the devices as shown in Table 53. Bit 0 is the upper (left most) bit in the parameter word.

Table 53 **`io3000` bit definition for direct SCSI device specification (words 8-19)**

Bit	Definition
0-3	SAGA
4-7	Slot
8-11	SCSI target
12-15	SCSI lun
16-19	SAGA
20-23	Slot
24-27	SCSI target
28-31	SCSI lun

Figure 58 **`io3000` test parameter device specification for Fibre Channel attached SCSI targets (words 20-37)**



Fields within each parameter word specify the devices as shown in Table 54. Bit 0 is the upper (left most) bit in the parameter word.

Table 54 **`io3000` bit definition for Fibre Channel attached SCSI device specification (words 29-37)**

Location	Bit	Definition
Word n	0-3	SAGA
Word n	4-7	Slot
Word n	8-31	AL_PA (or D_ID)
Word n+1	0-31	FC lun hi
Word n+2	0-31	FC lun lo

Devices are numbered according to their position in the parameter list. A device can be specified in any of the device specification locations in user parameter space. An unused device parameter should be initialized such that the slot field is 0xf (that is, device specification of 0x0f00). Therefore, if both device parameters in a given parameter word are unused, the parameter word would be set to 0x0f000f00.

As an example, to specify a disk on SAGA 0x4, slot 0x2, SCSI identification 0xa, SCSI lun 0x0, set parameter word 8 to 0x42a00f00. The lower (right) half of the parameter word has the slot field set to the 0xf. The device number is 0 since it was entered in device 0 parameter location.

As an example of specifying an FC device on saga 0x2, slot 0x1, AL_PA 0xcd, lun hi 0x00010000 and lun lo 0x000000, set parameter word 20 to 0x210000cd, parameter word 21 to 0x00010000, and parameter word 22 to 0x000000. This example applies to a Galaxy array with a single controller and a lun of 01. The Galaxy controller expects the lun to be in the 3rd and 4th nibble of the lun hi field. Each Fibre Channel device type can use a different method of specifying luns. Check the device specific documentation for help in specifying the lun hi and lun lo parameters for io3000.

When using cxtest to run `io3000`, SAGAs are referred to according to their position identification stamped in the node sheet metal. Table 55 correlates SAGA names with SAGA numbers:

Table 55 `io3000` SAGA name to number correlation

SAGA name	SAGA number
IOLF_A	4
IOLF_B	0
IOLR_A	5
IOLR_B	1
IORR_A	6
IORR_B	2
IORF_A	7
IORF_B	3

io3000 error codes

When a failure is encountered, an event code is set along with an error message. The least significant 12 bits of the event code contain the error code. Table 56 lists the `io3000` error codes.

io3000 general errors

`io3000` general error codes post no error messages. Table 56 shows each `io3000` general error code.

Table 56 **`io3000`** **general error codes**

Code	Description
0x1	Core logic SRAM allocation failure. This is a software error that indicates that the software has run out of core logic SRAM to store internal data structures.
0x2	Interrupt allocation failure. This is a software error that indicates that the software has run out of available external interrupt vectors.
0x3	No device specified. `io3000` was looking for a device in the user parameters and found none.
0x4	An invalid combination of processors has been selected. Due to the shortage of core logic SRAM, the per processor stack space is only 1Kbytes. This has proven to be inadequate for portions of `io3000.`Therefore, processor selection has been limited such that adjacent processors cannot be selected simultaneously. Also, processor 0xf can not be used.
0x5	A random number seed of 0 was specified. The seed must be nonzero.

io3000 device specification errors

`io3000` device specification errors post the following error message:

```
SAGA_name/ctlr_num/tgt_num/lun_num
```

Example of `io3000` device specification error message:

```
IOLF_A/ct0/idf/lu0
```

Table 57 shows each `io3000` general error code.

Table 57 **`io3000`** device specification error codes

Code	Description
0x8	Duplicate device specification. The same device was specified multiple times in the user parameters.
0x9	Invalid SAGA number. The number in the SAGA field of one of the device parameters is invalid (> 7).
0xa	Invalid slot number. The number in the slot field of one of the device parameters is invalid.
0xb	Invalid logical unit number. The number in the logical unit field of one of the device parameters is invalid (> 7).
0xc	Duplicate Fibre Channel device specification.
0xd	Invalid Fibre Channel SAGA number.
0xe	Invalid Fibre Channel slot number.
0xf	Invalid Fibre Channel LUN number.

io3000 SAGA general errors

`io3000` SAGA general errors post the following error message:

```
SAGA_name
```

Example of `io3000` SAGA general error message:

```
IOLF_B
```

Table 58 shows each `io3000` SAGA general error code.

Table 58 **`io3000` SAGA general errors**

Code	Description
0x10	An SAGA specified in the user parameters was not available.
0x11	Unable to reset SAGA. io3000 was unsuccessful in setting or resetting the SAGA online bit on it's associated SPAC.
0x12	Data prefetch timeout. The prefetch valid bits in the channel context never became valid, or did so too slowly.

io3000 SAGA CSR errors

`io3000` SAGA CSR error codes post the following error message:

SAGA_name/address/act_val/exp_val

Example of `io3000` SAGA CSR error message:

```
IOLF_B/fc010008/00e0000f0c000000/00e0000f0c100000
```

Table 59 shows each `io3000` SAGA CSR error code.

Table 59 **`io3000` SAGA CSR errors**

Code	Description
0x20	SAGA CSR failure.
0x21	SAGA PTE failure
0x22	SAGA read TLB even failure.
0x23	SAGA read TLB even failure.
0x24	SAGA read TLB odd failure.
0x25	SAGA write TLB even failure.
0x26	SAGA write TLB odd failure.

io3000 SAGA ErrorInfo CSR error

The `io3000` ErrorInfo CSR error code posts the following error message:

```
SAGA_name/cause_bit/address/act_val
```

Example of `io3000` SAGA ErrorInfo CSR error:

```
IOLF_A/5/fc210098/10e0000f0c000000
```

Table 60 shows the `io3000` SAGA ErrorInfo CSR error code.

Table 60 **`io3000` SAGA ErrorInfo CSR error**

Code	Description
0x50	SAGA ErrorInfo CSR failure.

io3000 SAGA ErrorCause CSR errors

`io3000` SAGA ErrorCause CSR error codes 0x54 and 0x55 post the following error message:

```
SAGA_name/address/act_val
```

Example of `io3000` SAGA ErrorCause CSR error message for 0x54 and 0x55 codes:

```
IOLF_A/fc210080/0000010000000000
```

`io3000` SAGA ErrorCause CSR error code 0x58 posts the following error message:

```
SAGA_name/ctlr_num/address/act_valPIC_name/address/
act_val
```

Example of `io3000` SAGA ErrorCause CSR error message for 0x58 code:

```
IOLF_A/ctl/fc210108/0010000000000000/fc210080/
0000010000000000
```

Error 0x58 occurs when a bit in the controller's corresponding SAGA PCIxStatCSR is set. Specifically, the bits that cause this error are SawAddrParErr, BrokenDev, and SawDataPtyErr.

Table 61 shows each `io3000` SAGA ErrorCause CSR error code.

Table 61 **`io3000`** **SAGA ErrorCause CSR errors**

Code	Description
0x54	SAGA ErrorCause CSR failure.
0x55	SRAM parity error expected. This error occurs when the cci_rdperr bit in the SAGA ErrorCause does not get set when SRAM parity errors are forced.
0x58	PCIx status failure.

io3000 SAGA SRAM errors

`io3000` SAGA SRAM error codes post the following error message:

SAGA_name/address/act_val/exp_val

Example of `io3000` SAGA SRAM error message:

```
IOLF_A/f81fc00080/5555555555555555/55f5555555555555
```

Table 62 shows each `io3000` SAGA SRAM error code.

Table 62 **`io3000`** **SAGA SRAM errors**

Code	Description
0x60	SRAM access failure. `io3000` was unable to successfully write and read SRAM on the SIOB.
0x61	SRAM march C failures. A failure was detected during the march C test of SRAM on the SIOB. The range of codes refer to incremental stages of the march C algorithm as follows: none - write ~patt (->) 0x61 - read ~patt, write patt (->) 0x62 - read patt, write ~patt (->) 0x63 - read ~patt, write patt (<-) 0x64 - read patt, write ~patt (<-) 0x65 - read ~patt (<-)
0x66	SRAM read access width test failed.
0x67	SRAM write access width test failed.
0x68	SRAM pattern test failure.

io3000 controller general errors

`io3000` Controller general error codes post the following error message:

SAGA_name/ctlr_num

Example of `io3000` controller general error message:

```
IOLF_B/ct0
```

Table 63 shows each `io3000` general controller error code.

Table 63 **`io3000` Controller general errors**

Code	Description
0x80	The controller was not detected as present per the SAGA's PcixStatCSR PCI card present bits.
0x81	SCSI flash read error. `io3000` was unable to successfully read the SCSI controller's flash memory.
0x82	`io3000` was unable to initialize the controller.
0x83	The loopback test on the controller failed.
0x84	The controller was unexpectedly offline.

io3000 PCI errors

`io3000` PCI error codes post the following error message:

SAGA_name/ctlr_num/address/act_val/exp_val

Example of `io3000` PCI error message:

```
IOLF_B/ct1/f804000010/ffffff01/00000001
```

Table 64 shows each `io3000` PCI error code.

Table 64 **`io3000` PCI errors**

Code	Description
0x90	PCI vendor ID failure. `io3000` was unable to successfully read the controller's PCI vendor ID
0x91	PCI device ID failure. `io3000` was unable to successfully read the controller's PCI device ID.
0x92	PCI io base address register failure. `io3000` was unable to successfully read and write the controller's PCI io base address register.
0x93	PCI memory base address register failure. `io3000` was unable to successfully read and write the controller's PCI memory base address register.
0x9A	Symbios SCRATCHA register failure. `io3000` was unable to successfully read and write the controller's SCRATCHA register.

io3000 controller command errors

`io3000` controller command error codes post the following error message:

```
SAGA_name/ctlr_num/tgt_num/lun_num/comp_stat/
scsi_stat:sense_key:sense_code:sense_code_qualifier
```

Example of `io3000` controller command error message:

```
IOLF_A/ct0/idf/lu0/comp:0/scsi:2
```

Table 65 shows each `io3000` controller command error code.

Table 65 **`io3000` controller command errors**

Code	Description
0xc0	SAGA command completion failure. This means a queued command has failed and has a nonzero completion status.
0xc1	SCSI status failure. This means a SCSI command has terminated with nonzero SCSI status.

io3000 DMA error

The `io3000` DMA error code posts the following error message:

```
SAGA_name/ctlr_num/tgt_num/lun_num/address/act_val/
exp_val
```

Example of `io3000` DMA error message:

```
IOLF_A/ct0/idf/lu0/0004148200/a5a5a5a4/a5a5a5a5
```

Table 66 shows the `io3000` DMA error code.

Table 66 **`io3000`** DMA error

Field	Description
0xd0	Data miscompare on DMA. Data in the destination buffer does not match data in the source buffer.

io3000 SCSI inquiry error

The `io3000` SCSI inquiry error code posts the following error message:

```
SAGA_name/ctlr_num/tgt_num/lun_num/act_val/exp_val
```

Example of `io3000` SCSI inquiry error message:

```
IOLF_A/ct0/idf/lu0/1/0
```

Table 67 shows the `io3000` SCSI inquiry error code.

Table 67 **`io3000`** SCSI inquiry error

Code	Description
0xe0	Wrong peripheral device type found in SCSI inquiry return data.

io3000 Symbios controller specific errors

`io3000` Symbios controller specific error codes post the following error message:

```
SAGA_name/ctlr_num/address/act_val/exp_val
```

Example of `io3000` Symbios controller specific error message:

```
IOLF_B/ctl/f804000010/ffffff01/00000001
```

Table 68 shows each io3000 Symbios controller specific error code.

Table 68 **`io3000`** Symbios controller specific errors

Code	Description
0x110	General failure detected on Symbios controller.
0x113	Error detected during SCRIPTS RAM pattern testing.
0x114	Interrupt test failed. The address is the address of the interrupt register. The expected data contains the bit of that interrupt register expected to be set, while the actual data contains the entire contents of the ISTAT or DSTAT register.
0x115	Symbios DMA test failed.

io3000 Tachyon controller specific errors

`io3000` Tachyon controller specific error codes post the following error message:

```
SAGA_name/ctlr_num/address/act_val/exp_val
```

Example of `io3000` Tachyon controller specific error message:

```
IOLF_B/ctl/f804000010/ffffff01/00000001
```

Table 68 shows each io3000 Tachyon controller specific error code.

Table 69 **`io3000`** Tachyon controller specific errors

Code	Description
0x90	PCI vendor ID not as expected.
0x91	PCI device ID not as expected.
0x93	PCI memory address Base Register write/read fail.

io3000 DIODC driver errors

`io3000` Diagnostic I/O Dependent Code (DIODC) driver error codes post the following error message:

SAGA_name/ctlr_num/tgt_num/lun_num/ctlr_status/dev_status

Example of `io3000` DIODC driver error message:

```
IOLF_A/ct1/ct0/idf/lu0/81/0
```

Table 70 shows each `io3000` Symbios controller specific error code.

Table 70 **`io3000` DIODC controller specific errors**

Code	Description
0x120	General controller error.
0x121	No controller detected in the selected slot.
0x122	Unsupportable controller detected.
0x130	General failure detected.
0x131	Attempted to open a device. An open consists of a SCSI Test Unit Ready followed by a SCSI Inquiry command. See the controller status codes for more details.

Table 71 **Symbios controller status codes**

Code	Description
0x81	Symbios Queue Overflow.
0x82	Symbios Queue Empty.
0x88	Invalid handle.
0x84	Timeout during select.
0x85	Timeout detected waiting for a SCRIPT to complete.
0x86	Device transitioned to an unexpected phase.
0x87	Device in an undefined SCSI phase.
0x88	Target not online.

10 mem3000

This chapter describes `mem3000`, a memory test for V2500/V2600 systems.

`mem3000` is core logic flash-based memory diagnostic that verifies the functionality of the memory subsystem.

`mem3000` requires a node with a minimum of one processor with two memory boards that must be installed in pairs in order for the test to properly execute.

mem3000 classes and subtests

`mem3000` verifies the V2500/V2600 memory subsystem using the Test Controller.

`mem3000` requires one node with a minimum of one process with associated SPAC and two EWMBs with associated SMACs.

`mem3000` consists of a series of tests grouped together in classes beginning with verification of the most basic functionality and progressing toward more complex functionality. Each class has several subtests that target specific functionality.

mem3000 classes

`mem3000` has six classes of tests shown in Table 72.

Table 72 **`mem3000` test classes**

Class	Description
1	Verifies the operation of the diagnostic CSRs on each EMB.
2	Verifies the tag field.
3	Verifies the data field.
4	Verifies the various coherent and noncoherent transactions.
5	Verifies the ECC.
6	Verifies miscellaneous memory capabilities.

Class 1 and class 2 subtests (with the exception of subtest 150) can be configured to test a single EMB. Subtest 640 can also be used to test a single EMB.

Running any other Class 4, 5, or 6 subtest with only one EMB selected is not recommended.

Class 3 subtests and subtest 150 use memory interleaving and do not work with a single EMB selected.

mem3000 subtests

The `mem3000` subtests are listed in Table 73 through Table 78.

Table 73 **`mem3000`** class 1 subtests

Subtest	Description
100	Verifies the diagnostic CSRs can be written and read
101	Verifies the other SMAC CSRs can be written and read
110	Verifies data can be written and read on each DIMM using the diag CSRs
120	Verifies ECC can be written and read on each DIMM using the diag CSR
130	Verifies the tag can be written and read on each DIMM using the diag CSRs
140	Verifies memory lines on each DIMM can be initialized using the diag CSRs
150	Verifies the first 64 memory lines of each EWMB using various data patterns
190	Verifies that each DIMM passes DIMM probing similar to the POST DIMM probe.

Table 74 **`mem3000`** class 2 subtests

Subtest	Description
200	Verifies the tag portion of a memory line using different patterns
210	Verifies the tag portion of a memory line using an addressing pattern
211	Verifies the tag portion of a memory line using a byte uniqueness pattern, i.e. 0x0001020304050607
230-238	Verifies the tag portion of a memory line using the MarchC algorithm and different patterns

Table 75 **`mem3000`** **class 3 subtests**

Subtest	Description
300	Verifies the memory lines on each DIMM can be written and read using coherent operations
310	Verifies the data portion of a memory line using an addressing pattern with coherent operations
311	Verifies the data portion of a memory line using a byte uniqueness pattern with coherent operations
330-338	Verifies the data portion of a memory line using the MarchC algorithm and different patterns with coherent operations

Table 76 **`mem3000`** **class 4 subtests**

Subtest	Description
400	Verifies load and store transactions to memory
410	Verifies data flush transactions to memory
420	Verifies non-coherent transactions to memory

Table 77 **`mem3000`** **class 5 subtests**

Subtest	Description
500	Verifies ECC single bit data portion errors are detected, logged, and corrected
501	Verifies ECC single bit tag portion errors are detected, logged, and corrected
502	Verifies ECC single bit ECC portion errors are detected, logged, and corrected

Subtest	Description
510	Verifies ECC double bit data errors are detected and logged using coherent operations
520	Verifies ECC double bit data errors are detected and logged using non-coherent operations
530	Verifies that ECC errors are ignored when disabled

Table 78 **`mem3000` class 6 subtests**

Subtest	Description
600	Verifies the memory system detects and reports accesses to all illegal and/or invalid memory space
610	Verifies the memory system detects and reports error conditions when the memory tag state is ERROR
640	Determines whether 80-bit or 88-bit DIMMs are installed

User parameters

The Test Controller allows `mem3000` eight user parameters. Table 79 defines these parameters:

Table 79 User parameter definitions

Words	Usage
0/1	64-bit user pattern 0 used in subtests 238 and 338 (defaults=0xa5a5a5a5/0xa5a5a5a5)
2/3	64-bit user pattern 1 used in subtests 238 and 338 (defaults=0x5a5a5a5a/0x5a5a5a5a)
4	Denotes 88-bit DIMMs are installed (default=2)
5	Denotes test is to run with errors disabled (default=0)
6/7	Octant mask. (default: 0xffffffff 0xffffffff)

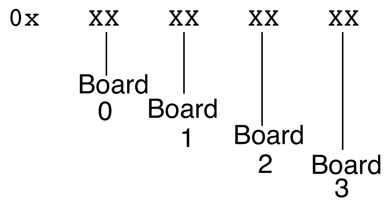
Parameter 4 defaults to the value 2 causing the test to automatically probe all known DIMMs to determine their type: 80- or 88-bit DIMMs. The test then changes the parameter from 2 to 0 or 1. It is set to 1 if only 88-bit DIMMs were found. If any 80-bit DIMMs were found, it is set to 0.

Parameters 6 and 7 default to the value 0xfffffffff, the bit mask that indicates whether a memory octant should be tested. When the Test Controller is started, `mem3000` changes the values to match the memory that `POST` enabled on the node.

Each range is 0x0 through 0xffffffff. Each byte represents the physical octant mask for a memory board.

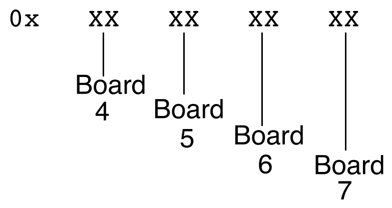
Parameter 6 contains the masks for boards 0 through 3 in the order shown in Figure 59.

Figure 59 **Format of parameter 6**



Parameter 7 contains the masks for boards 4-7 in the order shown in Figure 60.

Figure 60 **Format of parameter7**



As an example, the Octant Mask for board 0 is encoded in the first two digits of Parameter 6.

Subtests 100, 101, 150, and 310-338 DO NOT use the Octant Mask. Subtests 100 and 101 test CSRs on all enabled SMACs. Subtests 150 and 310-338 use the Main Memory Map built by POST.

mem3000 error codes

When a failure is encountered, an event code is set along with an error message. The least significant 12 bits of the event code contain the error code. Table 80 lists the `mem3000` error codes.

Table 80 **`mem3000`** error codes

Code	Meaning
001	Diagnostic address CSR miscompare occurred (upper 32-bits)
002	Diagnostic address CSR miscompare occurred (lower 32-bits)
003	Diagnostic data CSR miscompare occurred (used only by class 1)
004	Diagnostic data CSR miscompare occurred (in upper 32-bits)
005	Diagnostic data CSR miscompare occurred (in lower 32-bits)
008	Miscompare occurred in the upper 32-bits of the CSR
009	Miscompare occurred in the lower 32-bits of the CSR
010	Memory data miscompare occurred
011	Memory data miscompare occurred (upper 32-bits)
012	Memory data miscompare occurred (lower 32-bits)
013	Memory data matched when it shouldn't have (upper 32 bits)
014	Memory data matched when it shouldn't have (lower 32-bits)
020	Miscompare occurred in the upper 32-bits of the tag
021	Miscompare occurred in the lower 32-bits of the tag
022	The tag changed when it shouldn't have
030	ECC data miscompare occurred
031	An ECC error was logged when it shouldn't have been
032	SMAC did not correct the single bit ECC failure as expected

Code	Meaning
033	SMAC did not log the occurrence of a single bit ECC failure
035	SMAC did not log the occurrence of a double bit ECC failure
040	Data miscompare error occurred in sequence #1 of MarchC test (upper 32-bits)
041	Data miscompare error occurred in sequence #1 of MarchC test (lower 32-bits)
042	Data miscompare error occurred in sequence #2 of MarchC test (upper 32-bits)
043	Data miscompare error occurred in sequence #2 of MarchC test (lower 32-bits)
044	Data miscompare error occurred in sequence #3 of MarchC test (upper 32-bits)
045	Data miscompare error occurred in sequence #3 of MarchC test (lower 32-bits)
046	Data miscompare error occurred in sequence #4 of MarchC test (upper 32-bits)
047	Data miscompare error occurred in sequence #4 of MarchC test (lower 32-bits)
060	A semaphore operation did not trigger
070	Incorrect data returned for a semaphore operation
080*	Incorrect info in SMAC error CSRs (single bit data ECC - read)
090*	Incorrect info in SMAC error CSRs (single bit tag ECC - read)
0a0*	Incorrect info in SMAC error CSRs (double bit data ECC - read)
0b0*	Incorrect info in SMAC error CSRs (double bit data ECC - coh_inc op)
0c0*	Tag state did not equal ERROR as it should have

Code	Meaning
0d0*	Tag state did not equal INVALID as it should have
0e0*	An unexpected error was detected in the SMAC error CSRs
100*	Uninstalled Memory
110*	Invalid CSR
120*	Network Cache
130*	Unprotected Memory
140*	Alternate Interleave
150	An HPMC was detected on access to the specified address
200	Denotes the EWMB contains all 80-bit DIMMs
201	Denotes the EWMB contains all 88-bit DIMMs
202	Denotes the EWMB contains a mixture of 80-bit and 88-bit DIMMs
220	Some portion of test code is copied to memory and branched to in attempt to load the code into the icache. The initialization routine detected that code failed to implicitly encache when executed from coherent memory.

The asterisks next to the error codes listed in Table 80 actually indicate a range of events as shown in Table 81.

Table 81 **Extended range for error codes**

Code	Meaning
code+1	Error cause CSR miscompare error (upper 32-bits)
code+2	Error cause CSR miscompare error (lower 32-bits)
code+3	Error info CSR miscompare error in the err type field
code+4	Error info CSR miscompare error in the ENUM field
code+5	Error info CSR miscompare error in the cc/msg field

Code	Meaning
code+6	Error address CSR miscompare error (upper 32-bits)
code+7	Error address CSR miscompare error (lower 32-bits)
code+8	Error info CSR syndrome code miscompare error

Table 82 **Patterns used in specified subtests**

Subtest	Pattern
230/330	0x7f7f7f7f7f7f7f7f and 0x8080808080808080
231/331	0xbfbfbfbfbfbfbfbf and 0x4040404040404040
232/332	0xdfdfdfdfdfdfdfdf and 0x2020202020202020
233/333	0xefefefefefefefef and 0x1010101010101010
234/334	0xf7f7f7f7f7f7f7f7 and 0x0808080808080808
235/335	0xfbfbfbfbfbfbfbfb and 0x0404040404040404
236/336	0xfdfdfdfdfdfdfdfd and 0x0202020202020202
237/337	0xfefefefefefefefe and 0x0101010101010101
238/338	0xa5a5a5a5a5a5a5a5 and 0x5a5a5a5a5a5a5a5a (user parameters 0-3)

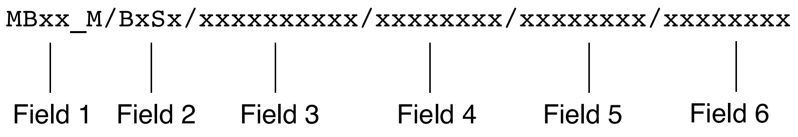
Error messages

When a failure is encountered an event code is set along with an error message. The least significant 12 bits of the event code contain the error code. The error codes and their error message descriptions are defined in the following section. Error codes can have one of three different formats.

Type one error format

Type one errors are used by many of the subtests. Figure 61 shows the format of the type one error format.

Figure 61 **Type one error message format**



```
MBxx_M/BxSx/xxxxxxxxxx/xxxxxxxx/xxxxxxxx/xxxxxxxx
```

There are six fields separated by / symbols. The meaning of each field is as follows:

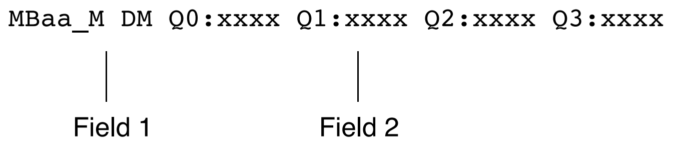
- Field 1—Specifies the EWMB on which the failure was detected
- Field 2—Specifies the DIMM on which the failure was detected
- Field 3—Specifies the failing 40-bit address
- Field 4—Specifies the actual 32-bits of data
- Field 5—Specifies the expected 32-bits of data
- Field 6—Specifies the error as follows:
  - COH-OP—Coherent operation
  - DCSR—Diagnostic CSR access
  - CSR DATA - CSR data mismatch
  - DECC— ECC mismatch
  - DTAG—TAG mismatch
  - DDAT—DATA mismatch

Type two errors

The type two error is used only by subtest 640 which determines what type of DIMMs are installed on the first EWMB specified.

A type two error is shown Figure 62.

Figure 62 **Type two error message format**



```
MBaa_M DM Q0:xxxx Q1:xxxx Q2:xxxx Q3:xxxx
```

The two fields of the type two error are as follows:

- Field 1—Specifies the EWMB to which the information pertains
- Field 2—Specifies the type of DIMM detected as follows:
  - x—Non-existent DIMM
  - 0—80-bit DIMM
  - 1—88-bit DIMM

The correspondence of these values to the actual DIMM locations is shown in Figure 63.

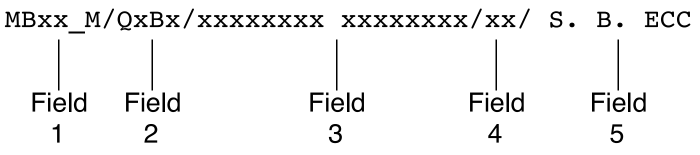
Figure 63 **Corresponding type two values to DIMM location**

```
     Q0:xxxx Q1:xxxx Q2:xxxx Q3:xxxx
Bus:    0123    4567    0123    4567
```

Type three errors

The type three error (shown in Figure 64) is used only by class 3 subtests to report spurious single-bit ECC errors that occur during testing. The test is designed specifically to bring out these types of failures. However, if failures of other types occur, they are reported in their respective format.

Figure 64 **Type 3 error message format**



```
MBxx_M/QxBx/xxxxxxxx xxxxxxxx/xx/ S. B. ECC
```

There are five fields separated by / symbols. The meaning of each field is as follows:

- Field 1—Specifies the EWMB on which the failure was detected
- Field 2—Specifies the DIMM on which the failure was detected
- Field 3—Specifies the SMAC error address CSR value
- Field 4—Specifies the syndrome bits
- Field 5—Reminds that this is a single bit ECC error

Notes on mem3000

There is a dependency upon POST to initialize the memory system. This test uses many of the CSR values from POST and does not reconfigure the system. There are some exceptions in which CSR values need to be changed in order for the test to run. In these cases, CSR values should be returned to their previous value upon successful completion of the subtest. If a failure occurs, these CSRs may not be returned to their pre-test state in an attempt to save the failing state and configuration.

`mem3000` currently uses the following algorithm for selecting processors to be used in testing: A list is made of processors. Even numbered processors under 16 are first, then odd numbered processors under 16, followed by even then odd CPUs over 16. This ordered list is then used to assign one processor per memory board.

The EWMBs must be installed in pairs (1 even for each odd). Three pairs is not a valid configuration and POST will hardware deconfigure the extra pair. Therefore, either 2, 4, or 8 EWMBs must be installed.

mem3000 uses memory that was enabled by POST to do the pre-test initialization and encaching. Therefore, the Octant Mask parameters (6 and 7) are ignored during subtest init. As a result, lines that are not tested may be re-initialized and used during the encaching sequence.

Subtest 150 and the class 3 subtests use memory interleave and thus test over a range of EWMBs. The memory tested is that which was enabled by POST in the Main Memory Map.

Subtest 150 and the class 3 subtests use coherent accesses to test consecutive memory lines which are interleaved across EWMBs, buses and banks. As a result, parameters 6 and 7 are ignored. When a failure occurs, the failing 40-bit address can be used to determine which logical row, bank, bus, and board was being accessed. The failing DIMM field (QxBx) takes interleaving into account and reports the actual physical Quadrant and Bus that failed.

Depending on the configuration, subtest 640 may not be able to test all EWMBs in the node at once. If subtest 640 does not report the status of all EWMBs the first time, deselect the EWMBs that were tested and rerun the subtest.

Subtests in class 6 will produce HPMCs (indicated by the Test Controller printing the # character). These are expected.

11 Scan test

The Exemplar scan test (`est`) is a diagnostic utility that uses the system scan hardware making it possible to perform connectivity tests and to test gate array internal registers. The `est` utility runs on the SSP and sends scan instructions to a given node by way of the Ethernet.

`est` utility test environment

`est` is started on the SSP and is located in /spp/bin/est. The user has the option of either starting up a user interface or having the `est` utility run a script.

`est` works on one node at a time by sending scan instructions and data and receiving the results over the diagnostic ethernet connection.

Since `est` has to communicate closely with the Utilities board, no other diagnostic can be run at the same time. Also, while `est` is moving data through the scan rings, the operating system can not be running.

`est` works on the JTAG scan rings throughout the system. Tests provided are:

- Ring (test command `r`)—Moves data through the scan rings to make sure the rings are connected and that basic scan hardware is operational.
- Dc connectivity (command `d`)—Checks that wires on the boards between scan devices are intact (no shorts or opens).
- Ac connectivity (command `a`)—Examines wires on the boards. Ac tests look for timing problems between parts at full speed. If dc connectivity patterns passed, but ac connectivity failed, the failure is bound to be timing related.
- Gate array (command `g`)—Executes scan tests internal to selected arrays. When these tests fail, the array usually has to be replaced.

These tests are listed in the order in which they should normally be run.

Control of utility board

To prevent unexpected shutdowns from hardware that is sensitive to scan operations, `est` takes control of a power signal on the Utility board. To control this signal, `est` must freeze some of the bits of the Utilities board. Therefore, when `est` starts, it automatically performs the `id_verify` operation and ring test (command `r`) on the Utility Ring (ring 22). `est` then locks the bits to control the power signal. If the user needs to run the `id_verify` or ring test function, scan operations will occur in all scan rings except the Utility Ring.

To perform ID and ring checks in the utility system, the user should turn off the power control feature either though the command line argument `-p` or through a runtime option command (`power_control`). The latter should seldom occur, because `est` automatically runs these tests on the utility scan path at start up and reports any errors found.

est exit and reset

To quit, `est` calls a script called `est_exit`. The default script performs a `do_reset` function to reset the node under test. When the CTI cables are tested, `est` directs `est_exit` and `do_reset` to reset the entire complex. To accomplish this reset, `est` passes to the `est_exit` script a parameter that indicates which node of the complex to reset. The script then hands this parameter to `do_reset` which then performs the reset operation.

The default script resides at /spp/scripts/est_exit. If the user wishes to run his own version, he should create the file in a local subdirectory ./scripts/est_exit. If est sees such a file, it will runs the local copy instead of the default. The purpose of the default `do_reset` function is to make sure that the utility system is restored in order to monitor environmental conditions.

est user interfaces

`est` can be run from either a GUI or a command line interface. The est GUI is described in “Running the est GUI” on page 224. The command line interface is described in “Running est from command line” on page 238.

Running the est GUI

The `est` GUI may be started at the command prompt. The following is the `est` command usage:

```
/spp/bin/est [-option] node_number
```

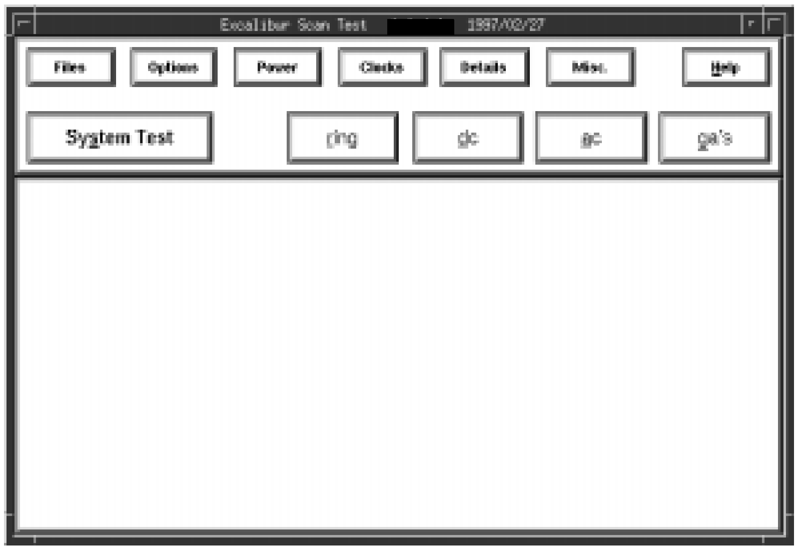
As an example to bring up the GUI and test node 0, enter the following command:

```
% /spp/bin/est -x 0
```

Table 83 on page 238 provides a complete list of options.

Figure 65 shows the `est` main window.

Figure 65 **`est` main window**



The main window has two sections. The upper section has two rows of buttons. The top row provides the user several options to control system and test parameters, and the bottom row allows the user to run all available tests. The lower section is the main window pane that displays messages and test status.

The lower set of buttons allows the user to quickly and easily run the scan tests in a wholesale fashion. The test can be modified to run fewer patterns, to loop continuously or for a finite number of times, to test non-default limits, etc.

Each button is explained in the following sections.

System Test button

Clicking the System Test button runs each set of tests in the following order: ring tests, dc connectivity tests, ac connectivity tests, and gate array tests. It is equivalent to entering the `r`, `d`, `a`, and `g` commands from the command line interface.

ring button

Clicking the ring button runs the system scan tests on all rings and scan paths using the default patterns. It is equivalent to entering **`r`** from the command line interface. Most scan rings are defined in the IEEE 1149.1 JTAG specification.

dc button

Clicking the dc button runs only the dc connectivity tests using default parameters. It is equivalent to entering **`d`** from the command line interface.

ac button

Clicking the ac button runs only the ac connectivity tests using default parameters. It is equivalent to entering **`a`** from the command line interface.

ga's button

Clicking the ga's button runs only the gate array tests using default parameters. It is equivalent to entering **`g`** from the command line interface. When the Limit Test Patterns option is set in the Options window, however, clicking the ga's button runs the gate array tests with the limited number of patterns specified. See "Options button" on page 226.

Files button

Clicking the Files button opens pop-up menu with three selections:

- Execute Scripts—Runs a file containing `est` commands.
- Reset Log File—Clears the log file.
- Exit—Closes the est main window and exits the program.

Options button

Clicking the Options button opens pop-up menu with seven selections:

- Log_File—Generates a log file and stores it in /spp/data/est.log.
- Stop On Error—Causes the test(s) to halt whenever an error is detected.
- Limit Test Patterns—Limits the number of test patterns so that the test runs in approximately one-half the normal time. Test coverage drops to approximately 90%.
- Limit Error Report—Limits the length of the error report to 10 errors.
- Normal Font Size—Prints status to the main window pane using the standard font size.
- Large Font Size—Prints status to the main window pane using a large font size.
- Show time—Prints current time and date.

Power button

Clicking the Power button opens pop-up menu with four selections:

- Upper—Sets the upper limit of the power supplies.
- Nominal—Sets the power supplies to their nominal values.
- Lower—Sets the lower limit of the power supplies.
- Status—Displays the current settings of the power supply voltages (upper, normal, or lower). When this option is invoked, it displays both the power supplies and clock settings.

Clocks button

Clicking the Clocks button opens pop-up menu with four selections:

- Upper—Sets the upper limit of the system clocks.
- Nominal—Sets the system clocks to their nominal values.
- External—Selects an external clock from the ECUB.
- Status—Displays the current settings of the power supply voltages (upper, normal, or lower). When this option is invoked, it displays both the clock and power supplies settings.

Details button

Clicking the Details button opens a pop-up menu with seven selections:

- P/F Each Pattern—Displays the number and the results of each pattern in the test.
- Test File Msgs—Prints the pattern file, both instructions and data. This option is primarily used for troubleshooting `est`.
- Show Scan Instr—Displays the instruction portion of the scan packet. This option is primarily used for troubleshooting `est`.
- Show Scan Data—Displays the data portion of the scan packet.This option is primarily used for troubleshooting `est`.
- Show SDP Data—Displays the scan data protocol portion of the scan packet. This option is primarily used for troubleshooting `est`.
- Show GUI Commands—Displays each command in the main window pane. This option is useful for writing test scripts.
- Enable GUI Commands—Toggles between enabling and disabling GUI commands. This option is used with the Show GUI option to assist in writing test scripts.

Misc. button

Clicking the Misc. button opens pop-up menu with nine selections:

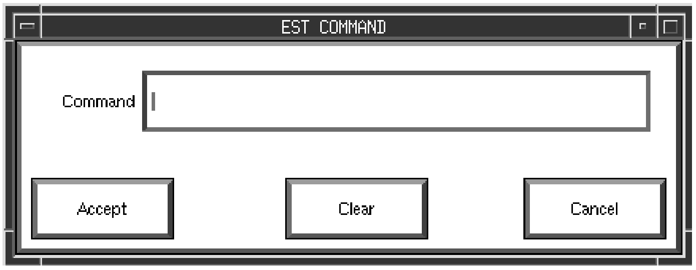
- Goto Safe State—Places the system hardware in a safe state
- Verify Config—Compares returned test scan data from the JTAG interface against the configuration file.

- Command Menu—Opens the command line window which allows the user to enter `est` commands directly from the GUI system.
- Scan Debug Menu—Opens the debug window.
- Connectivity Test Menu—Opens the connectivity test window.
- Gate Array Test Menu—Opens the gate array test window. Gate array tests use test vectors that have been generated for the certain arrays (each array has multiple files associated with it).
- Sci Test Menu—Opens the SCI test window. The tests verify the Coherent Toroidal Interface (CTI) cables between nodes.
- Abort—Stops the currently running test

Command line window

The `est` command line window allows the user the freedom to enter a command directly from the `est` GUI system. Figure 66 shows the `est` command line window.

Figure 66 **`est` command line window**

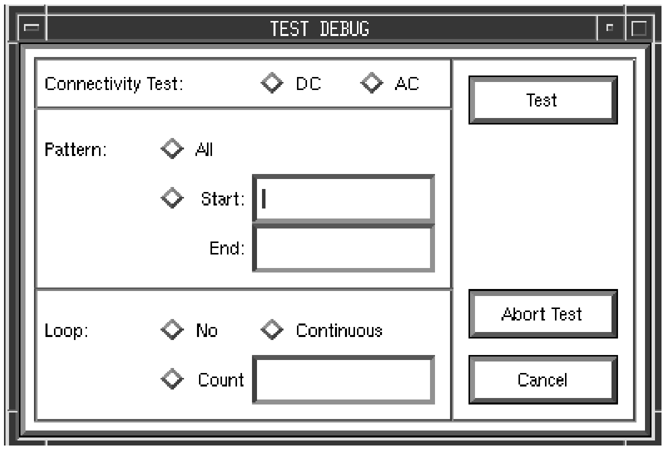


To issue a direct command, click in the Command field, enter the command and then press the **Return** key. `est` executes the command with output going to the main window. Clicking the Accept button repeats the command. The Clear button clears the command line. Clicking the Cancel button closes the window.

Connectivity test window

The connectivity test window invokes the connectivity tests. With this window, the user can select either the ac or dc test, the starting, ending, or all patterns, and the test looping parameters. Figure 67 shows the `est` connect window.

Figure 67 **`est` connectivity window**



To select a connectivity test, click on either the dc or ac button in the Connectivity Test panel.

In the Pattern panel, clicking the All button runs each test pattern. est creates the patterns on the fly based on the number of testable wires in the system. The user can also select the starting and ending patterns by clicking the button next to the start field. Enter the appropriate data in the Start and End fields. The Start and End options are normally used when debugging a system or board.

The Loop panel has three check buttons:

- No—Disables looping.
- Continuous—Enables continuously test looping.
- Count—Enables test looping a finite number of times. To set the number of times the test loops, click the Count button and enter the number of loops in the Count field.

To start the test, click the Test button; to stop it, click the Abort Test button.

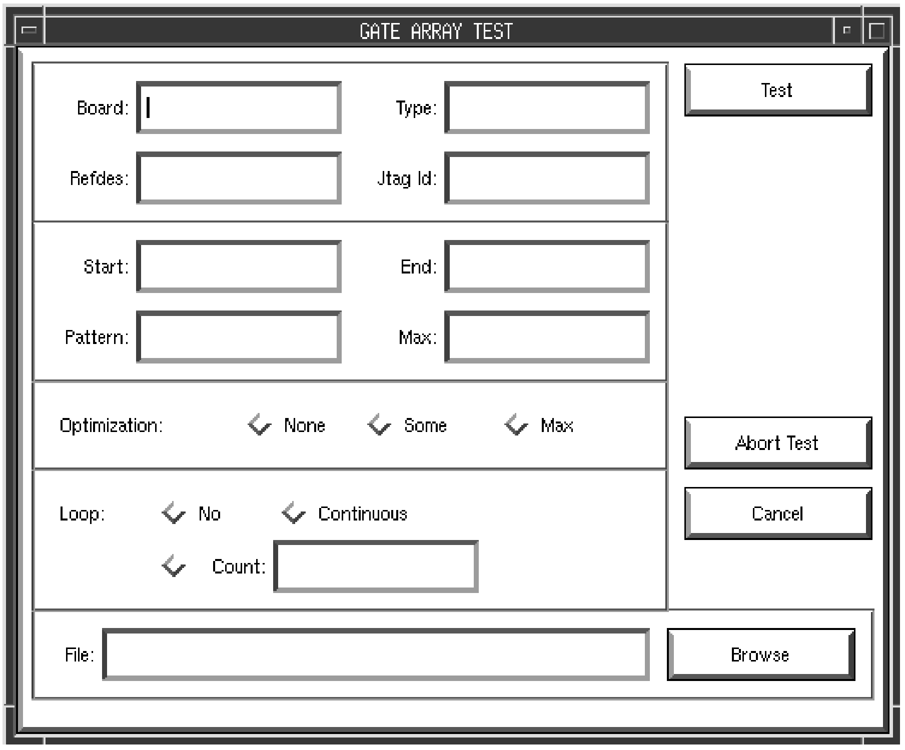
Clicking the Cancel button closes the connectivity window.

Gate array test window

The gate array test window provides a means to test all gate arrays in the Exemplar system. The window is simple to use.

Figure 68 shows the `est` gate array test window.

Figure 68 **`est` gate array test window**



In the top panel, enter the following data in the appropriate fields:

- Board—Sets the location of the gate array.
- Type—Sets the type of gate array.
- Refdes—Sets the reference designation of the gate array.
- Jtag—Sets the associated JTAG identification.

When more than one field is used, `est` picks what to test by ANDing the fields.

The next lower panel determines which and how many patterns are used in the gate array test. The test normally uses all patterns, but, for troubleshooting, you may set the starting and ending patterns, set the maximum number of patterns (a range of patterns), or set a single, custom pattern. Enter the following test pattern information in the appropriate fields:

- Start—Sets the starting pattern.
- End—Sets the ending pattern.
- Pattern—Sets a custom pattern.
- Max—Sets the range of patterns to be used in the gate array test. The default is to use all patterns.

In the next lower panel, click the appropriate test optimization buttons:

- None—No optimization.
- Some—Increased optimization.
- Max—Maximum test optimization.

The next lower panel controls the looping parameters:

- No—Disables looping. The test is only run once.
- Continuous—Enables continuous test looping. When running in continuous looping, the test is halted by clicking the Abort Test button.
- Count—Enables controlled looping. The number of loops is entered in the Count field.

The gate array test window may also be loaded with predefined parameters file. To load a file, click the Browse button and locate the appropriate file in the browse window.

Clicking the large buttons in the gate array test window has the following effect:

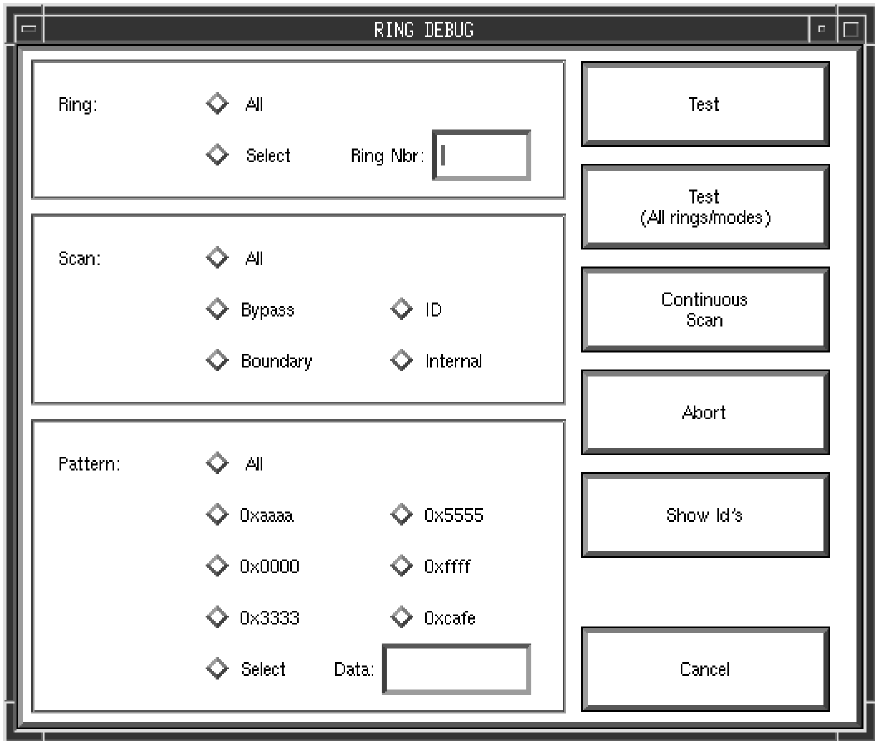
- Test—Starts the gate array test.
- Abort Test—Stops the test.
- Cancel—Closes the gate array test window.

Scan window

The scan window provides means of testing the system scan rings. Figure 69 shows the `est` scan window.

NOTE For more information on scan rings and modes, see the IEEE 1149.1 JTAG specification.

Figure 69 **`est` scan window**



The window has three panels: Ring, Scan, and Pattern.

Clicking the buttons in the Ring panel has the following effect:

- All—Tests all available rings in the system.
- Select—Allows the user to test a particular ring by entering the ring number to be tested in the Ring Nbr field.

Clicking the buttons in the Scan panel sets the scan paths. All scan modes can be selected or the test can be set up to test the individual pathways as follows:

- All—Tests all scan modes.
- Bypass—Test the bypass ring.
- ID—Tests JTAG identification ring.
- Boundary—Tests the ring boundary.
- Internal—Test the internal ring.

In the Pattern panel, clicking the All button causes the test to use all available patterns. Clicking the button next to a particular pattern causes the test to only use that pattern (plus any others that are checked at the same time). Clicking the Select button allows the user to specify the test pattern by entering it in the Data field.

Clicking the large buttons in the scan test window has the following effect:

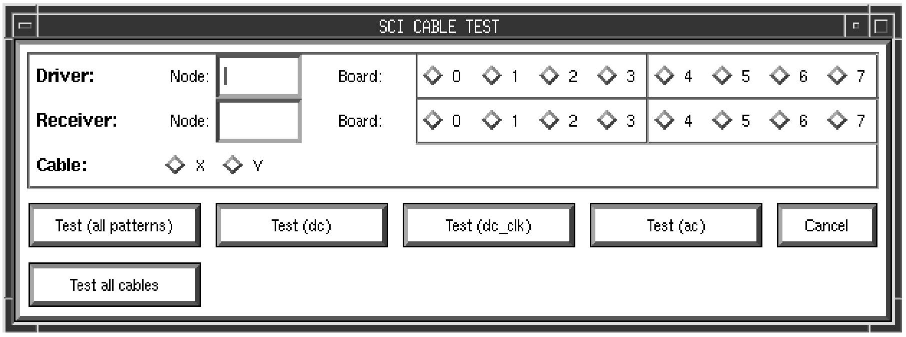
- Test—Starts the scan test. The rings, scan parameters and patterns selected in the scan window are invoked by this button.
- Test (All rings/modes)—Starts the scan test using all rings and patterns regardless of what is selected in the scan window.
- Continuous Scan—Places the scan test in continuous looping.
- Abort Test—Stops the test.
- Show Id's—Shows the JTAG IDs of all devices in the appropriate scan rings.
- Cancel—Closes the scan test window.

SCI cable test window

The SCI cable test window provides a means to test the cables that connect the scalable coherent interfaces between nodes. All cables are tested by default, but an individual cable can be tested using this window.

Figure 70 shows the `est` SCI cable test window.

Figure 70 **`est` SCI cable test window**



In the top panel are two rows of fields and buttons that determine source port (Driver) of the cable and the destination (Receiver). A third row selects either the X or Y cable. For both the Driver and Receiver select a node and EMB. Enter the desired node number in the node field. Click the interface number (0 through 8). Click either or both the X-ring cable or Y-ring cable.

The buttons in the lower portion of the window have the following effect:

- Test (all patterns)—Runs the SCI cable test using all test patterns.
- Test (dc)—Performs the continuity test on the cable.
- Test (dc_clk)—Performs the continuity test on the cable clock lines.
- Test (ac)—Performs dynamic test on the cable.
- Test all cables—performs full cable test suite on all interface cables.

CAUTION Before running the full cable test suite, refer to “SCI_all test” on page 246.

- Cancel—Closes the window.

Help

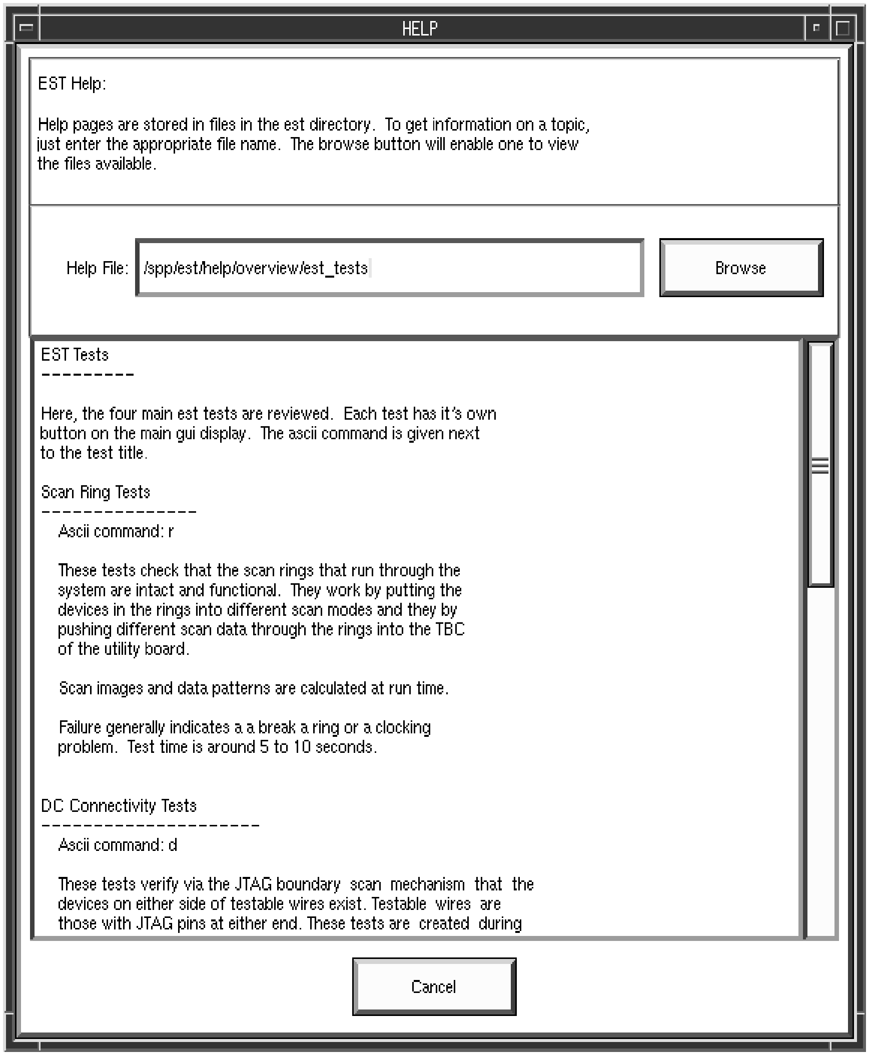
Clicking the Help button opens pop-up menu with five topic selections:

- Overview
- Commands
- GUI
- Input Files
- Options

Clicking on one of these options opens the Help window shown in Figure 71. This window is initially blank.

To open the topic of interest, click the Browser button. This opens the Help browser window shown in Figure 72. Double click on a topic listed in the browser.

Figure 71 **est Help window**



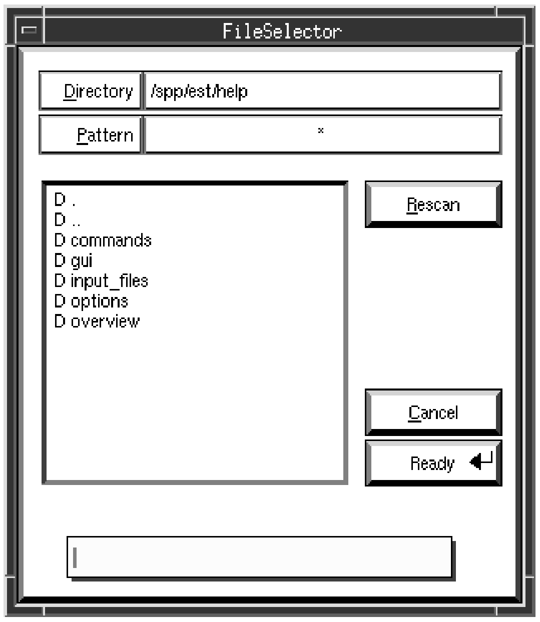
EST Help:

Help pages are stored in files in the est directory. To get information on a topic, just enter the appropriate file name. The browse button will enable one to view the files available.

Help File: /spp/est/help/overview/est_tests

```
EST Tests
---------

Here, the four main est tests are reviewed.  Each test has it's own
button on the main gui display.  The ascii command is given next
to the test title.

Scan Ring Tests
---------------
   Ascii command: r

   These tests check that the scan rings that run through the
   system are intact and functional.  They work by putting the
   devices in the rings into different scan modes and they by
   pushing different scan data through the rings into the TBC
   of the utility board.

   Scan images and data patterns are calculated at run time.

   Failure generally indicates a a break a ring or a clocking
   problem.  Test time is around 5 to 10 seconds.


DC Connectivity Tests
---------------------
   Ascii command: d

   These tests verify via the JTAG boundary  scan  mechanism  that  the
   devices on either side of testable wires exist. Testable  wires  are
   those with JTAG pins at either end. These tests are  created  during
```

Figure 72 **`est` Help browser window**



Running `est` from command line

The following is the command line usage for `est`:

```
est [-options] <node_number>
```

For example, to test node 0, enter:

```
% est 0
```

`est` reads configuration information from files stored in /spp/data (e.g node_0.cfg). These configuration files are automatically generated by `ccmd` each time the system is powered up. While `ccmd` is running, it prints its status to the console window. When database generation is complete and no errors are reported, it writes the necessary configuration files and `est` may be executed.

Table 83 shows `est` command line options.

Table 83 **`est` command line options**

Option	Description
`-v`	Print version and exit
`-f <filename>`	Run a given script file
`-l`	Do not generate a log file
`-o <filename>`	Redirect the log file to the given filename
`-x`	Open X windows GUI interface
`-y`	Say “yes” if asked to take over a locked ECUB
`-C`	Use old style configuration formats
`-V`	VT100 command menu
`-A`	Standalone; does not connect to node
`-B`	Do not build database
`-H`	Hardware mode on (default)
`-N`	Hardware mode off

Option	Description
`-P`	Do not let `est` handle the MIB power control
`-U`	<on|off> UTS support option
`-Y`	Force `est_config` to be run
`-Z`	Force `est_config` not to be run

Some examples of `est` usage are:

```
est -v
```

```
est -l -f my_script 0
```

```
est -o ./my_log_file 0
```

The est utility uses certain data and vector files located in the /spp/est directory.

Unless disabled or redirected, the `est` utility will generate a log file, est.log, and store it /spp/data/est.log. Any previous log file will be renamed to est.log.old

Example of output when `est` is started:

```
% est 0

Excalibur Scan Test     1.0.0.2  1998/11/25 10:32:58   Steven Terry
.........................
.....
General EST Tests:
c       ... compare id's to config file
r       ... scan ring test
d       ... board level dc tests
a       ... board level ac tests
g [options] [file] ...  gate array tests

Special Scan Tests:
b       ... bypass/id test
i       ... print id's found in design

EST Options:
F       ... set option & debug flags
q       ... quit nicely, ask first
qq      ... quit nicely, don't ask
Q       ... quit, not so nice
h       ... print this help message
v       ... print EST version info
!cmd    ... send the command to Unix (ex. "!ls patterns")
>>
```

Example output when using the `est -h` option:

```
% est -h

Excalibur Scan Test    1.0.0.2 1998/11/25 10:32:58   Steven Terry

usage: est [-options] [server] node [-cp port] [-sp port]

options:
        -h ... print this help message
        -v ... print the version of the program and exit

        -l ... turn OFF log file for this session
        -f <file> ... get commands from <file>
        -o <file> ... redirect log file to <file>

        -x ... X windows gui interface
        -y ... say "yes" if asked to take over a locked ecub
        -C ... use old style config file formats
        -V ... vt100 command menu

        -A ... stand alone; does not connect to node
        -B ... do not build database
        -H ... harware mode on (default)
        -N ... hardware mode off
        -U <on|off> ... UTS support option
        -Y ... force est_config to be run
        -Z ... force est_config not to be run
        -P ... do not let est handle the midplane's power control

ports:
        -cp ... client port
        -sp ... server port
```

AC Connectivity test

The ac Connectivity test format is:

`a [-s -p #]`

Table 84 shows the options for the this test.

Table 84 **AC Connectivity test options**

Option	Description
`-s`	Step mode (for debug purposes).
`-p <number>`	Run pattern number only.

Bypass test

The Bypass test format is:

```
b
```

The Bypass test places the scan ring hardware into bypass mode.

DC Connectivity test

DC Connectivity test format is:

```
d [-s -p #]
```

Table 85 shows the options for the this test.

Table 85 **Dc Connectivity test options**

Option	Description
`-s`	Step mode (for debug purposes).
`-p <number>`	Run pattern number only.

Gate Array test

The Gate Array test format is:

```
g [options] [pattern file]
```

Table 86 shows the options for the this test.

Table 86 **Gate Array test options**

Option	Description
`-r <refdes>`	Test arrays with matching reference designator value.
`-b <board>`	Test arrays on given board. <board> may either be a number or a name.
`-j <jtag_id>`	Test arrays matching a jtag_id.
`-t <type>`	Test an array type (For example, ERAC).
`-s <number>`	Start with a given pattern number.
`-e <number>`	End on a certain pattern number.
`-m <number>`	Run a maximum of <number> patterns per file.
`-o <number>`	Optimization level (0, 1, or 2) Two is the most optimized and is the default.

By default, the `g` command tests all arrays. When the `-r`, `-b`, `-j`, or `-t` options are used, only arrays that meet all criteria are tested.

Gate array tests use test vectors that have been pregenerated for the certain arrays (each array has multiple files associated with it). The `-s`, `-e`, and `-o` options can be used to limit the number of patterns that are run for each pattern file. The default is to run all patterns in each file. While it may take more time to run all patterns, using options to limit the number of patterns may result in a significant loss of test coverage.

The `-o` option controls how much parallelism takes place. There are three levels:

- 0—No optimization.
- 1—Same parts on the same ring are tested together.
- 2—Identical rings are tested in parallel.

If an error is encountered during parallel testing, the following message may appear:

```
*** errors found - must switch to serial testing ***
```

When an error occurs, parallel scans into the scan hardware may result in bus conflicts on TDO pins. Therefore, `est` automatically stops using parallel scans when errors happen.

SCI test

The `sci` utility tests the Coherent Toroidal Interface (CTI) cables between nodes. The term SCI (Scalable Coherent Interface) is often used in place of the term CTI; the terms are interchangeable.

The usage of `sci` is as follows:

```
sci [driver] [receiver] ring test
```

where:

[driver]	Refers to the node and memory board to which the CTI cable is connected and from which the test data originates.
[receiver]	Refers to the node and memory board to which the other end of the CTI cable is connected and receives test data.
ring	Refers to the CTI ring associated with the cable, either x or y.
test	Refers to the specific test: dc, dc_clk, ac. With the dc test, the clock from the receiver node is used. The dc_clk test derives its clock from the cable.

The following is an example of `sci` usage:

```
% sci 0 mb01 1 mb01 y dc_clk
```

This command runs the `dc_clk` test on the cable connected between node 0, memory board 1 (driver) and node 1, memory board 1 receiver.

SCI_all test

The `sci_all` utility tests all SCI cables in a complex.

The usage of `sci_all` is as follows:

```
sci_all [test]
```

where:

test	Refers to the specific test: dc, dc_clk, ac. With the dc test, the clock from the receiver node is used. The dc_clk test derives its clock from the cable.

If all cables are not connected or there is an unusual cable configuration, `sci_all` will not work until the cable configuration files in /spp/est/sci_tests have been updated:

CAUTION Use extreme care when modifying any `est` data files; damage to the system may result. If you are not sure what to do, seek help.

To run `est` with modified data files, create a local directory and copy the files to it. For the case of `sci_all`, create a local ./sci_tests directory and then copy the sci cable files (shown above) to it. Edit the files and then run `est`. `est` looks into the ./sci_tests directory for the files before the /spp/est/sci_tests directory.

JTAG Identification test

The JTAG Identification test prints all JTAG IDs. the format is:

```
i
```

Margin commands

The Margin command for clocks and power format is:

```
m [-c | -p [supply] [value]]
```

The `-c` option specifies the clock, and the `-p` option specifies power. These two options can not be used together; use either `-c` or `-p`.

When a value is not supplied, the current states are displayed.

By itself, **`m`** shows all margins.

The following are examples of margin command options:

- **`-c high`**—Displays the upper clock limit.
- **`-p 1 nom`**—Sets the supply 1 margin to nominal.

There are four power supplies, 1 through 4.

Table 87 shows the valid values for clock and power.

Table 87 Valid values for clock and power supplies

Clock	Power
up or high	up or high
nom	nom
ext	low

est miscellaneous commands

This section gives the following useful commands entered at the est prompt:

- **`ms`**—Puts all the scan hardware into a safe state.
- **`q`**—Quits, but asks the user first.
- **`qq`**—Quits without asking.
- **`script <file>`**—Runs a file containing est commands.
- **`v`**—Prints version information.
- **`F`**—Opens the flags submenu.
- **`t`**—Prints current time and date.

est run time option commands

`est` provides commands that update the main option settings that control the run time operation of est tests. Each command uses one command line argument, on or off. Table 88 lists these commands.

NOTE When in GUI mode, all of the above choices are available through the options and details pull-down menus.

Table 88 **`est` runtime option commands**

Command	Description	Default argument
`log_file`	Turn on/off writing to the log file.	On
`stop_on_error`	Stops the test when an error is detected.	On
`limit_patterns`	Runs a limited set of patterns when testing arrays. This runs faster, but reduces coverage.	Off
`limit_errors`	Limits to 10 the max number of errors that will get printed. The total error count is still printed.	On
`pass_fail`	When enabled, it prints the pass or fail status of each test pattern.	Off
`test_file_msgs`	Debug option that enables printing of gate array pattern file information.	Off
`show_scan_instr`	Show scan instructions when running tests.	Off
`show_scan_data`	Show scan data when running tests.	Off
`show_sdp`	Show SDP ethernet information when running `est`.	Off
`power_control`	Affects whether or not `est` takes control of the power-down signal in the utility system. Turning this feature off could result in unexpected power shutdowns, but that may be needed for some special debug efforts.	On

est command flags and options

There are a number of flags or options that operate on and enhance the `est` commands. Some of these flags and options perform the same functions as the run time option commands.

To set these options, enter **`F`** at the `est` prompt. This invokes the flags submenu. To exit, press **return** at the flags prompt. This returns the main `est` prompt.

Some of the more useful options are:

- `l`—Limits the number of internal array patterns executed by the `g` command. This has the affect of decreasing coverage to approximately 90 percent.
- `s`—Stops testing when an error is detected.
- `A [number]`—Limits the number of ac connectivity tests.

  Setting a limit of zero or less results in all patterns being used.
- `D [number]`—Limits the number of dc connectivity tests.
- `E`—Shows SDP packets transferred across the ethernet.
- `P`—Controls whether or not the pass/fail status of individual patterns are displayed.

Script files

There are two ways of running script files: from the command line (**`-f <filename>`**) or from the `est` prompt. From the command line, `est` executes the instructions listed and when finished, displays the `est` prompt. To cause `est` to quit when the script is finished, put **`q`** at the end of the script file.

The script command reads `est` commands from an ASCII file and runs those commands. The following rules apply to the file:

- The file must have only one command per line.
- Command syntax must be the same as entered at the est prompt.
- Comments lines must start with a # sign.

An example file might contain the following lines:

```
# check the rings
r
# show pattern pass/fail steps
F P
#limit dc testing to 3 patterns
F D 3
#do dc testing
d
q
```

12 Utilities

This chapter details most of the diagnostic utilities which include:

- `address_decode`
- `arrm`
- `autoreset`
- `console`
- `consolebar`
- `cpu_hang`
- `dcm`
- `dfdutil`
- `dump_rdrs`
- `fwcp`
- `fw_init` and `fw_install`
- `get_node_info`
- `hard_logger`
- `lcd`
- `load_eprom`
- `opie`
- `pciromldr`
- `pim_dumper`
- `set_complex`
- `soft_decode`
- `sppconsole`

- `tc_init`
- `tc_ioutil`
- `tc_show_struct`
- Version utilities
  - `diag_version`
  - `flash_info`
  - `ver`
- Event processing
  - `event_logger`
  - `log_event`
- Miscellaneous tools
  - `fix_boot_sector`
  - `kill_by_name`

address decode

`address_decode` decodes 40-bit virtual address using the current memory configuration to resolve the physical node, SMAC, row, bus, and bank.

It has the following format:

```
address_decode <40-bit address in hex>
```

In order to determine the current memory configuration, `address_decode` invokes some `sppdsh` commands to read certain CSR values so that it can take into account the board mapping, row mapping, interleave values, and DIMM sizes present in the system. Consequently, it must be run on a SSP that can access the node via `sppdsh`.

`address_decode` reports an error if the address entered does not exist.

Example of `address_decode` use:

```
% address_decode 0x010f000020
```

In this example, the address decodes the following

- Node ID: 0
- SMAC: 7
- Bus: 3
- Row: 2
- Bank: 1

Example of address_decode error:

```
% address_decode 0x01ff000020

Detected non-existent row!

Decode failure
```

AutoRaid recovery map (arrm)

NOTE `arrm` has been replaced by the ARDIAG test module that is invoked by `tc_ioutil` on the SSP. See “opie” on page 292 for a description of how to use this utility.

autoreset

`autoreset` enables or disables automatic reset after an error. It determines the behavior of `ccmd` when it first encounters an error condition. If an error exists on the system when `autoreset` is enabled, then `ccmd` does not reset the system. Subsequent errors will force `ccmd` to reset the system.

autoreset has the following format:

```
autoreset [<chk> | <complex_name <on|off>]
```

where:

on—Enables automatic reset after an error on the complex specified

`off` —Disables automatic reset after an error on the complex specified

`chk`—Prints current state of the automatic reset indicators.

`autoreset` allows the user to specify whether `ccmd` should automatically reset a node after a hard error and after the hard logger error analysis software has run. The automatic reset occurs if a /spp/data/.ccmd_reset file does not exist.

As an example, the output of the `chk` option on complex_name hw2a looks like:

```
Autoreset for hw2a is enabled.
```

or

```
Autoreset for hw2a is disabled.
```

console

`console` is a console server client program that manipulates console terminals remotely or by polling running conserver(8) daemons for status information.

`console` queries the user for the root password before granting interactive access to a console (on a non-trusted system), because such a session may provide single-user access.

In the non-interactive mode console outputs only the requested information.

`console` has the following format:

```
console [-rv] [-AFSafs] [-e esc] [-M server] host
```

```
console [-dDqQ] [-v] [-M server] host
```

```
console [-v] [-huVwx]
```

Table 89 shows console options.

Table 89 **`console` options**

`-a`	Access a console with a two-way connection (this is the default).
`-d`	Display daemon versions. The console client connects to each server to request its version information. The uppercase variant of this option only requests the primary server's version.
`-e esc`	Set the initial two-character escape sequence to those represented by **esc**. Any of the forms output by the `cat -v` option are accepted. The default value is **^Ec**.
`-f`	Same as `-a` except it forces any existing connection into spy mode.
`-h`	Display a brief help message.
`-M server`	Poll server as the primary server rather than the hard coded default (console.cc.purdue.edu).

`-q`	Requests that the server daemon quit (shutdown). A password is sent in the protocol stream. If none is required for the local host to shutdown the server, press **return**. The uppercase variant of this command acts only on the primary server.
`-r`	Request a raw connection to the group control virtual console. This is only useful for learning the interactive sequence.
`-s`	Requests a read-only (spy mode) connection. In this mode, all the escape sequences (below) work or report errors, but all other keyboard input is ignored.
`-V`	Output the version of the console client program.
`-v`	Invoke verbose mode when building the connection(s). Use this option in combination with any of “show” (`-u`, `-w`, or `-x`) options for added benefit.
`-u`	Show a list of consoles and the users on each.
`-w`	Show a list of all connections to all consoles.
`-x`	Show a list of consoles and devices.

The -A, -F, or -S options have the same effect as their lower case variants. In addition, they each request the last 20 lines of the console output after making the connection. Any default (`-a`) connection is dropped to spy mode if someone else is attached.

Escape Sequences

Connections can be controlled by a two-character escape sequence, followed by a command. The default escape sequence is **control-E c** (octal 005 143). Table 90 shows describes each escape sequence command.

Table 90 **`console`** escape sequence commands

`a`	switch to attach mode
`c`	toggle flow control (don't do this)
`d`	down the current console

`e`	change the escape sequence to the next two characters
`f`	force a switch to attach mode
`l1`	send a 3 second serial line break (might halt a Sun)
`o`	reopen the line to clear errors (silo overflows)
`r`	replay the last 20 lines of output
`s`	switch to spy mode
`u`	show other users on this port
`v`	show the version of the group server
`w`	who is using this console
`x`	examine this group's devices and modes.
`z`	suspend this connection
`?`	display list of commands
`^I`	toggle tab expansion
`^J`	continue, ignore the escape sequence
`^R`	replay the last line only
`\.`	disconnect
`;`	provide a new login or shift to a new console
`+(-)`	be more (less) free with new lines

If any other character is hit after the escape sequence, all three characters are discarded.

Note that a line break or a down command can only be sent from a full two-way attachment.

To send the escape sequence through the connection, the outer escape sequence must be redefined.

In the `-u` output, the login <none> indicates no someone is viewing that console, the login <spies> indicates that no one has a full two-way attachment. When no one is attached to a console, its output is cloned to the `stdout` of the server process.

Example of console

Using the `-u` option produces output similar to the following:

```
% console -u

l18 l l.
dumb   up   <none>
expert   up   ksb@mentor
tyro  up  <spies>
mentor up  <none>
sage up fine@cis
```

`<none>` indicates no one is viewing “dumb” or “mentor.”

`<spies>` indicates only read-only connections exist for tyro; other login@host entries are the currently attached “sage” and “expert.”

Using the `-w` options produces the following output:

```
% console -w

l l l.
ksb@extra on expert Fri Feb 15 16:40:36 1991
file@cis on sage Thu Feb 14  1:04:10 1991
dmr@alice spy tyro Thu Feb  7 10:09:59 1991
```

The following -e option requests a connection to the host ``lv426'' with the escape characters set to “escape one.”

```
% console -e \*(lq^[1\*(rq lv426
```

consolebar

The `consolebar` utility is an X application that provides a simple interface capable of starting console windows to all V2500/V2600 nodes configured on the SSP. It has the following format:

```
consolebar [-display displayname]
```

`consolebar` retrieves the list of configured nodes and displays the node IDs, grouped by complex. When the push-button for a node is pressed, an xterm is started and the `sppconsole` program is run against the specified node.

To start `consolebar` from the SSP root menu, select the `consolebar` menu item.

To start from a shell (local or remote), ensure that your DISPLAY environment variable is set appropriately before starting `consolebar`.

For example:

```
$ DISPLAY=myws:0; export DISPLAY     (sh/ksh/sppdsh)
% setenv DISPLAY myws:0              (csh/tcsh)
```

As another example, use the -display start-up option:

```
# consolebar  -display myws:0
```

NOTE For shells run from the SSP desktop, the DISPLAY variable is set (at shell start-up) to the local SSP display.

cpu_hang

`cpu_hang` can be used to identify the failing CPU in a system that has completely hung due to a malfunctioning CPU. The assumptions before running this script are:

- The system is completely hung.
- A processor is at fault.
- The internal scan ring still works.

The scan information retrieved is written to a log file in the /tmp directory called cpu_hang.log. If this file already exists, the new data is appended to it. Each time the log is updated a timestamp is inserted.

The script also writes the internal scan state of the node to a log file called scan_dump.log in the /tmp directory. If this file already exists, it is archived to scan_dump.log.old in the same directory.

cpu_hang uses the following format:

```
cpu_hang -h <node] >
```

where:

The `-h` prints a description of the methods used to isolate a defective CPU, and `<node>` is the node number (defaults to 0).

Fault isolation methods

`cpu_hang` uses three methods to try to identify the processor that first hung. None of these methods is foolproof, because all of the scenarios they try to identify can happen in a working system under the right conditions. These methods do, however, provide a good basis for finding the processor at fault.

Outstanding Coherency requests

In order to maintain data coherency between processors, packets are transmitted among processors to control sharing of data. If a processor hangs, its queue of incoming coherency requests may become backed up. This can often be a tell-tale sign identifying which processor originally hung. However, keep in mind that on a busy system, occasionally the queue backs up even under normal circumstances.

Outstanding read/write short requests

When a write short or read short request is sent to a processor, it is required to complete that request before responding to any other requests. If the processor hangs while a write/read short request is outstanding, it leaves evidence of this in the r_csr_busy scan field.

Revision 2.0 PCXW timeout condition

Revision 2.0 of the processor has a bug that prevents a processor that has timed out from executing the HPMC handler. Instead, after a timeout has occurred, the processor hang. However, `cpu_hang` can not determine the actual revision of the processor. The user should determine which, if any, processor boards have revision 2.0 PCXW's installed. This bug has been fixed in revision 2.1 of PCXW.

Example using cpu_hang

To attempt to isolate the failing processor in a hung system the following command could be used:

```
cpu_hang 0
```

The following is an example of what is printed to the screen. Note that the detailed scan information does not print to the screen, but is available in the log file in /tmp/cpu_hang.log.

```
Turning off clocks to Node 0 ...
Scanning SPAC's on Node 0 for pertinent data ...
Evaluating coherency queue on each CPU ...
      CPU(s) potentially responsible for hang: PB1L_A
Evaluating outstanding TLB count on each CPU ...
Evaluating outstanding write/read short status on each CPU ...
Evaluating timeout status on each CPU ...
```

In this example, the coherency queue was full on processor PB1L_A, therefore, reporting PB1L_A as the possible originator of the hang.

dcm

`dcm` dumps the boot configuration map information for the specified node. There are two main reporting modes; one for general hardware configuration and one for the DIMM type.

The general hardware mode reports processors, ASICs, and memory size information. The DIMM type mode provides pass/fail tests for specific DIMM types, and a general DIMM type report option.

`dcm` uses following format:

```
dcm [-d <80|88|all>] <node id> <node id> ...
```

`-d 80` checks to see if only 80-bit DIMMs are installed.

`-d 88` checks to see if only 88-bit DIMMs are installed.

`-d all` dumps the status of all installed DIMMs, 80- or 88-bit.

This option returns an exit code: a zero value indicates dump was successful and a one value indicates the dump failed.

`<node id>` may be a node number or IP name.

When invoked as **`dcm <node id>`**, `dcm` returns 0 and prints a table with the following format for a node with eight processors, eight SPACs, one SIOB, and EWMBs half-populated with 128-Mbyte DIMMs:

Output table using `dcm <node_id>`

```
Acquiring Boot Configuration Map...
         Stingray Configuration Map Dump:        Node:  0  (hw2a-0000)
         =============================================================
                 VERSION: 1.0     compiled: 1998/12/16 18:35:00
         CheckSum:0xf407a073
         Boot Config Map Size:164 words
         POST Revision:1.0
CPUs (Rev, ICache, DCache Size in MegaBytes)
============================================
PB0L_A PASS (2.0, 0.50, 1.00)          PB0L_B EMPTY
PB0R_A EMPTY                           PB0R_B EMPTY
PB1R_A PASS (2.0, 0.50, 1.00)          PB1R_B EMPTY
PB1L_A EMPTY                           PB1L_B EMPTY
PB2L_A PASS (2.0, 0.50, 1.00)          PB2L_B EMPTY
PB2R_A EMPTY                           PB2R_B EMPTY
PB3R_A PASS (2.0, 0.50, 1.00)          PB3R_B EMPTY
PB3L_A PASS (2.0, 0.50, 1.00)          PB3L_B PASS (2.0, 0.50, 1.00)
PB4L_A PASS (2.0, 0.50, 1.00)          PB4L_B EMPTY
PB4R_A EMPTY                           PB4R_B EMPTY
PB5R_A PASS (2.0, 0.50, 1.00)          PB5R_B EMPTY
PB5L_A EMPTY                           PB5L_B EMPTY
PB6L_A PASS (2.0, 0.50, 1.00)          PB6L_B EMPTY
PB6R_A EMPTY                           PB6R_B EMPTY
PB7R_A PASS (2.0, 0.50, 1.00)          PB7R_B EMPTY
PB7L_A PASS (2.0, 0.50, 1.00)          PB7L_B PASS (2.0, 0.50, 1.00)
SPACs
=====
0L    -  PASS
P1R    -  PASS
P2L    -  PASS
P3R    -  PASS
P4L    -  PASS
P5R    -  PASS
P6L    -  PASS
P7R    -  PASS
SAGAs
=====
IOLF_B    -  PASS
IOLR_B    -  PASS
IORR_B    -  EMPTY
IORF_B    -  PASS
IOLF_A    -  PASS
IOLR_A    -  PASS
IORR_A    -  EMPTY
IORF_A    -  PASS
SMACs
=====
MB0L_M    -  PASS
MB1L_M    -  PASS
MB2R_M    -  EMPTY
MB3R_M    -  EMPTY
MB4L_M    -  EMPTY
MB5L_M    -  EMPTY
MB6R_M    -  EMPTY
MB7R_M    -  EMPTY
STACs
=====
MB0L_T    -  DECONFIG
MB2R_T    -  EMPTY
MB3R_T    -  EMPTY
MB4L_T    -  EMPTY
```

```
MB5L_T    -  EMPTY
MB6R_T    -  EMPTY
MB7R_T    -  EMPTY
Memory:
=======
    Physical: L=128MB, M=64MB, S=16MB      Logical: l=128MB, m=64MB, s=16MB
    (If logical memory not specified, then it matches physical memory size)

                 * = Software Deconfigured      - = Not In Use
EWMB0:
======
EWMB0: Q0B0 S/S         Q1B4 -/-         Q2B0 -/-         Q3B4 -/-
EWMB0: Q0B1 S/S         Q1B5 -/-         Q2B1 -/-         Q3B5 -/-
EWMB0: Q0B2 S/S         Q1B6 -/-         Q2B2 -/-         Q3B6 -/-
EWMB0: Q0B3 S/S         Q1B7 -/-         Q2B3 -/-         Q3B7 -/-
EWMB1:
======
EWMB1: Q0B0 S/S         Q1B4 -/-         Q2B0 -/-         Q3B4 -/-
EWMB1: Q0B1 S/S         Q1B5 -/-         Q2B1 -/-         Q3B5 -/-
EWMB1: Q0B2 S/S         Q1B6 -/-         Q2B2 -/-         Q3B6 -/-
EWMB1: Q0B3 S/S         Q1B7 -/-         Q2B3 -/-         Q3B7 -/-
```

When invoked with **`dcm -d 80 <node id>`**, `dcm` returns 0 if all installed DIMMs are 80-bit single-node DIMMs. `dcm` returns a 1 if one or more 88-bit multinode DIMMs are detected.

When invoked with **`dcm -d 88 <node id>`**, `dcm` returns 0 if all installed DIMMs are 88-bit single-node DIMMs. `dcm` returns a 1 if one or more 80-bit multinode DIMMs are detected.

When invoked with **`dcm -d all <node id>`**, `dcm` returns 0 and prints a table with the following format for a node with two EWMBs installed that were half-populated with 88-bit DIMMs:

Output table using `dcm -d all <node_id>`

```
Stingray Configuration Map DIMM Info:   Node:  0(hw2b-0000)
      ============================================================
    VERSION: 0.8.0.1 compiled: 1998/10/23 14:34:01
Memory Type:
============
      Physical: 88=Multi node 88-bit DIMM, 80=Single node 80-bit DIMM
 (Only physical DIMM type is reported.)
      * = Software Deconfigured     - = Not In Use
EWMB0:
======
EWMB0: Q0B0 88/88 Q1B4 88/88   Q2B0 -/-     Q3B4 -/-
EWMB0: Q0B1 88/88 Q1B5 88/88   Q2B1 -/-     Q3B5 -/-
EWMB0: Q0B2 88/88 Q1B6 88/88   Q2B2 -/-     Q3B6 -/-
EWMB0: Q0B3 88/88 Q1B7 88/88   Q2B3 -/-     Q3B7 -/-
EWMB1:
======
EWMB1: Q0B0 88/88 Q1B4 88/88   Q2B0 -/-     Q3B4 -/-
EWMB1: Q0B1 88/88 Q1B5 88/88   Q2B1 -/-     Q3B5 -/-
EWMB1: Q0B2 88/88 Q1B6 88/88   Q2B2 -/-     Q3B6 -/-
EWMB1: Q0B3 88/88 Q1B7 88/88   Q2B3 -/-     Q3B7 -/-
```

`dcm` returns a negative number for all scan-related failures.

dfdutil

`dfdutil` is a stand-alone off-line utility that downloads firmware to SCSI devices including disks, arrays, and fibrechannel devices such as SCSI MUX and fibrechannel arrays.

The firmware image(s) are contained in a Logical Interchange Format (LIF) volume on the SSP at /spp/firmware/DFDUTIL.LIF. The raw (usually binary) firmware image of one or more devices is contained in the LIF filesystem. `dfdutil` reads this file when it initializes and examines header of each file for a standard firmware header. The firmware header is required for download capability. Since most HP firmware distributions are already packaged in this format, the procedure for putting a raw binary firmware image into the proper format for `dfdutil` is not covered in this document.

NOTE DFDUTIL.LIF must have world read permissions to be accessed by `dfdutil`.

To load and run `dfdutil`, enter the following command at the SSP prompt:

```
tc_ioutil <node id or all> dfdutil.fw
```

To run `dfdutil` on a specific complex, enter the following command at the SSP prompt:

```
tc_ioutil <complex name> <node number or "all"> dfdutil.fw
```

This command issues a system reset. The test controller bootstrap loads the executable image, dfdutil.fw, from the SSP file /spp/firmware/ dfdutil.fw and executes it.

Once started, `dfdutil` loads the file DFDUTIL.LIF from the SSP and scans all SCSI and Fibrechannel busses on the system.

Example of dfdutil output when loading

```
*****************************************************************************
***                              DFDUTIL                                  ***
***                                                                       ***
***              (C) Copyright Hewlett-Packard Co. 1998                   ***
***                         All Rights Reserved                           ***
***                                                                       ***
***  This program may only be used by HP support personneland             ***
***  those customers with the appropriate Class license or                ***
***  Node license for systems specified by the license.  HP               ***
***  shall not be liable for any damages resulting from misuse            ***
***  or unauthorized use of this program.  This program                   ***
***  remains the property of HP.                                          ***
***                                                                       ***
***                           Version 4.4.0.0                             ***
***                                                                       ***
********************************************************************

Please wait while I check all the firmware files...
Please wait while I check all the firmware files...
Opening file /spp/firmware/DFDUTIL.LIF
DNS server   : 15.99.111.99 (f636f63)
NFS ip       : 15.99.111.99 (f636f63)
Mounting    : /spp/ - mounted.

  Indx Path               Product ID                Bus     Size    Rev
  ---- ------------------ ------------------------- ------- ------  ------
  0    5/1:0.6.0          SEAGATE  ST15150W         SCSI    4095    HP10
  1    7/1.8.0.255.0.0.0  HP       HPA3308          FC      0       d373
  2    7/1.8.0.0.0.14.0   DGC      DISK             FCMUX   4006    0860
  2.0  ^array^            SEAGATE  ST15150N         SCSI    4024    HP02
Legend:
Indx = Index number used for referencing the device
Rev  = Firmware Revision of the device
Note:  Due to different calculation mthods used, the size
       of the device shown is only a rough approximation.
        File name        Intended Product ID             Rev.   Size
        ---------------- -----------------------------   ------ -----
        R18CUDA9        SEAGATE         ST19171W        0018   257888
        R23CUDA9        SEAGATE         ST19171W        0023   257888
        ST34371W84      SEAGATE         ST34371W        0484   276512
        ST39173WD5      SEAGATE         ST39173W        HP05   303360
        ST39173WD8      SEAGATE         ST39173W        HP08   303360
        ST15150W23      SEAGATE         ST15150W        0023   261632
        ST15150W22      SEAGATE         ST15150W        0022   261632
        DVD316          PIONEER         DVD303          0016   135168
        DVD317          PIONEER         DVD303          0016   135168
        DVD316H         PIONEER         DVD303          0016   330679
        DVD317H         PIONEER         DVD303          0016   331354
        MUX373          HP              FC-SCSI_MUX     d373   2162516
        MUX40_1         HP              FC-SCSI_MUX     40_1   2162516

Legend:
File name           = name of the firmware file
Intended Product ID = firmware file's intended product name
Rev.                = firmware Revision of the firmware file
Size                = exact byte size of the firmware image
DFDUTIL>
```

The output above shows the two main data structures used by this program: the bootable device table and the LIF file table. It ends with the DFDUTIL> prompt for a built-in command line interpreter. The bootable device table shows the drives and array controllers to which the operator can send a download, and the LIF file table shows the firmware files that are available to be downloaded.

`dfdutil` bootable device table

The descriptions of the fields in the bootable device table are as follows:

- Index—Specifies in the `DOWNLOAD` command which device is used to download firmware to. FRUs in an array (the individual drives) are shown with a subindex (X.Y), where X is the array controller, and Y is the index of the physical drive. Array drives must be specified with the X.Y notation.
- Path—Specifies the hardware path to the drive. There are two possibilities: fibrechannel or direct attach SCSI. In the case of directly attached SCSI, the path is formatted as `a/b:c.d.e`. Each letter in the path is defined as follows:
  - a—SAGA number
  - b—slot number
  - c—path level (always 0)
  - d—target ID
  - e—LUN number

  In the case of Fibrechannel bus, there are two possibilities: direct attach fibrechannel or fibrechannel SCSI MUX. The path of the direct attach fibrechannel is formatted as `a/b.c.d.255.e.f.g` with the definition of the letters as follows:

  - a—SAGA number
  - b—slot number
  - c—path level (always 0)
  - d—always 8 for FC storage
  - e—upper 4 bits of loop address
  - f—lower 4 bits of loop address

- g—LUN number

If the device is attached to an FC SCSI MUX, the path is formatted as `a/b.c.d.e.f.g.h`. with the letter definitions as follows:

- a—SAGA number
- b—slot number
- c—path level (always 0)
- d—always 8 for fibrechannel storage
- e—loop address (fibrechannel loop address of the MUX to which this device is attached)
- f—backside SCSI bus number
- g—target number
- h—LUN number

NOTE Array drives (FRUs) are not listed with an absolute hardware path since they are not directly accessible from the SCSI bus. They are listed with the special token "^array^" in the path field.

- Vendor ID—Specifies ID read from the drive in the inquiry data.
- Product ID—Specifies the ID strings read from the device using the `INQUIRY` command.
- Bus Type—Specifies how the devices is connected to the node. Either direct attach SCSI, direct attach Fibre Channel Loop on the back of a FC SCSI Mux.
- Size—Specifies the approximate unformatted capacity in megabytes for a disk drive or the approximate formatted capacity for an element of an array.
- Revision level—Specifies the revision reported by the device in the inquiry data.

`dfdutil` LIF file table

The descriptions of the fields in the LIF file table are as follows:

- Filename—Specifies the name of the file in the LIF volume. The operator specifies this name when issuing download commands to the devices.

- Intended Product ID—Specifies the concatenation of the vendor ID and the product ID for the drive or array for which this file is intended. (If this data does not match the vendor ID and product ID of a drive in the bootable device table, a download will not be allowed.)
- Rev.—Specifies the firmware revision of the file. This is also setup during firmware packaging.
- Size—This is the size in bytes of the file not including the file header.

dfdutil commands

The `DFDUTIL>` prompt indicates that the built-in command line interpreter is waiting for a command.

The commands available to this command line interpreter are:

- `DOWNLOAD <filename> <index>`
- `DISPMAP <disk index>`
- `DISPDILES`
- `LS`
- `RESET`
- `UTILINFO`
- `NODE <node number>`
- `HELP` <command>

`DOWNLOAD` command

Use the `DOWNLOAD` command to download firmware to a particular device. `DOWNLOAD` transfers the contents of a particular firmware file to a device. It prompts the user for any arguments that were not specified on the command line.

NOTE Once the download begins, do not interrupt the process, or the devices to which the firmware is being loaded could be rendered useless.

The syntax for the `DOWNLOAD` command is:

```
DISMAP <filename> <disk index>
```

`filename` must match one of the file names in the LIF file table, and `index` must match one of the index numbers in the bootable device list (displayed when the program starts). If the file specified does not have the same vendor and product ID as the device whose index number is specified, an error message will be issued to the operator and the download will be aborted.

As an example, to download firmware to the SCSI MUX HPA3308 (the mux controller firmware), enter the following command line:

```
DFDUTIL> download MUX1 0
```

`dfdutil` prompts the user for confirmation since FC-SCSI_MUX does not match the product ID of the device, HPA3308.

To download to FRUs in an array, enter the following command line:

```
DFDUTIL> download firmware_file_name 2.1
```

DISPMAP command

The DISPMAP command displays a list of all devices connected to the system. The information displayed includes:

- Index number
- Product identification
- Device size
- Index number
- Firmware revision

The syntax for the this command is:

```
DISPMAP <disk index>
```

The user may enter the index number of a single device; using no index number causes `DISPMAP` to list all devices.

This command will display the bootable device table displayed when `dfdutil` is started. If the optional argument [index] is specified, then only the information for the given index number will be displayed, not the entire table. This display may not reflect any downloads that may have been done since the program was started.

The following two examples show output using no index number and one index number, respectively.

Example output of `dfdutil DISPMAP` command with no index number

```
Indx Path           Product ID                     Bus    Size   Rev
---- -------------- ------------------------------ ------ ------ ----
0    2/0/1.2.0      QUANTUMLP270S disc drive       SCSI   258 MB 5909
1    2/0/1.6.0      QUANTUMLP270S disc drive       SCSI   258 MB 1234

Legend:
Indx = Index number used for referencing the device
Rev  = Firmware Revision of the device

Note:  Due to different calculation methods used, the size
       of the device shown is only a rough approximation.

DFDUTIL>
```

Example output of `dfdutil DISPMAP` command with one index number

```
Indx Path           Product ID                     Bus    Size   Rev
---- -------------- -----------------------------  ------ ------ ----
1    2/0/1.6.0      QUANTUMLP270S disc drive       SCSI   258 MB 1234

Legend:
Indx = Index number used for referencing the device
Rev  = Firmware Revision of the device

Note:  Due to different calculation methods used, the size
       of the device shown is only a rough approximation.

DFDUTIL>
```

`DISPFILES` command

The DISPFILES command displays a list of all available firmware files found on a LIF device. The command displays:

- File name
- Intended product identification
- New revision number
- Size of firmware (not file size)

The syntax for this command is:

```
DISPFILES
```

The user may enter the index number of a single device; using no index number causes `DISPFILES` to list all devices.

`LS` command

The `LS` command displays information about the LIF volume. The display is similar to that displayed by a `lifls -l` command. This command is used for writing and maintaining `dfdutil`.

`RESET` command

The `RESET` command only resets the internal variables of the `dfdutil` utility by resetting all variables and lists of original values. It rescans each bus to detect any devices. It does not reset any SCSI buses. Therefore, the resets display produced may not reflect any downloads that may have been done since `dfdutil` was started.

The syntax for this command is:

```
RESET
```

`UTILINFO` command

This command provides general information about the use of `dfdutil`.

`NODE` command

`dfdutil` uses the networked console feature to allow all command input and output to be sent through the console connected to node 0. The `NODE` command allows the user to move execution to another node for firmware download to the desired device. The command causes `dfdutil` to rescan the device bus in the node to which the user is switching. The syntax for this command is:

```
NODE <node number>
```

where `<node number>` is the appropriate node ID (0, 2, 4, or 6).

Typically, bootable devices are connected to node 0 and the `NODE` command is not needed.

`HELP` command

The HELP command provides useful information about `dfdutil` commands.

The syntax for this command is:

```
HELP <command name>
```

Entering **`HELP`** without a command name displays a list of all available `dfdutil` commands. Entering the specific command name after **`HELP`** outputs specific information about the command.

Notes and cautions about `dfdutil`

This section presents some limitations and cautions concerning `dfdutil`.

Backup before downloads

Some firmware downloads may affect formatting resulting in the loss of some or all the data on the disk.

CAUTION Back up all disks before loading firmware onto them.

Halting the system during downloads

Halting the system during a download may leave the drive being downloaded in an unusable state.

CAUTION Never halt the computer, power cycle it, or in any way interrupt operation during a download.

Power cycling after a download

Some disk drives store downloaded code to nonvolatile memory but do not load and run this code until after the next bus reset or power cycle.

NOTE Power cycle the system and all cabinets or racks containing drives that have been downloaded after all downloads have been completed. Restart `dfdutil` and examine the revision levels in the bootable device table to make sure that all downloads were successful.

If attempting to download a corrupted or inappropriate firmware file to some drives, the drives drop the downloaded data and return good status. For this reason, `dfdutil` can not always determine if a download did, in fact, complete successfully.

Shared SCSI Buses

If `dfdutil` is running on a system which shares any of its SCSI busses with another system or systems, the other system or systems must be halted while this program is running. This program can not determine that a bus is shared, so the operator must determine if any bus is shared and halt the other computer(s).

Shared Nike Arrays

If `dfdutil` is running on a system which shares a Nike array with another system, it is not possible to update firmware on the Nike's SP boards or drives without manual intervention. This program can detect that the array is shared and display a message to pull the SP board connected to the other system and reinsert the board after the download(s) is complete.

Nike and Galaxy drive download to the individual disks in the array is not possible with two active SP controllers in the cabinet. One SP must either be physically removed or shut down via remote maintenance software (accessed via the serial port).

dump_rdrs

The `dump_rdrs` utility automatically resets the specified node and directs it to boot the RDR dumper firmware module. Once it detects that the RDR dumper firmware has completed, it scans out the results and places a formatted RDR dump of each processor in /spp/data/<complex>/nodeX.cpuY.rdrs. X is the node number specified and Y is a processor number from 0 - 31.

Example of dump_rdrs utility:

```
dump_rdrs <node id>
```

fwcp

`fwcp` is an OBP command that upgrades system firmware. A single firmware package may be loaded by the following command:

```
ok fwcp <filename> <module name>
```

To load all system firmware packages, use the following master download script:

```
source /core@f0,f0000000/
lan@0,d30000;15.99.111.99:/spp/scripts/dl-diags
```

The master download script output is shown below:

```
v-c-t:/spp/firmware$ cat /spp/scripts/dl-diags
fwcp 15.99.111.99:/spp/firmware/pdcfl.fw PDCFL
fwcp 15.99.111.99:/spp/firmware/post.fw POST
fwcp 15.99.111.99:/spp/firmware/test_controller.fw TC
fwcp 15.99.111.99:/spp/firmware/cpu3000.fw CPU3000
fwcp 15.99.111.99:/spp/firmware/io3000.fw IO3000
fwcp 15.99.111.99:/spp/firmware/mem3000.fw MEM3000
fwcp 15.99.111.99:/spp/firmware/diodc.fw DIODC
fwcp 15.99.111.99:/spp/unsupported/rdr_dumper.fw
RDR_DUMPER
fwcp 15.99.111.99:/spp/firmware/entry2500.pdc /flash@0,0
fwcp 15.99.111.99:/spp/firmware/obp2500.pdc OBP
```

fw_init and fw_install

`fw_init` provides an automatic means for downloading firmware to each node and initializing certain data structures in NVRAM. Using this script prevents problems that could occur when executing this procedure manually. The format if `fw_init` is as follows:

```
fw_init [-c complex name]
```

`-c complex name` specifies the complex to update.

For example:

`fw_init`	updates all nodes in the current complex.
`fw_init -c hw2a`	updates all nodes in the complex hw2a.

If the `-c option` is not specified, then the `complex_name` value is obtained either from an environment variable of the same name or it defaults to `mu`.

`fw_init` first loads the JTAG core firmware and the JTAG firmware. The complex is then reset to OBP. The script then loads the diagnostic LIF header to each node in the complex. The complex is then reset to OBP again in order to download firmware to all the nodes. A source command is issued to OBP that loads all the firmware listed in the "/spp/scripts/dl-diags" file into Flash memory. After this completes, the /spp/bin/tc_init utility is executed which initializes certain NVRAM data structures used by the Test Controller.

The `fw_init` script then initiates the downloading of PCI ROM firmware by using the `pciromldr` firmware.

This script must be executed as root. If not then an error message is printed and the script terminates. The error message is as follows:

```
This script must be run as root.
```

Messages are periodically printed to the console while `fw_init` is executing. Examples of these messages are show below:

`fw_init` message example 1

```
****************
**   WARNING  **
****************
To allow interaction with OBP, this script will
automatically reset the “<complex name>” complex.

If HP-UX is currently running on “<complex name>”,
perform a clean shutdown (to the OBP prompt) before
proceeding.

Do you wish to continue? (y or n)
```

`fw_init` message example 2

```
Starting the firmware download and initialization
process.
```

`fw_init` message example 3

```
Loading JTAG core firmware on "hw2a-0000".
```

`fw_init` message example 4

```
Loading JTAG firmware on "hw2a-0000".
```

`fw_init` message example 5

```
Loading/verifying Diagnostic LIF header on "hw2a-0000".
```

`fw_init` message example 6

```
The new LIF header will now be activated.
```

`fw_init` message example 7

```
Saving NVRAM contents and beginning firmware download
via OBP.
```

`fw_init` message example 8

```
Now resetting to boot new POST/OBP.
```

`fw_init` message example 9

```
Now restoring NVRAM.  Please wait.
```

`fw_init` message example 10

```
Initializing the test_controller data structures.
```

`fw_init` message example 11

```
Performing file cleanup and miscellaneous tasks.
```

`fw_init` message example 12

```
Downloading PCI ROM images (if out of date).
```

`fw_init` message example 13

```
The <complex name> complex has been reset to OBP.

The firmware download and initialization has been
completed.
```

fw_install

`fw_install` documents the manual process of updating firmware. To use it, enter:

```
man fw_install
```

Using `fw_init` is the suggested method, but there are cases, however, where it can not be used. If an error occurs during the `fw_init` script, the following procedure and is an example of how to upgrade firmware. It assumes a node number of 0 and a complex name of `vserver`. Example commands for nodes 0 and 2 are shown below. Similar commands would be necessary for nodes 4 and 6 as well.

As this is an example, substitute the appropriate node entries into each command. For example, if the complex name is datacenter6, the node names would be datacenter6-0000, datacenter6-0002, etc.

Step 1. At a shell prompt on the SSP, type the following commands:

```
set_complex vserver

load_eprom -n vserver-0000 -c /spp/firmware/jtag_core.fw

load_eprom -n vserver-0000 -j /spp/firmware/jtag.fw
```

If the system has multiple nodes, perform a similar command for each of the nodes in the configuration using the appropriate node number.

Step 2. Power cycle the node at this time to allow new JTAG firmware to take effect.

Step 3. Wait for `ccmd` to complete scan interrogation of the node.

Step 4. At the shell prompt, load the new processor-dependent code and off-line diagnostics firmware by typing the following commands:

```
load_eprom -n vserver-0000 -l /spp/firmware/diaglifhdr.fw
```

If the system has multiple nodes, perform a similar command for each of the nodes in the configuration using the appropriate node number.

Step 5. Reset the system by entering the following command:

```
reset all 1 obp
```

Step 6. Wait for the node(s) to boot to OBP, then at the console for node 0 change to forthmode by entering the following command:

`fm` <enter>

Step 7. At the OBP “OK” prompt, type the following:

```
bcast fc nvsave
source /core@f0,f0000000/lan@0,d30000;15.99.111.99:/spp/scripts/dl-diags
```

Step 8. Wait for the new PDC and off-line diagnostics firmware to be loaded on each node. When the download is complete, enter the following command:

```
reset
```

Step 9. Wait for OBP to reboot and then change to forthmode by entering:

`fm` <enter>

Step 10. At the “ok” prompt type the following command:

```
bcast fc nvrestore
```

Step 11. Initialize test controller NVRAM structures by entering the following commands at a shell prompt on the SSP:

```
su root
<enter root password>
```

```
/spp/bin/tc_init
```

```
exit
```

Step 12. Download the PCI ROM images by entering the following command at the shell prompt on the SSP:

```
/spp/bin/tc_ioutil all pciromldr.fw
```

Step 13. When the `pciromldr` prompt appears on the console, enter the following command:

```
bcast download
```

The `pciromldr` firmware verifies the PCI ROM firmware in each node and performs updates as necessary.

Step 14. When the `pciromldr` prompt appears on the console, reset the system from the SSP using the following command:

```
do_reset all 1 obp
```

get_node_info

The `get_node_info` utility provides as a mechanism for scripts or programs to access the SSP configuration information generated by the `ts_config` configuration tool. It has the following format:

```
get_node_info [node_info] [OPTIONS]
```

When a node is configured by `ts_config`, an entry is added to a node configuration file. Each node entry contains the following information:

- `Complex Name`—Complex name assigned in `ts_config`
- `Node ID`—V Class Node ID
- `Diagnostic IP hostname`—IP hostname of Diagnostic Utility interface
- `OBP IP hostname`—IP hostname assigned to OBP LAN interface
- `SSP Diagnostic hostname`—IP hostname assigned to the SSP (teststation) Diagnostic interface
- `Console name`—Name assigned to V Class console

`get_node_info` obtains the SSP configuration information about all nodes or a single node. If `-A` is used to request information on all nodes, the node entries are returned in the order they appear in the configuration file (they are not sorted).

By default, the information returned includes all of the configuration fields. `OPTIONS` select a subset of the available fields. The output fields are returned (to standard output) in the order shown above, regardless of the ordering of `OPTIONS`.

`[node_info]` must uniquely identify a node on the SSP, a `Node ID` (for example, `0`) or the `Diagnostic IP hostname` (for example, `swtest-0000`)

If a `Node ID` is specified, `get_node_info` determines the node `Complex Name` from the `COMPLEX_NAME` environment variable. Use the `set_complex` command to set the desired complex name.

`[OPTIONS]` include the following:

- `-a`—Display all fields (default)
- `-A`—Display all configured nodes

The selected fields will be printed in the order below)

- `-c`—Display the Complex name
- `-n`—Display the Node id
- `-m`—Display the Diagnostic IP hostname
- `-o`—Display the OBP IP hostname
- `-t`—Display the SSP Diagnostic hostname
- `-s`—Display the console name

The following are examples of the `get_node_info` utility:

Example showing the return all information about Node Id 0:

```
joker-t(hw2a):/users/sppuser$ get_node_info 0
hw2a 0 hw2a-0000 obp-hw2a-0000 tsdart-d Serial_1 2
```

Example of returning the complex name associated with the Diagnostic name

```
joker-t (hw2a): /users/sppuser$ get_node_info hw2a-0000 -c
hw2a
```

The `sppconsole` script contains an example use of the `get_node_info` utility.

hard_logger

`hard_logger` is a script that invokes the interrogators and extractors to log all error information on a node

The usage of the script is:

```
hard_logger [node number]
```

`[node number]` is a hex number.

`hard_logger` resides in /spp/scripts/hard_logger and is automatically invoked be `ccmd` when a hard error occurs.

The `hard_logger` script performs the following tasks:

- Parses the command line arguments to determine on which node it should run. `ccmd` sets up the `COMPLEX_NAME` environment variable before invoking `hard_logger`. The SSP utilities called by `hard_logger` use the combination of `node_id` and `COMPLEX_NAME` to determine with which node to communicate.
- Acquires COP information for the node using `sppdsh` and saves the output to /spp/data/<COMPLEX_NAME>/hl/T_FILE_n$node
- Acquires PCE information using `sppdsh` and saves the output to /spp/data/<COMPLEX_NAME>/hl/T_FILE_n$node.
- Checks the Stop On Hard bits of each SPAC to find one that is running. If an SPAC is running, then `hard_logger` gets information from the SMUC CSRs.
- Reads SMUC CSRs. If there is no hard error, `hard_logger` quits.
- Traverses the list of hard error buses. If a bus reports a hard error, then it performs the following:
    - Interrogates each controller on that bus for hard errors.

      If the hard error group pin is set to a one value, it ignores the controller.

      If the pin is a zero value, the controller may have been the first to record the error.
    - Interrogates the controller reporting the error.

To interrogate the controllers, `hard_logger` calls the ASIC specific interrogator located in /spp/scripts/<asic>.

For example, the SMAC interrogator is located in /spp/scripts/smac

The interrogator returns a list of extractors to run on that ASIC in /spp/data/<COMPLEX_NAME>/hl/inter_n$node.

- Runs each extractor returned by the interrogator.

- Sends the COP, PCE, interrogator, and extractor output to `event_logger`. `event_logger` forwards the COP, PCE, and extractor output to both the SSP message console window and the `ccmd` log file /spp/data/ccmd_log.
- Logs the results in /spp/data/<complex_name>/hard_hist.

  hard_hist is a permanent file that records the date, time, and results of the script.

The last run of `hardlogger` on a node is preserved in /spp/data/<complex_name>/hl/OUTPUT_FILE_n$node.

lcd

`lcd` prints the current contents of the liquid crystal display for node 0 of the current complex. It has the following format:

```
lcd
```

The complex can be changed by using the `set_complex` utility. The output is sent to `stdout` output.

Example output of `lcd`

```
0 (0,0)
I-I- ---- I-P- ----
---- ---- ---- ----
abcedfghijklr-
```

For more information, see “Front panel LCD” on page 13.

load_eprom

The `load_eprom` utility resides on the SSP. It downloads the core firmware products into the EEPROM on the Utilities board through the scan interface. It can also update the JTAG scan interface controller firmware. If, during a download, it detects any errors, it automatically retries the download.

The `load_eprom` utility uses subroutines that perform the following functions:

- It reads a raw binary file on the SSP.
- It erases the specified Flash sector and verifies that the erase was successful. It will retry if the erase fails.
- It scan downloads the contents of the binary in 4096-byte page increments, updating the screen for each page. A “w” is printed during the write operation, an “r” during the optional read operation, a “v” during the optional verify operation and a “.” when the page is complete.
- It can optionally read each page back for verification.
- It can read-verify a binary in the Flash EEPROM and compare it to the binary on the SSP, without performing the write operation.

The `load_eprom` utility usage is as follows:

```
load_eprom -n <IP name> [-QRV] [-P #] [-j|c|e|p|o|t|l|f]
<file>
```

The options available are given in Table 91.

Table 91 `load_eprom` options

Option	Description
`-Q`	Quiet (no) output mode.
`-R`	Read and verify data only-No writing.
`-P number`	SPAC to use for scan operations where number is 0-7, 8 is UBUS.
`-V`	Verify data after a write.
`-j <file>`	Load binary into JTAG flash.
`-c <file>`	Load binary into JTAG_CORE flash.
`-e <file>`	Load binary into PDC Entry section.
`-p <file>`	Load binary into PDC POST section.
`-o <file>`	Load binary into PDC OBP section.
`-t <file>`	Load binary into PDC Test Controller section.
`-l <file>`	Load binary into LIF file section.
`-f <file>`	load binary into PDC firmware loader section

As an example, entering the following reads the file /spp/firmware/post.fw and updates the POST section of Flash EEPROM on the Utilities board.

```
xns3_d% load_eprom -n hw2a-0000 -p /spp/firmware/
post.fw
```

Entering the following reads the file ./jtag.fw and updates the Flash EEPROM for the JTAG controller:

```
xns3_d% load_eprom -n hw2a-0000 -j jtag.fw
```

The following are three addition examples of the `load_eprom` command. The first two write to one sector in EEPROM and the last writes across several sectors.

Example output of `load_eprom -n hw2a-0000 -e entry.pdc` command

```
Reading file “entry.pdc”: 4253 (0x109d) bytes read.
Using default SPAC (P0L).
Erasing sector 0  (0xf0000000) OK
Writing sector 0  (0xf0000000) .. OK
```

Example output of `load_eprom -n hw2a-0000 -p post.fw` command

```
Reading file “post.fw”: 92820 (0x16a94) bytes read.
Using default SPAC (P0L).
Erasing sector 4  (0xf0020000) OK
Writing sector 4  (0xf0020000) ....................... OK
```

Example output of `load_eprom -n hw2a-0000 -o obp.pdc` command

```
Reading file “obp.pdc”: 499712 (0x7a000) bytes read.
Using default SPAC (P0L).
Erasing sector 7  (0xf0080000) OK
Writing sector 7  (0xf0080000) ................................ OK
Erasing sector 8  (0xf00a0000) OK
Writing sector 8  (0xf00a0000) ................................ OK
Erasing sector 9  (0xf00c0000) OK
Writing sector 9  (0xf00c0000) ................................ OK
Erasing sector 10 (0xf00e0000) OK
Writing sector 10 (0xf00e0000) .......................... OK
```

While `load_eprom` is writing a block of data, a “w” is printed. If the write is successful, a dot is printed. The dots continue until the whole sector is successfully written, at which time the “OK” is printed.

opie

The V-Class ODE Peripheral Interface Emulation utility (`opie`) allows some HP ODE Test Modules (TMs) to be executed on V-Class platforms. ODE (Off-line Diagnostics Environment) is the utility used to test, configure, and update I/O devices and controllers on most HP platforms.

The ODE architecture is incompatible with V-Class hardware. `opie` emulates and translates the ODE calls made by TMs to calls that the V-Class firmware understand. Test modules and `opie` are executed via the Test Controller.

Version 2.0 or higher of the V-Class off-line diagnostics provide several components required to use `opie`.

Test modules

The lif volume containing the test module and related firmware should be placed in the /spp/firmware directory on the SSP. After HP-UX is shutdown on the V-Class (**`shutdown -h`**), the system is ready to run test modules via `opie`.

`opie` accommodates most I/O ODE test modules including the following:

- DFDUTIL
- AR60DIAG
- ARDIAG
- NIKEARRY

Test modules can be obtained from

`http://fwserver.mayfield.hp.com/firmware/`

or from ODE distribution Cads.

Any CPU or Memory test modules are ineffective on V-Class systems.

Using opie

To use a particular test module, perform these steps:

Step 1. Install the test module in the /spp/firmware directory

Step 2. Login in the SSP as sspuser,

Step 3. Execute the following command:

```
tc_ioutil all [TMNAME]
```

SSP packs and runs test module.

For example:

Entering

```
tc_ioutil all DFDUTIL
```

executes the `dfdutil` test module on the V-Class platform.

In this example, when `tc_ioutil` is invoked, it opens the lif volume DFDUTILlif in the directory /spp/firmware which contains the `dfdutil` test module. It then copies the test module, along with the `opie` utility and a loader utility called `opie_loader.fw`, into a file called DFDTUTILTM.

`tc_ioutil` then issues a `reset` command to all selected nodes with the ioutil mode of the test controller as the boot target. After going through POST, the node executes the test controller in ioutil mode. The test controller "tftps" the TM file (DFDUTILTM in the above example) into coherent memory on the node. It then branches to the loader program, which extracts the test module and `opie` utility and begins executing the test module.

The following is an example console output:

```
Booting DIAG
HP9000/V2500 Test Controller, version 2.0 1999/09/17 17:47:21
Monarch = PB0L_A
Loading file DFDUTILTM
................................................................................
................................................................................
.................................
DFDUTILTM TM copied, calling loader
    *********************************************************************
    ***                                                               ***
    ***                 Disk Firmware Download Utility (DFDUTIL)      ***
    ***                                                               ***
    ***        (C) Copyright 1994 - 1999 by Hewlett-Packard Company   ***
    ***                   All Rights Reserved                         ***
    ***                                                               ***
    ***  This program may only be used by HP support personnel and    ***
    ***  those customers with the appropriate Class license or        ***
    ***  Node license for systems specified by the license.  HP       ***
    ***  shall not be liable for any damages resulting from misuse    ***
    ***  or unauthorized use of this program.  This program           ***
    ***                                                               ***
    ***                 Version A.02.05 (7/1/99)                      ***
    ***                                                               ***
    *********************************************************************

***************************************************************************
*                              WARNING!                                   *
*                             ---------                                   *
* DFDUTIL must have exclusive access to all the disks you wish to update. *
* If you are in a multihost environment such as Switchover & ServiceGuard,*
* make sure all other hosts are powered down before continuing.           *
***************************************************************************
Please wait while I scan the device busses...
***************************************************************************
*                    HP Supported Disks Found                             *
***************************************************************************
Indx  Path              Product ID                   Bus    Size   Rev
---- -----------------  ---------------------------- ------ --- --- ----
0    7/0.8.0.100.0.6.0  SEAGATE ST15150W             Fibre  3.9 GB HP10
 Legend:
Indx = Index number used for referencing the device
Rev  = Firmware Revision of the device
 Note:  Due to different calculation methods used, the size
       of the device shown is only a rough approximation.
Create a firmware file list? (q for quit) [default for y]
Please wait while I search for all the firmware files.
Note: This may take a while if you are booting from tape.
  OPIE: Please wait while LIF file is loaded from teststation.
  OPIE: Opening file "/spp/firmware/DFDUTILlif".
  OPIE: Transferring file (cursor will spin during transfer)...

************************************************************************
            Firmware Files Found (not disks)                           *
************************************************************************
File name        Intended Product ID                     Rev.   Size
FRU_SEA4         HPC1300WD or HPC2300WD disk array (Mech) HP     269824
ST136403FC       SEAGATEST136403FC disk drive            HP00   439424
ST136475LC       SEAGATEST136475LC disk drive            HP01   309376
ST31230N         SEAGATEST31230N disk drive              hpm4   261632
ST32171N         SEAGATEST32171N disk drive              HPM1   276512
ST39102LC        SEAGATEST39102LC disk drive             HP02   303360
ST39103FC0       SEAGATEST39103FC disk drive             HP00   439424
ST318203FC       SEAGATEST318203FC disk drive            HP00   439424
ST318275LC       SEAGATEST318275LC disk drive            HP01   309376
 Legend:
File name           = name of the firmware file
Intended Product ID = firmware file's intended product name
Rev.                = firmware Revision of the firmware file
Size                = exact byte size of the firmware image
 DFDUTILTM>
```

The test module is now at the user prompt as shown by:

```
DFDUTILTM>
```

waiting for input.

If the test module calls a function that is not implemented by `opie`, there error message is printed on the console. If it is a critical function, the test module may exit or prompt the user for another command.

Every test module has different commands, but each must implement a HELP command that documents all the other commands the test module provides. Type **`HELP`** or **`HELP [command]`** for information on how to perform specific tasks.

Test modules may be added from time to time. Check the engineering group responsible for certification of the hardware to verify that the off-line diagnostic is compatible with the V-Class Off-line Diagnostic Environment and has been tested with `opie`.

pciromldr

The `pciromldr` utility downloads Fcode to the symbios 875 and 895 controllers. It is invoked by the Test Controller I/O utility interface by way of `tc_ioutil`.

The user interface is an extension of the `dfdutil` interface and complies with the standard network console functionality provided by POST, `pdcfl`, and `dfdutil`. There are, however, some features unique to the `pciromldr` interface:

- Automatic selection of all appropriate controllers as targets for download in all nodes in a complex.
- Automatic assignment of the appropriate default firmware file for each controller. For example, the symbios875.pcirom file is automatically assigned to the Symbios 875 controller without user intervention.
- Automatic upgrade on multiple controllers and multiple nodes without user intervention.

pciromldr commands

This section describes `pciromldr` commands.

Broadcast command

`bcast` has the format:

```
bcast <command> [<arg> <arg> ...]
```

This command sequences through the node list issuing the `<command>` to each node. Commands are issued serially. In other words, the command is not sent to the next node until the current node has completely processed the command.

Change target node command

This command changes the current target node and has the following format:

```
node <number>
```

where `<number>` is the desired node target number.

Display map command

This command displays the I/O configuration for each node in the complex and has the following format:

```
dispmap
```

The following is an example of the results after a `bcast dispmap` command:

```
Node 0
      saga/                Flash    FW
Index slot  Vendor/Dev.    ID       Rev  File                 / Rev
-------------------------------------------------------------------
* 0   4/2   0x1000/0x000f 0x2002   7.0  symbios875.pcirom / 7.1
* 1   4/2   0x1000/0x000c 0x2002   7.0  symbios895.pcirom / 7.1
* 2   4/2   0x1000/0x000f 0x2002   7.0  symbios875.pcirom / 7.1

Node 2

      saga/                 Flash   FW
Index slot  Vendor/Dev.     ID      Rev  File                 / Rev
-------------------------------------------------------------------
  0   4/2    0x1000/0x000f 0x2002  7.0  symbios875.pcirom / 7.1
* 1   4/2    0x1000/0x000f 0x2002  7.0  symbios875.pcirom / 7.1
* 2   4/2    0x1000/0x000c 0x2002  7.0  symbios895.pcirom / 7.1
```

The asterisks indicate that the card is selected and will be upgraded when the download command is issued on that node. Notice that the SAGA number is a node-relative (not complex) SAGA number.

Download command

This command downloads a firmware file and has the following format:

```
download <filename>
```

If `<file_name>` is specified, that specified file is the downloaded firmware source instead of the currently assigned firmware file listed in the `dispmap` output.

Display files command

This command displays the file name and vendor/device ID cross reference table. This table defines the default firmware file name for each supported PCI card.

This command has the following format:

```
dispfiles
```

The following is a an example output of the `dispfiles`:

```
% dispfiles

 Index  Vendor/Dev.    File name          Supported ROM types
 ---------------------------------------------------------

 0      0x1000/0x000f  symbios875.pcirom  0x012a, 0x89bd, 0x01a7,
                                          0x89b4, 0x0125, 0x89b8,
                                          0x2002, 0x01a1
 1      0x1000/0x000c  symbios895.pcirom  0x012a, 0x89bd, 0x01a7,
                                          0x89b4, 0x0125, 0x89b8,
                                          0x2002, 0x01a1
```

Change cross reference table command

This command changes the table cross reference entry and has the following format:

```
assign <index> <file_name>
```

Example:

```
assign 0 symbios.pcirom
```

Select and deselect cards commands

This `select` command selects the card(s) corresponding to the index as a firmware upgrade target and has the following format:

```
select <index> [ [- <index> ] | [, <index> , <index> ...] ]  | all
```

This `deselect` command deselects the card(s) corresponding to the index as a firmware upgrade target and has the following format:

```
deselect <index> [ [- <index> ] | [, <index> , <index> ...] ]  | all
```

Set option and value command

This command sets the value for the `verchk` and `flashdevchk` options and has the following format:

```
set <option> <value>
```

where <option is either `verchk` or `flashdevchk` and the value is on or `off` as follows:

```
verchk on|off
```

If the value is set to on, then the firmware is upgraded only when the target controller's firmware revision is different than the revision of the firmware file. The default is on.

```
flashdevchk on|off
```

If the value is set to on, then the firmware is only upgraded when the ROM ID in the selected card matches a compiled list of ROM IDs for the corresponding card. The default is on.

pim_dumper

`pim_dumper` is a utility used to display Process Internal Memory (PIM) information after a TOC, LPMC, or HPMC. The PIM dump information includes the processor registers and various ASIC registers. It has the following format:

```
pim_dumper [-c CPU#] [-n NODE_PARM] [-t][-l][-h] [-e][-help]
```

Example of pim_dumper use:

`pim_dumper -h -c 2`

This example displays HPMC information for Processor 2 on Node 0.

The PIM information will be appended to the file /spp/data/ `<COMPLEX_NAME>`/pimlog, where `<COMPLEX_NAME>` is the name associated with the desired node. Optionally, a copy of the PIM information can be written to standard output.

`pim_dumper` can be invoked without any command line options. By default, it dumps all available (TOC/LPMC/HPMC) information for all enabled processors on node 0.

Table 92 lists `pim_dumper` options.

Table 92 **`pim_dumper`** options

Option	Description
`-c CPU number`	Request a specific processor
`-c all`	Select all processors (default: all)
`-n NODE_PARM`	Specify the desired node ID (default: 0) or node name (e.g. test-0000)
`-t`	Display TOC information (default: on)
`-l`	Display LPMC information (default: on)
`-h`	Display HPMC information (default: on)
`-e`	Echo PIM to standard output (default: off)
`-help`	Display usage information

The TOC/LPMC/HPMC options are mutually exclusive. Specify only one of these options; do not specify any, and the default mode dumps all TOC/LPMC/HPMC data.

If `pim_dumper` is able to accomplish the desired action, it returns zero. If for any reason the requested operation cannot be completed, a non-zero exit code is used.

set_complex

The `set_complex` sets the default V2500/V2600 Complex Name in the current shell environment.

```
set_complex [COMPLEX_NAME]
```

Once set, SSP diagnostic or console utilities that are run from within the shell operate on the specified complex.

If multiple complexes are configured on a single SSP, individual shells can each be set to a specific default complex using `set_complex`. Diagnostic and console commands entered from the shell access the desired node as if it were the only complex on the SSP.

Example of command entered from the shell

```
joker-t (hw2a): /users/sppuser$ sppconsole 0
```

In this example, the command accesses the console for Node ID 0 in the hw2a complex.

Users may temporarily override the default complex by including the full Diagnostic Node name in the Diagnostic or console command. For example, even though the default complex is set to hw2a, the following command requests `flash_info` from Node ID 0 in the hw2b complex:

```
joker-t(hw2a):/users/sppuser$ flash_info hw2b-0000
```

NOTE `jf-ccmd_info` lists the diagnostic node names for all active nodes on the SSP Diagnostic LAN.

The customized shell environment for the sppuser account automatically runs `set_complex` during login. If a single V2500/V2600 complex has been configured, the default `COMPLEX_NAME` is assigned automatically. If more than one complex is configured, the user is prompted for the desired complex. With help from the parent shell, `set_complex` causes the `COMPLEX_NAME` environment variable to be set appropriately.

`set_complex` also updates the shell prompt to reflect the default complex name. The complex name is enclosed in parenthesis in the prompt string. If the shell is running on the SSP desktop, `set_complex` also updates the shell window title.

`set_complex` can be invoked anytime the user wants to change the shell default complex. If the user enters an invalid `COMPLEX_NAME`, the default complex becomes unset and the prompt string indicates this condition. If the user does not enter a `COMPLEX_NAME`, the complex name remains set (assuming it is still a valid complex).

`set_complex` does not work from within a shell script. An alternative is to explicitly set the COMPLEX_NAME environment variable using the appropriate mechanism for the current shell script type.

Example showing change of complex name in a shell script

```
+++++++++++++++++++++++++++++++++++++++++++
+#!<shell>                                +
+                                         +
+ COMPLEX_NAME=hw2a; export COMPLEX_NAME  +  (sh/ksh/sppdsh)
+ <OR>                                    +
+ setenv COMPLEX_NAME hw2a                +  (csh/tcsh)
+                                         +
+ dcm 0                                   +
+ ...                                     +
+++++++++++++++++++++++++++++++++++++++++++
```

NOTE Scripts that are run from a shell using `set_complex` receive the correct COMPLEX_NAME environment variable from the parent shell. The limitation is that `set_complex` cannot set the `COMPLEX_NAME` environment variable when run from within a script.

soft_decode

`soft_decode` decodes single-bit ECC error data. This `perl` script decodes single-bit ECC error information. It prompts for syndrome, row, and address information that is parsed, decoded, and displayed in an easy-to-read format that can be cut-and-pasted into quasar.

To exit enter **`q`**.

Example of `soft_decode` use:

```
% soft_decode

Enter RAM size (16, 64 or 128): 16
Enter syndrome code: 64
Enter row number: 2

Enter address: 04589030
Single-Bit ECC Error RAM Information
====================================
Location      Bit    Pin#    Row   Address
--------      ---    ----    ---   -------
U013A7       DQ2      6      2     589030
Enter RAM size (16 or 64): q
exiting
```

sppconsole

`sppconsole` connects the user to the console for a specified node.

`sppconsole` has the following format:

```
% sppconsole node [opt1, ..., optN
```

There are several ways to initiate the `sppconsole` interface.

- Run the `sppconsole` command in a shell on the SSP.
- Select from the SSP root menu the desired V2500/V2600 complex, then select “Console” and the desired node.
- Use the `consolebar` utility to select the desired node.

The `sppconsole` script invokes the /spp/etc/console program (passing any optional arguments and the node number) to provide the console interface to the V2500/V2600 node. This interface communicates with POST, OBP, the Test Controller, and the HP-UX operating system. It starts up a window and connects the user to the console server, that is the `conserver` daemon, running on the SSP. After making the connection, the last 20 lines of the console output are displayed.

The `conserver` daemon is started by `init` when the SSP is booted. The daemon reads the /spp/data/conserver.cf file to determine which console terminals to open and maintain.

All errors and information messages are logged in the system log file /var/adm/syslog/syslog.log.

The `sppconsole` script invokes the `/spp/etc/console` program to provide the operating system a console interface. Refer to the `console(8)` man page for more information about this program.

The following shows the typical output in the console window when the node boots.

Example of `sppconsole` boot output

```
joker-t(hw2b)% sppconsole
[enter '^Ec?' for help]
[no, sppuser@joker-t is attached]
[replay]
POST Hard Boot on [0:PB0L_A]
HP9000/V2500 POST Revision 1.0.0.1, compiled 1998/12/03 09:50:10 (#0039)
Probing CPUs: PB0L_A PB1R_A PB2L_A PB3R_A PB4L_A PB5R_A PB6L_A PB7R_A
Completing core logic SRAM initialization.
Starting main memory initialization.
 Probing memory: MB0L MB1L MB2R MB3R MB4L MB5L MB6R MB7R
 Installed memory: 24576 MBs, available memory: 13312 MBs.
 Initializing main memory.
  Parallel memory initialization in progress.
                r0         r1         r2         r3
  PB0L_A MB0L [:::: ::::][:::: ::::][:::: ....][:::: ....]
  PB1R_A MB1L [:::: ::::][:::: ::::][:::: ....][:::: ....]
  PB2L_A MB2R [:::: ::::][:::: ::::][:::: ....][:::: ....]
  PB3R_A MB3R [:::: ::::][:::: ::::][:::: ....][:::: ....]
  PB4L_A MB4L [:::: ::::][:::: ::::][:::: ....][:::: ....]
  PB5R_A MB5L [:::: ::::][:::: ::::][:::: ....][:::: ....]
  PB6L_A MB6R [:::: ::::][:::: ::::][:::: ....][:::: ....]
  PB7R_A MB7R [:::: ::::][:::: ::::][:::: ....][:::: ....]
 Building main memory map.
Main memory initialization complete.
Booting OBP
```

After POST initializes the system, OBP boots. The following is a sample of the output.

Example of OBP output while booting

```
OBP Power-On Boot on [0:0]
------------------------------------------------------------------------------
                       PDC Firmware Version Information
                            PDC_ENTRY version 4.1.0.9
                             POST Revision: 1.0.0.1
          OBP Fieldtest Release 4.1.0.9, compiled 98/10/30 14:11:20  (3)
                 SPP_PDC Fieldtest 1.4.0.19 (11/12/98 19:17:49)

------------------------------------------------------------------------------
Proc type   Proc#   Proc Rev  Speed     State   Dcache   Icache   I-prefetch
----------  -----   --------  -------  -------  -------  -------  ----------
HP,PA8500   0       2.0       440 MHz  Active   1024 KB  512 KB   On
HP,PA8500   2       2.0       440 MHz  Active   1024 KB  512 KB   On
HP,PA8500   4       2.0       440 MHz  Active   1024 KB  512 KB   On
HP,PA8500   6       2.0       440 MHz  Active   1024 KB  512 KB   On
HP,PA8500   8       2.0       440 MHz  Active   1024 KB  512 KB   On
HP,PA8500   10      2.0       440 MHz  Active   1024 KB  512 KB   On
HP,PA8500   12      2.0       440 MHz  Active   1024 KB  512 KB   On
HP,PA8500   14      2.0       440 MHz  Active   1024 KB  512 KB   On
Primary boot path   = 0/0/0.6.0
Alternate boot path = 15/3 NFS 15.99.111.99:/spp/os/uxinstlf
Console path        = 15/1
Keyboard path       = 15/1
[*** Manufacturing (or Debug) Permissions ON ***]
System is HP9000/800/V2500 series
Autoboot and Autosearch flags are both OFF or we are in HP core mode.
Processor is entering manual boot mode.
Command                         Description
-------                         -----------
AUto [BOot|SEArch ON|OFF]       Display or set the specified flag
BOot [PRI|ALT|<path> <args>]    Boot from a specified path
BootTimer [time]                Display or set boot delay time
CLEARPIM                        Clear PIM storage
CPUconfig [<cpu>] [ON|OFF|SHOW] (De)Configure/Show Processor
DEfault                         Set the system to defined values
DEfault                         Set the system to defined values
DIsplay                         Display this menu
ForthMode                       Switch to the Forth OBP interface
IO                              List the I/O devices in the system
LS [<path>|flash]               List the boot or flash volume
OS [hpux|sppux]                 Display/Select Operating System
PASSword                        Set the Forth password
PAth [PRI|ALT|CON] [<path>]     Display or modify a path
PDT [CLEAR|DEBUG]               Display/clear Non-Volatile PDT state
PIM_info [cpu#] [HPMC|TOC|LPMC] Display PIM of current or any CPU
RESET [hard|debug]              Force a reset of the system
RESTrict [ON|OFF]               Display/Select restricted access to Forth
SCSI [INIT|RATE] [bus slot val] List/Set SCSI controller parms
SEArch [<path>]                 Search for boot devices
SECure [ON|OFF]                 Display or set secure boot mode
TIme [cn:yr:mo:dy:hr:mn[:ss]]   Display or set the real-time clock
VErsion                         Display the firmware versions
Command:
```

The following message appears in the console window:

```
[0:1] ok [read-only -- use `^Ecf' to attach, `^Ec?' for
help]
```

Attach to the node by entering **Ctrl** **ecf**.

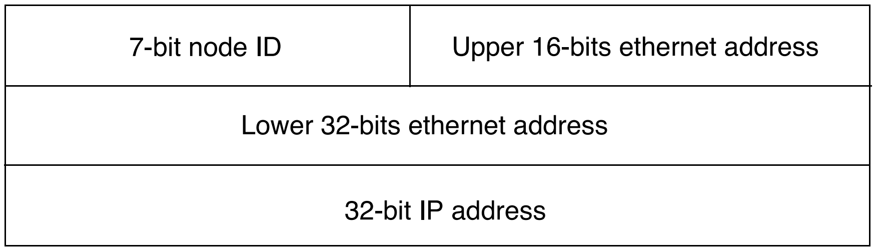
Press the **Ctrl** key **e** simultaneously; do not press the **Ctrl** key with the **c** and **f**.

All information and error messages are logged into the /usr/adm/syslog system error log file.

tc_init

`tc_init` determines the node ID, ethernet address, and IP address for all nodes in the complex. This information is then stored in the NVRAM of all nodes as one 12-byte entry per node. Each 12-byte entry has the format shown in Figure 73:

Figure 73 **`tc_init`** **NVRAM entry**


7-bit node ID
Upper 16-bits ethernet address
Lower 32-bits ethernet address
32-bit IP address

In addition, `tc_init` updates the ARP entries on the SSP by executing as root. If it can not execute as root, then the following error is displayed:

```
** This utility must be executed as root.
** Please login as root and try again.
```

`tc_init` outputs node information shown in the following example.

`tc_init` sample output

```
ex-c2-t% tc_init

  Node = 0 [index=0]
    7-bit node id    = 00
    Host Name        = obp-hw2a-0000
    Upper ether addr = 0x000000a0
    Lower ether addr = 0xd900adb3
    IP addr          = 0x0f636fa6 [15.99.111.166]
    ARP delete command = arp -d obp-hw2a-0000
  obp-hw2a-0000 (15.99.111.166) deleted
    ARP add command = arp -s obp-0000 0:a0:d9:0:ad:b3
```

Execute `tc_init` after the node has been configured by `jf-node_ip_set` and `xconfig`. `ccmd` must finish the scan database generation. Once `ccmd` executes, the changes become effective the next time `test_controller` is running. If `ccmd` is running when `tc_init` is executed then `test_controller` must be restarted.

`tc_init` only needs to be executed once. The following are the only reasons for having to rerun this utility:

- NVRAM is corrupted.
- The system is reconfigured

NOTE `tc_init` is run as part of the `fw_init` script and should not be run under normal circumstances.

tc_ioutil

`tc_ioutil` resets the node and requests that the Test Controller load, (via tftp) and boot the specified file. It has the following format:

```
tc_iotuil <node id or all>  <file>
```

`<node id>` may be a node number, the IP name, or all.

`<file>` should be the name of a file in /spp/firmware.

If the file has a `.fw` extension, then `<file>` is copied, as is, into coherent memory and then executed. If it does not have the `.fw` extension, it is considered to be an ODE test module. See “opie” on page 292 for details in how these test modules are processed.

tc_show_struct

The `tc_show_struct` tool examines certain structures that the test controller uses to set up and run tests. It has the following format:

`tc_show_struct <test_name> <node_number OR node_name>`

Possible selections for the tests are:

- `-mem`
- `-io`
- `-cpu`
- `-eri`

The `tc_show_struct` tool takes two arguments: the first is the test of interest, the second is the node of interest.

Example of `tc_show_struct` output

```
joker-t(hw2b):/users/sppuser$ tc_show_struct -mem 0
-----------------------------------------------------------------------
NODE 0  (hw2b-0000)
-----------------------------------------------------------------------
Name : MEM3000 - EEPROM based memory tests
Entry Pt     ClTb ptr    StTb ptr      HwReq     ParmTbptr  Parm_ptr
-----------------------------------------------------------------------
0xf01d0000  0xf01d0074  0xf01d02a8  0xf01d006c  0xf0840760  0xf0863158
      -----------------------------------------------------------------
Hardware req met = 0 | Test inited = 1
Selected = 0         | TC State = TC_RUNNING
-----------------------------------------------------------------------
Test error cnt = 0   | Loop enable = 0
Loop count = 0       | Paused mask = 0x00
-----------------------------------------------------------------------
Class[0]   =          0    Subtest[0] =       640
Class[1]   =  538968128    Subtest[1] =         0
Class[2]   =    1050624    Subtest[2] =    8389636
Class[3]   =  537395328    Subtest[3] =   12588160
Class[4]   =         32    Subtest[4] =  537395200
-----------------------------------------------------------------------
Current Values for Parameters
 00) 0xa5a5a5a5 01) 0xa5a5a5a5 02) 0x5a5a5a5a 03) 0x5a5a5a5a
 04) 0x00000007 05) 0x00000001 06) 0x00000002 07) 0x00000000
 08) 0x00000000 09) 0x00000000 10) 0x000000f0 11) 0x00000000
 12) 0x00000000 13) 0x00000000 14) 0x00000000 15) 0x00000000
 16) 0x00000000 17) 0x00000000 18) 0x00000000 19) 0x00000000
 20) 0x00000000 21) 0x00000000 22) 0x00000000 23) 0x00000000
 28) 0x00000000 29) 0x00000000 30) 0x00000000 31) 0x00000000
 32) 0x00000000 33) 0x00000000 34) 0x00000000 35) 0x00000000
 36) 0x00000000 37) 0x00000000 38) 0x00000000 39) 0x00000000
 40) 0x00000000 41) 0x00000000 42) 0x00000000 43) 0x00000000
 44) 0x00000000 45) 0x00000000 46) 0x00000000 47) 0x00000000
 48) 0x00000000 49) 0x00000000 50) 0x00000000 51) 0x00000000
 52) 0x00000000 53) 0x00000000 54) 0x00000000 55) 0x00000000
 56) 0x00000000 57) 0x00000000 58) 0x00000000 59) 0x00000000
 60) 0x00000000 61) 0x00000000 62) 0x00000000 63) 0x00000000
 64) 0x00000000 65) 0x00000000 66) 0x00000000 67) 0x00000000
 68) 0x00000000 69) 0x00000000 70) 0x00000000 71) 0x00000000
 72) 0x00000000 73) 0x00000000 74) 0x00000000 75) 0x00000000
 76) 0x00000000 77) 0x00000000 78) 0x00000000 79) 0x00000000
 80) 0x00000000 81) 0x00000000 82) 0x00000000 83) 0x00000000
 84) 0x00000000 85) 0x00000000 86) 0x00000000 87) 0x00000000
 88) 0x00000000 89) 0x00000000 90) 0x00000000 91) 0x00000000
 92) 0x00000000 93) 0x00000000 94) 0x00000000 95) 0x00000000
 96) 0x00000000 97) 0x00000000 98) 0x00000000 99) 0x00000000
100) 0x00000000 101) 0x00000000 102) 0x00000000 103) 0x00000000
104) 0x00000000 105) 0x00000000 106) 0x00000000 107) 0x00000000
108) 0x00000000 109) 0x00000000 110) 0x00000000 111) 0x00000000
112) 0x00000000 113) 0x00000000 114) 0x00000000 115) 0x00000000
116) 0x00000000 117) 0x00000000 118) 0x00000000 119) 0x00000000
120) 0x00000000 121) 0x00000000 122) 0x00000000 123) 0x00000000
124) 0x00000000 125) 0x00000000 126) 0x00000000 127) 0x00000000
-----------------------------------------------------------------------
CPU Mask  = 0x0000    SPAC Mask = 0x00
SMAC Mask = 0x00      STAC Mask = 0x00
SAGA Mask = 0x00
-----------------------------------------------------------------------
CPU 0  - State - TC_CPU_RUNNING                 Subtest 310
CPU 1  - State - TC_CPU_RUNNING                 Subtest 310
CPU 2  - State - TC_CPU_RUNNING                 Subtest 310
CPU 3  - State - TC_CPU_RUNNING                 Subtest 310
CPU 4  - State - TC_CPU_RUNNING                 Subtest 310
CPU 5  - State - TC_CPU_RUNNING                 Subtest 310
CPU 6  - State - TC_CPU_RUNNING                 Subtest 310
CPU 7  - State - TC_CPU_RUNNING                 Subtest 310
CPU 8  - State - TC_CPU_RUNNING                 Subtest 310
CPU 9  - State - TC_CPU_RUNNING                 Subtest 310
CPU 10 - State - TC_CPU_RUNNING                 Subtest 310
CPU 11 - State - TC_CPU_RUNNING                 Subtest 310
CPU 12 - State - TC_CPU_RUNNING                 Subtest 310
CPU 13 - State - TC_CPU_RUNNING                 Subtest 310
CPU 14 - State - TC_CPU_RUNNING                 Subtest 310
CPU 15 - State - TC_CPU_RUNNING                 Subtest 310
CPU 16 - State - TC_CPU_NOT_AVAIL               Subtest 0
CPU 17 - State - TC_CPU_NOT_AVAIL               Subtest 0
CPU 18 - State - TC_CPU_NOT_AVAIL               Subtest 0
CPU 19 - State - TC_CPU_RUNNING                 Subtest 310
CPU 20 - State - TC_CPU_NOT_AVAIL               Subtest 0
CPU 21 - State - TC_CPU_NOT_AVAIL               Subtest 0
CPU 22 - State - TC_CPU_NOT_AVAIL               Subtest 0
CPU 23 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 24 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 25 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 26 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 27 - State - TC_CPU_RUNNING                Subtest 310
CPU 28 - State - TC_CPU_RUNNING                Subtest 310
CPU 29 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 30 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 31 - State - TC_CPU_RUNNING                Subtest 310
```

Version utilities

This section describes the three version utilities.

diag_version

The `diag_version` utility displays the product name and the version of the current SSP software. For example:

```
$ diag_version
```

```
HP9000/V2500_V2600 Diagnostics, Version 2.0
```

flash_info

`flash_info` reads the known entry points for the various products that are stored in flash EEPROM. If they have the correct magic number and the pointer to the version string is not null, the version string is extracted.

If no argument is provided, the lowest node in the complex is used. The node number is entered in hexadecimal.

Example of `flash_info` output

```
uts-t (hw2b) % flash_info 0

Node : 0 (hw2b-0000)

  Program Name       Version           Date        Build Level
----------------------------------------------------------------
pdcfl                   2.0          1999/11/03        0006
post                    2.0          1999/11/03        0114
rdr_dumper              2.0          1999/11/03
test_controller         2.0          1999/11/03        0002
mem3000                 2.0          1999/11/03        0423
eri3000                 2.0          1999/11/03
cpu3000                 2.0          1999/11/03        0001
io3000                  2.0          1999/11/03        0043
diodc                   2.0          1999/11/03        0004
obp                  4.2.0.8
pdc_entry            4.2.0.8
```

ver

`ver` is a SSP version retriever utility. It is used to read and display the version information built into each diagnostic product. Its usage is:

```
ver <file>
```

ver searches the specified file for a version string previously compiled or inserted into the file and extracts and displays a version and date stamp. This works for most SSP utilities and diagnostics firmware. Special options are required to display OBP, Entry PDC, SPP PDC and Symbios Fcode firmware revisions, as shown below:

```
ver -e <Entry PDC file>
```

```
ver -o <OBP2500 file>
```

```
ver -p <SPP PDC file>
```

```
ver -s <Symbios Fcode file>
```

Event processing

This section discusses three event processing utilities:

- `event_logger`
- `log_event`

event_logger

The `event_logger` utility is the SSP Event Logger and has a format as follows:

```
event_logger [-d]
```

`event_logger` receives messages from diagnostic utilities through `rpc` calls and writes them to the event log for later review or processing.

The `-d` option keeps `event_logger` from running as a daemon which is useful for debugging.

`event_logger` is a background daemon and is started by `init` through `inittab`. `event_logger` receives messages for the `event_log` via two different mechanisms. SSP utilities programs send events to the `event_logger` through `rpc` calls. OS events on the other hand, use `UDP` datagrams that are sent out over a specific port. These must be detected and logged as well. Upon receiving an OS event, the event travels the same path as a SSP event.

On reception of an event, the event is written to a complex-specific event_log file at /spp/data/<COMPLEX_NAME>/event_log. When the `event_log` reaches the maximum size (approximately one Mbyte), the `event_logger` compresses the `event_log`, then truncates the file and continue logging events.

Other programs can request that events be sent via `rpc` to them. These programs can use the `libevent_client` library to establish a service and notify the `event_logger` what events it would like to see with a fair degree of simplicity. On reception of an event, once the log file is written, the linked list of interested programs is searched to see if the event matches the criterion requested. If there is a match, the event is sent to that program.

`event_logger` should never terminate, but must be killed. If a second copy of `event_logger` is started it attempts to kill the existing copy of the `event_logger`. There should only be one copy of `event_logger` running at any one time.

The following return code indicates a fatal error occurred.

```
-1   unknown option
```

log_event

`log_event` logs its `STDIN` to the event log as a single event.

`log_event` has the following format:

```
log_event [-c] [event number] -n NODE_ID
```

where:

- `[event number]`—Specifies is the event code to use in one of three ways:
    - Command line
    - First line of the input
    - Default
- `[-c]`—Specifies that event is displayed to the console in addition to logging it to the `event_log`
- `-n NODE_ID`—Specifies the node this event is being logged against

The default event code is used if one is not specified using one of the other methods. The command line method is used by specifying a decimal or hex (leading 0x) as the only number on the command line (the `-c` option optionally may be present). The input mode is used if a number is the first thing on the first line read from `STDIN`. The input mode overrides the command line. In the last case, the entire first line is not logged.

When entering text for an event, you may terminate and send the event with a ctrl-D. You may cancel the event with ctrl-C.

log_event always returns 0.

`log_event` is used by scripts such as the interrogators and extractors to put information into the event log as follows:

```
log_event [-c] [number] -n NODE_ID
```

The `-c` option displays event information output on the console as well. If the event severity is high enough, this happens automatically. `event_logger` displays any events that have a severity greater than the warning level.

The following two examples show how `log_event` can be used:

```
cat data_file | log_event 0x86340001 -n 0
```

This example puts an event in the event log with the event code of 0x86340001. The data will be the information contained in the file data_file.

```
echo “This is a test event” | log_event -n 0
```

This example puts the message “This is a test event” in the event log with the default event code from `log_event`.

Miscellaneous tools

The following miscellaneous tools are described in this section:

- `fix_boot_vector`
- `kill_by_name`

fix_boot_vector

This `sppdsh` script restores the four words at the beginning of NVRAM to point to POST. These four words are used by the ENTRY firmware to determine which process was executing last when an HPMC, TOC, or reset occurs.

kill_by_name

The `kill_by_name` script kills processes by name rather than by process identification. The following is the usage of this script:

```
kill_by_name <process name> <signal to send> <process id to not kill>
```

Table 93 describes the options in `kill_by_name`.

Table 93 **`kill_by_name`** options

Option	Description
`process name`	Process name to kill.
`signal to send`	Default is kill command.
`process id to not kill`	Kills all processes that match the specified name except the one matching this ID.

If the third argument is used, the second argument must also be specified. For example:

```
kill_by_name foo 15 1234
```

13 Scan tools

This chapter details most of the scan tools which include:

- `sppdsh`
- `do_reset`
- `jf-node_info`
- `jf-ccmd_info`
- `jf-reserve_info`

sppdsh

`sppdsh` is an enhanced version of the Korn Shell (`ksh`) with all of the functionality of `ksh`, as well as new commands that are suited to a diagnostic environment. `sppdsh` resides on the SSP in /spp/bin/sppdsh.

The diagnostic shell runs on a SSP that is totally independent of the system itself. The shell requires information about the complexes and nodes attached to the SSP. `ccmd` interrogates the complexes and nodes on the DART bus and generates a database of information on the SSP; it does not act unilaterally.

POST passes system information to `ccmd` through NVRAM about the system itself. If POST fails to initialize the system, `ccmd` will time out and print a warning. If this occurs, many diagnostic shell operations will not work as expected.

On start-up, the diagnostic shell reads the database that `ccmd` provides. If major changes are made to the system, `sppdsh` should be restarted to be sure that the shell has an accurate representation of the system. If `ccmd` is restarted, then the shell *must* be restarted.

`sppdsh` commands are sorted into the following five categories:

- Miscellaneous commands—Control the system behavior and aid in generating useful scripts
- Data transfer commands—Allow the user to transfer register state or memory information back and forth between the system and the SSP
- Data conversion commands—Reformat data to make it more useful
- System information commands—Provide information about the system hardware to run diagnostic upon
- I/O buffering commands—Aid in the testing of I/O devices and memory.
- Configuration commands—Alters a system configuration for POST after a reboot.
- SPP enhancements—New sppdsh commands for the V2500/V2600 server.

The commands in each category are described in the following sections.

Definitions

The following definitions will help user with the operation of `sppdsh`:

- node id—An identification (ID) that can be either the node IP name or a node number. To distinguish between one node number and another, the environmental variable, COMPLEX_NAME, indicates the complex. No complex can have non-unique node numbers.
- complex name—Identifies a grouping of nodes. The `ts_config` utility groups nodes into complexes where each node shares the same OS and memory space.
- all—reference to all nodes associated with the only complex on the diagnostic bus in a single complex configuration.
- <n<node number> | node id>:<ring>:<path>:<part>:<field>—The general description of a register or group of registers that are accessible through scan. Each register is identified by its node, the scan ring that can access it, the part or device that contains it, the scan path within the part, and the text-based description of the register.
- address—A 40-bit value that allows access to the memory, CSR space, IO space and Core Logic bus space across all possible nodes in a complex.
- iteration—An iteration argument indicates the number of consecutive memory addresses to access.
- byte_size—The byte_size argument represents the number of bytes to access at an address. Valid byte_sizes are 1, 2, 4 or 8 but may also be limited by the type of memory accessed. If the byte_size argument is not used, the maximum valid size for the argument is used by default.
- value—A representation of the data transferred to a memory address or scan field.
- parameter—A name that represents configuration data that is initialized by POST. Parameters may be changed to aid testing or to deconfigure hardware that is marginal. Table 94 provides a list of valid parameters.

Table 94 **sppdsh parameters**

Parameter	Value
Unknown	0xff
Reserved	0x00
Pass	0x01
Fail	0x10
Deconfigured by POST	0x20
Empty	0x30
Deconfigured by software	0x40 [a]
16MB deconfigured	0x04
16MB 88-bit deconfigured to 80	0x24
16MB 88-bit deconfigured	0x34
16MB SW deconfigured	0x44
16MB 88-bit SW deconfigured to 80	0x64
16MB 88-bit SW deconfigured	0x74
64MB deconfigured	0x08
64MB 88-bit deconfigured to 80	0x28
64MB 88-bit deconfigured	0x38
64MB SW deconfigured	0x48
64MB 88-bit SW deconfigured to 80	0x68
64MB 88-bit SW deconfigured	0x78
128MB deconfigured	0x0c
128MB 88-bit v to 80	0x2c
128MB 88-bit deconfigured	0x3c
128MB SW deconfigured	0x4c

Parameter	Value
128MB 88-bit SW deconfigured to 80	0x6c
128MB 88-bit SW deconfigured	0x7c
64MB deconfigured to 16MB	0x89
64MB deconfigured to 16MB (88-bit to 80)	0xa9
64MB deconfigured to 16MB (88-bit)	0xb9
SW deconfigured 64MB to 16MB	0xc9
SW deconfigured 64MB to 16MB (88-bit to 80)	0xe9
SW deconfigured 64MB to 16MB (88-bit)	0xf9
128MB deconfigured to 16MB	0x8d
128MB deconfigured to 16MB (88-bit to 80)	0xad
128MB deconfigured to 16MB (88-bit)	0xbd
SW deconfigured 128MB to 16MB	0xcd
SW deconfigured 128MB to 16MB (88-bit to 80)	0xed
SW deconfigured 128MB to 16MB (88-bit)	0xfd
128MB deconfigured to 64MB	0x8e
128MB deconfigured to 64MB (88-bit to 80)	0xae
128MB deconfigured to 64MB (88-bit)	0xbe
SW deconfigured 128MB to 64MB	0xce
SW deconfigured 128MB to 64MB (88-bit to 80)	0xee
SW deconfigured 128MB to 64MB (88-bit)	0xfe

a. System memory can be modified through partial deconfiguration.

- buf[1..4]—A buffer is a 4K byte block of memory on the SSP that is used as a temporary holding area.

- backplane_serial_number—Identifies a specific board on the diagnostic network. This number may be read with the COP command. It is used to assign new node numbers or complex serial numbers.
- complex_serial_number—Identifies all the nodes in a complex. Software licensing is often based on the complex serial number.
- key—A 32-bit hexadecimal number used as an encryption code for complex serial numbers.
- cop_id—A name associated with a board in a node. Table 95 lists valid cop IDs.

Table 95 Valid COP IDs

ID	Description
scub	A system communications and utility board
mib	A midplane or backplane
pb0l	A processor board on the left side of the cabinet
pb0r	A processor board on the right side of the cabinet
pb1l	A processor board on the left side of the cabinet
pb1r	A processor board on the right side of the cabinet
pb2l	A processor board on the left side of the cabinet
b2r	A processor board on the right side of the cabinet
pb3l	A processor board on the left side of the cabinet
pb3r	A processor board on the right side of the cabinet
pb4l	A processor board on the left side of the cabinet
pb4r	A processor board on the right side of the cabinet
pb5l	A processor board on the left side of the cabinet
pb5r	A processor board on the right side of the cabinet
pb6l	A processor board on the left side of the cabinet
pb6r	A processor board on the right side of the cabinet

ID	Description
pb7l	A processor board on the left side of the cabinet
pb7r	A processor board on the right side of the cabinet
mb0l	A memory board on the left side of the cabinet
mb1l	A memory board on the left side of the cabinet
mb2r	A memory board on the right side of the cabinet
mb3r	A memory board on the right side of the cabinet
mb4l	A memory board on the left side of the cabinet
mb5l	A memory board on the left side of the cabinet
mb6r	A memory board on the right side of the cabinet
mb7r	A memory board on the right side of the cabinet
iolf	An IO board on the left-front
iolr	An IO board on the left-rear
iorf	An IO board on the right-front
iorr	An IO board on the right-rear

- Device_name—Refers to a major electrical component or subsection of a node. Examples of device names are:
  - SPAC—Processor agent chip
  - SMAC—Memory chip
  - STAC—SCI transfer chip
  - SAGA—IO controller chip
  - ERAC—Crossbar network chip
  - CPU—Processor
- ID_number—Refers to a specific instance of the device named.
- <A|C|M|D|I|S>—Notation that refers to the processor agent chip, processor, memory, DIMM, IO chip or SCI chip.

- memory size—An argument used to deconfigure larger amounts of memory across all memory boards on a node.
- net cache size—Refers to the memory shared between nodes in each node's network cache. The network cache should be the same across all nodes in a complex.

Miscellaneous commands

`sppdsh` miscellaneous commands are described below:

- `assert <node_id>`—Assert reset on node_id; a deassert must follow.
- `assert_soft <node id>`—Asserts a soft reset on node id.
- `assert_toc <node_id> alt_name <E_name> <ID_number>`—Asserts transfer of control on `node_id`.
- `deassert <node_id>`—Negate reset to `<node_id>`.
- `clock <stop|clock1> <node>:<ring>:<part>`—Issue special clock operations to node:ring:part.
- `fprint "hello world %s is %d" $variable 0xff`—Format output.
- `alt_name <E_name> <ID_number>`—Return the alternate name of a system board or component. For example, entering **`alt_name SAGA 0`**, produces `IOLF_B`. This command does not support the SPUC or SMUC.
- `deassert <node id>`—Deassert reset to node id.
- `reboot <node id| complex name| all> <default|tc_stand|tc_int|obp|epsdv|post_int|loader>`—Reboots the node or complex specified. It can use default or new values for POST configuration data. Default values are determined by POST after ignoring existing values. After POST runs, control can be transferred to OBP, EPSDV, POST interactive mode, Test Controller in stand alone mode, or the Test Controller in interactive mode. When default is specified control is transferred to OBP.
- `clock <node id> [ext|nom|high]`—Changes the clock margin on all nodes in contact with the SSP.

- `power <node id> supply[1..4] [low|nom|up`—Changes the power margin on the supply indicated across all nodes in contact with the SSP.
- `pswitch <node id>`—Identifies whether N or N+1 fans have been enabled for the system. This switch is located on the SCUB board of a node.
- `pce <node id|complex name|all> [-c <n|u|e>] [-r <on|off>] [-p <l|n|u>]`—Displays the current power, clock and temperature state where:
  - `-c [n|u|e]`—Sets the following clock tolerances on the current node:
    - n[ominal]—Nominal frequency
    - u[pper]—Upper frequency
    - e[xternal]—External connector
  - `-r`—Sets the power flag to on or off
  - `-p [l|n|u]`—Displays the following power supply voltages tolerances on the current nodes.
    - `l`[ower]—Lower voltage tolerance
    - n[ominal]—Nominal voltage
    - u[pper]—Upper voltage tolerance

NOTE Clocks are stopped by putting all scannable parts in internal scan mode. Other scan paths are not allowed when clocks are stopped. A system reset must follow an internal scan node operation.

- `ds1620 <ode id|complex name|all> [-c][-b][\-s \<w T>|<W T>|<h T> |<H T>|<s T>|<S T>|<c X>]`—Displays the cold, warm and hot trip points for temperature settings.
  - -c reads and displays the configuration register.
  - -b begins temperature monitor mode.
  - -s set a warm, high or shutdown temperature for N or N+1 fans or a configuration register.

Data transfer commands

This section lists the `sppdsh` data transfer commands. The addresses in the data transfer commands are 40 bits. Underbars are ignored in addresses.

NOTE For clarity, a 0x0 style notation is returned by the shell rather than the 16#0 notation of ksh. The 16#0 notation is acceptable for data that can be expressed in 32 bits or less.

- `list <n<node number> | node id>:<ring>:<path>:<part>:<field>`—Lists the possible paths, parts or fields that match the argument. Common wild card symbols are supported by this command to help identify fields names.
- `put [<node_number>:]<address>[,<byte_size>] <value>`—Starting at the node indicated by `<node id>`, write `<byte_size>` bytes into the memory address using the `<value>`. `sppdsh` is aware of the various memory sizes assumed at various addresses and retrieves the appropriate size (For example, 0x20:0x21 = 0x30)
- `put [-q] n<node_number>:<ring>:<path>:<part>:<field>`—Writes the scan location node `<node_id>:<ring>:<path>:<part>:<field>`. The `-q` option is used to display the result without the scan field name. If a node number is used as the node id then an “n” should precede the node number as “n0”.
- `get [<node_number>:]<address>[,<byte_size>] [<iterations>]`—Reads `<byte_size>` bytes from the memory location `<address>` on node `<node id>`. This command can be repeated with the address incremented. One or `<iterations>` different addresses will be read.
- `get [-q] n<node_number>:<ring>:<path>:<part>:<field>`—Reads scan location node `<node_id>:<ring>:<path>:<part>:<field>`. The `-q` option is used then the result should be displayed without the scan field name. The `-a` option displays both the address and the data. The `-b` option eliminates leading zeros. If a node number is used as the node id then an “n” should precede the node number as “n0”
- `block n<node_number>:<ring>:<path>`—Reads the scan ring at `<node id>:<ring>:<path>` and lock the scan ring image for `bget` and `bput` operations.

- `bget [-q] <part>:<field>`—Extracts data from the locked scan ring image. When the `-q` option is used, the results are displayed without the scan field name
- `bput [-q] <part>:<field> <value>`—Inserts data into the locked scan ring image. When the `-q` option is used, the results are displayed without the scan field name.
- `bunlock n<node_number>:<ring>:<path>`—Writes the scan ring image and unlocks it.
- `packet [-q] [NR | R=number] [P=number] [6=number] node8_0 <packet symbols>`—Requests input to a crossbar device from SPAC 0 on node 8. The request waits for a response and returns it. The `-q` option suppresses some output. Other arguments are as follows:
  - NR—No response
  - R = N—Response of N symbols
  - P = N—Select port N
  - 6 = N—Use N as the R6 symbol

  The following is an example of this command:

  ```
  packet R=3 P=1 node0_2 rd_short R=3 lgth=1 route=4
  adr=0 dl=0 lcl=1 mstr=2a tid=1c size=3 Q=0
  ```
- `file_to_mem <file_name> <address>`—Reads a file and loads the file into memory starting at address.
- `mem_cmp <address1> <address | buffer> <size>`—Compares the memory at `address1` to `address1+size` to that at `address2`. `size` is a value in bytes.
- `mem_dump <address> [size]`—Dumps the memory starting at `address`. `size` is a value in 64-bit words.
- `mem_cpy <address | buffer> <address | buffer> [size]`—Copies the memory from `address1` to `buffer1` - up to `size` or 4K bytes. `size` is a value in 64-bit words.
- `tag_dump <address> [size]`—Dumps the tags associated with the cache line of `address` and repeat for `size` cache lines.
- `tag_cpy <address> <data> [size]`—Copies the data into the tags associated with the cache line of `address` and repeat for `size` cache lines.

- `ecc_dump <address> [size]`—Dumps the ECC bits associated with the cache line of `address` and repeat for `size` cache lines.
- `ecc_cpy <address> <data> [size]`—Copies the data into the ECC associated with the cache line of `address` and repeats for `size` cache lines.

Data conversion commands

Data conversion commands manipulate, evaluate or interpret data within the diagnostic shell. They support a variety of logical, arithmetic and string based operations. The following example is representative of the data conversion commands:

```
abc='and 0xff 0x55'
```

Unless otherwise stated, these commands support data types that are greater than 32 bits, not supported under `ksh`.

The following is a list of `sppdsh` data conversion commands:

`and <arg1> <arg2>`—And two data arguments.

or <arg1> <arg2>—OR two data arguments. For example:

```
abc='or 0xff 0x55'
```

`xor <arg1> <arg2>`—Exclusively OR two data arguments. For example:

```
abc='xor 0xff 0x55'
```

`even_parity <arg1> <arg2>`—Return even parity. For example:

```
abc='even_parity 0xff''
```

`odd_parity <arg1> <arg2>`—Return odd parity. Parity is based on 4 bytes, as an example:

```
abc='odd_parity 0xff'
```

`comp <arg1> <arg2>`—Compare two data arguments. For example:

```
abc='comp 0xff 0xff'
```

`rshift <arg1> <arg2>`—Right shift two data arguments. For example:

```
abc='rshift 0x55 0x1'
```

`lshift <arg1> <arg2>`—Left shift two data arguments. For example:

```
abc='lshift 0x55 0x1'
```

`l_add <arg1> <arg2>`—Left add two data arguments. For example:

```
abc='l_add 0x55 0x1'
```

`l_sub <arg1> <arg2>`—Left subtract two data arguments. For example:

```
abc='l_sub 0x55 0x1'
```

`l_mod <arg1> <arg2>`—Left modulo two data arguments. For example:

```
abc='l_mod 0x55 0x1'
```

`l_mult <arg1> <arg2>`—Left multiply two data arguments. For example:

```
abc='l_mult 0x55 0x1'
```

`b2h <arg1> <arg2>`—Converts a binary number to hex (abc = 0xb). This command is limited to 32-bit data types. For example:

```
abc='b2h 1011'
```

`h2b <arg1> <arg2>`—Converts a hex number to binary (abc = 1011). This command is limited to 32-bit data types. For example:

```
abc='h2b 0xb'
```

`conv <arg1> <arg2>`—Converts from hex to decimal. This command is limited to 32-bit data types. For example:

```
abc='conv 0xb'
```

`conv <arg1> <arg2>`—Converts from decimal to hex. For example:

```
abc='conv 11'
```

converts 11 from decimal to hex and assign it to abc (abc = 0xb). This command is limited to 32-bit data types.

`s_tos <arg1> <arg2>`—Removes underbar from a hex number. For example:

```
abc='s_tos 0xff_abcd'
```

Converts 0xff_abcd to 0xffabcd and assigns it to abc.

`fprint "hello world %s is %d" $variable 0xff`—This command formats data for reasonable output.

`alt_name <device_name> <ID_number>`—Specifies the <device_name> and <id_number> of a device that may be replicated more than once in a system. For example, the device SAGA is used on four IO boards in two different locations. Instance 0 of the SAGA refers to the IOLF board in the B location.

`alt_name SAGA 0` produces IOLF_B. The device names SPUC and SMUC are not supported.

System information commands

The following are system information commands. For all these command, `-v` produces the verbose manufacturing name, `-q` produces the name alone without additional information, and `-a` produces the available memory.

`complex <c_name>`—Set the current, default complex to be complex c_name. If only one complex is available, this command is not necessary.

`node <node _number>`— set default node to be node _number in the current complex.

`fi_node`—Find all available nodes in the current complex.

`fi_complex`—Finds all available complexes.

`fi_cpu [-v] [-q] <node_number>`—Find all available processors of `node_number` in the current complex.

`fi_emb [-v] [-q] <node_number>`—Find all available EMBs of `node_number` in the complex.

`fi_sci [-v] [-q] <node_number>`—Find all available SCIs of `node_number` in the current complex.

`fi_saga [-v] [-q] <node_number>`—Find all available SAGAs of `node_number` in the current complex.

`fi_pac [-v] [-q] <node_number>`—Find all available SPACs of `node_number` in the current complex.

`fi_rac [-v] [-q] <node_number>`—Find all available ERACs of `node_number` in the current complex.

`fi_tac [-v] [-q] <node id>`—Find all available STACs of node id in the current complex. This is the same command as fi_sci.

`fi_slice [-v] [-q] <node_number>`—Find all available slices of `node_number` in the current complex.

`fi_mem_inst [-a] [-q] <node_number>`—Find the installed memory size per EMB of `node_number` in the current complex.

`swid <node>|<complex name>|"all"`—Display the Software Identifier (SW_ID) for the specified node or complex.

Configuration commands

The following is a list of the `sppdsh` configuration commands:

`retrieve <node_number>`—Retrieve the `node_number` configuration parameters to the SSP from NVRAM.

`replace <node_number>`—replaces the `node_number` configuration parameters from the SSP to NVRAM.

`clist <parameter>`—Parameters are names representing POST configurable data where ever possible these parameters should have the same names as used in OBP.

`cput <node_number> parameter_name 0xnnnnnnnn`–Set the configuration `parameter_name` of `node_number` to 0xnnnnnnn in the SSP buffer.

`cget <node_number> parameter_name`—Get the value stored in the configuration `parameter_name` of `node_number` in the SSP buffer.

I/O buffering commands

This section presents a list of the `sppdsh` I/O buffering commands. For these commands, four default buffers are created: buf1 - buf4.

`buf_cmp buf1 buf2`—Compares two buffers. Null is returned if they are the same. If they are different, the index and data of the first conflict is returned.

`buf_cpy buf1 buf2`—Copy buf1 to buf2

`buf_clear buf`—Clear buf1

`seed [get|set 0xseed_value]`—Set or get a seed value.

`buf_mod [-bw|-s len value] buf_name [value | key_data] [nbr] [offset]`—Write to buffer. The following are three examples:

1. **`buf_mod buf1 0x01234567 10 2`**

   writes 0x01234567 10 times with an offset of 2 words.

2. **`buf_mod -b buf1 0x3d 1 10`**

   writes the byte 0x3d once at 0x10

3. **`buf_mod -s 5 0 buf2`**

   write five zeros then five ones for all of the buffer space using the following key data:

   - rand—Produces random data based on the seed
   - zeroes—Produces all zero data
   - ones—Produces all one data
   - alt1—Alternates zeros and ones
   - alt2—Alternates ones and zeros

`buf_print buf1`—Print buffer contents.

`buf_read buf1 [size]`—Prints the value of a byte in the given buffer.

Enhancements

The following is a list of enhancements for V2500/V2600 systems.

`assign <node id> <mib_sn> <complex_sn> [<swid upper> <swid lower>] <key>]`—Links a node number (`node_id`) or complex serial number (`complex_sn`) to a specific MIB backplane.

The `node_id` and `complex_sn` number are required by the system OS and many diagnostic utilities. When a complex serial number is assigned, the `swid` of the node is also updated. If the optional `swid` arguments are not specified, the `swid` will be derived based on the complex serial number. `key` is required to assign a complex serial number to a node.

`<complex_sn>` must be exactly 10 alphanumeric characters. If the `swid` arguments are specified, the complex serial number can be any combination of alphanumeric characters, as long as it is exactly 10 characters. If the `swid` argument is not specified, the format of the complex serial number is restricted even further. This is necessary to ensure a valid `swid` is generated. The format of the complex serial number is:

`LLLYYWWNNN`

where the location code LLL is restricted to one of the following: DEH, GBM, USJ, USM, USN, USR, and USS.

The sequence number NNN is an alphanumeric value between 000 and P6L where the following alpha characters are excluded from use: G, I, O, Q, U, and Z.

NOTE When upgrading a V22xx system to a V2500 or V2600, the optional swid arguments may be used to retain SWIDs used on the V22xx system. Use `uname -i` on the V22xx to obtain the existing V22xx swid. When assigning the new V2500/V2600 complex serial number, use 0 in place of the `<swid upper>` argument and the previous swid as the `<swid lower>` argument.

key is an 8 digit hexadecimal number. The assign command requires a leading 0x before the key.

If the complex key is 11223344, an example assign command might be:

```
assign 0 2014998 USR2222P6L 0x11223344
```

`blink <node id>`—Physically identifies a node. This command forces the node leds to blink or turns off blinking, provided an error does not exist on the node.

`48v <node id>`—Resets the 48-volt glitch detector for the node power supply.

`cop <node id|complex_id|all> <cop_id> <-ASICs>`—Checks for all cop devices through scan. If a scan ring is broken then the cop utility may try to read cop memories for boards that may not exist.

`copmod <node id> <cop device> [-<n|b|p|e|c|a|s> value]`—Modifies cop data. The cop command is used for reads only. The options are defined as follows:

- n—Node number
- b—Board serial number
- p—Part number
- e—Engineering Date Code
- c—Clear the cop
- a—Artwork revision
- s—Scan revision

`mnetcache <node id> <?|net_cache_size>`—Specifies the CTI cache size for a node. The network or CTI cache is a valid parameter for a single node system.

`msize <node id> <?|memory_size>`—Reports the memory size on the node. `memory size #` resets NVRAM to support smaller memory sizes

`stop_on_hard <node id|complex_id|all> <on|off|chk>`—Prevents the OS from trying to clean-up any system errors.

`query <node id> <A|C|M||I|S|D#>`—Queries a node for configuration. The options are defined as follows:

- `A`—agents or SPACs
- `C`—CPUs or processors
- `M`—memory boards or SMACs
- `I`—I/O devices or SAGAs
- `D`—DIMMs on a memory board
- `S`—SCSI devices

`toggle [-u] <node id> <A#|C#|M#|I#|S#|D<dimm_identifier>>`—Changes the state of a part of the system between active and disabled. The `-u` option allows updates to be applied via utility bus scan rather than using the SPAC `unwedger`. This is necessary when attempting to disable/enable devices when the clocks are stopped, as is the case after a hard error.

The options are the same as used in the `query` command. Each is followed by a single number. DIMMs may be toggled individually with either the physical representation as `mb01.q0b0r0` or the software representation as `ewmb_0.dimm_0.row_0`. The `clist` command may be helpful in identifying the name of a DIMM to toggle.

`nodemap <node id>`—Allows the user to reconfigure some OBP and OS support data.

Map of alternate names

Table 96 lists the alternate names of ring numbers to system parts.

Table 96 **System rings to alternates names**

Ring	Parts	Alternate names
0	pb0l, p0l, pb0r	[pcxw], spac0, [pcxw]
1	pb1r, p1l, pb1l	[pcxw], spac1, [pcxw]
2	pb2l, p2l, pb2r	[pcxw], spac2, [pcxw]
3	pb3r, p3l, pb3l	[pcxw], spac3, [pcxw]
4	pb4l, p4l, pb4r	[pcxw], spac4, [pcxw]
5	pb5r, p5l, pb5l	[pcxw], spac5, [pcxw]
6	pb6l, p6l, pb6r	[pcxw], spac6, [pcxw]
7	pb7r, p7l, pb7l	[pcxw], spac7, [pcxw]
8	mb0l_m, mb0l_t	smac0, [stac0]
9	mb1l_m, mb1l_t	smac1, [stac1]
10	mb2r_m, mb2r_t	smac2, [stac2]
11	mb3r_m, mb3r_t	smac3, [stac3]
12	mb4l_m, mb4l_t	smac4, [stac4]
13	mb5l_m, mb5l_t	smac5, [stac5]
14	mb6r_m, mb6r_t	smac6, [stac6]
15	mb7r_m, mb7r_t	smac7, [stac7]
16	iolf_b, iolf_a	saga0, saga4
17	iolr_b, iolr_a	saga1, saga5
18	iorr_b, iorr_a	saga2, saga6
19	iorf_b, iorf_a	saga3, saga7
20	rol, r2r	erac0, erac1
21	r1l, r3r	erac2, erac3
22	u_p, u_m	spuc, smuc

do_reset

`do_reset` performs one of four levels of reset on a node or complex and has the following format:

```
do_reset [node id | complex name | all] [level] [boot option]
```

The first argument is either a node ID, complex, or the keyword, `all`, which resets all nodes. If no nodes are specified, the default is to reset all nodes in contact with the SSP. If a node number is specified, the `level` argument must be specified as well.

`node id` may be a node number or a node IP name within the current complex.

`complex name` is the name of the desired complex. All nodes within the specified complex are reset.

`all` specifies all nodes within the current complex are reset.

The second argument specified is the level of reset. All levels of reset are expressed as numbers. If no level is specified, then a reset level of 1 is assumed.

The following reset levels are available:

1. JTAG controller SCUB reset, hard reset, clear options bits, and send messages to ccmd
2. JTAG controller SCUB reset and system soft reset
3. JTAG controller SCUB reset
4. TOC reset

`do_reset` halts any scan activity taking place on the nodes. Larger systems require more time to reset. There may be a minor delay between the time that the reset occurs and when `ccmd` reports it.

If the reset completes normally, `do_reset` returns zero. If the operation cannot be completed, `do_reset` returns a nonzero exit code.

`do_reset` has the following options:

- `OBP`—Reboots to OBP.
- `post_interactive`—Reboots to POST in the interactive mode.
- `spsdv`—Reboots to SPSDV mode.

- `tc_standalone`—Reboots to the test controller in stand-alone mode.
- `tc_interactive`—Reboots to the test controller in interactive.
- `loader`—Reboots to the firmware loader.
- `rdr_dumper`—Reboots to the RDR dumper.

If a node or parts of a node have clocks stopped due to the hard logger or other scan operation, the boot option cannot be changed. Instead two resets need to be issued: one to restart the system and the second to change the boot option.

jf-node_info

`jf-node_info` displays the IP address, UDP port and JTAG firmware version string for each node in a complex. The `-e` option adds the ethernet address to the display. The `-c` option adds the core version to the display.

jf-ccmd_info

`jf-ccmd_info` displays information about active V2500/V2600 nodes connected to the diagnostic LAN. It has the following format:

`jf-ccmd_info`

The display includes the Ethernet address, IP address, Complex Serial number, Node number, environmental LED status, and the Diagnostic node name of each active V2500/V2600 node.

`jf-ccmd_info` sends a broadcast packet to all nodes on the diagnostic LAN requesting this information. `jf-ccmd_info` accumulates responses received within a short timeout period then sorts the responses based on node name.

The JTAG utility firmware responds to the request with output similar to the following:

```
joker-t(hw2a):/users/sppuser$ jf-ccmd_info

                              Complex   Node Env  Pwr  Cub  Diagnostic
Ethernet Addr  IP Address     Serial #    #  Led  Sts  Sts  Node name
-------------- -------------- ---------- - ---  ---  ---  ----------
0x00A0D900BF03 15.99.111.116  SN12757550 0  0x00 0x80 0x00 hw2a-0000
0x00A0D900C3A3 15.99.111.117  SN13169380 0  0x00 0x80 0x00 hw2b-0000
```

CAUTION `jf-ccmd_info` displays information about all active V2500/V2600 nodes that answer the broadcast request, even if the node is not configured.

If the `jf-ccmd_info` utility displays information about a node, but the node has not been detected by the `ccmd` daemon, then the node is not configured. Use the `ts_config` utility to configure the node.

jf-reserve_info

Before using the JTAG scan interface on the Utilities board, SSP utilities must *reserve* the JTAG hardware on a time-sharing basis. It has the following format:

```
jf-reserve_info
```

`jf-reserve_info` sends a broadcast packet to all nodes on the diagnostic LAN requesting the latest JTAG reservation information. The JTAG utility firmware responds to the request with output similar to the following:

```
joker-t(hw2a)% jf-reserve_info

RSV Node Node name        UID     PID  TTY     Time of reserve     Command
--- ---- -------------- ------- ----- -----   ------------------- -------
 -   0   hw2a-0000       sppuser  2934 pts/3   Oct 26 16:40:19:229 sppdsh
```

The RSV column indicates whether the JTAG hardware is currently reserved. This column may contain a Y (indicating YES) or "-], indicating no current reservation.

If the JTAG hardware is reserved, the output includes information about the SSP utility that is currently using the JTAG hardware.

If the JTAG hardware is not reserved, the process information shown is historical data about the last process that reserved the JTAG hardware on the specified node.

A List of diagnostics

This appendix provides a list of all utilities and diagnostics in this book and where they are located.

Table 97 List of diagnostics

Name	Locations
`address_decode`	Page 253
`arrm`	Page 254
`autoreset`	Page 59
`ccmd`	Page 49
`console`	Page 256
`consolebar`	Page 260
`cpu3000`	Chapter 7, page 149
`cpu_hang`	Page 261
`cxtest`	Chapter 5, page 125
`dcm`	Page 263
`dfdutil`	Page 267
`diag_version`	Page 315
`do_reset`	Page 340
`dump_rdrs`	Page 277
`eri3000`	Chapter 8, page 157
`est`	Chapter 11, page 221
`est_config`	Page 59
`event_logger`	Page 317
`fix_boot_vector`	Page 320
`flash_info`	Page 315

Name	Locations
fwcp	Page 278
fw_init	Page 279
fw_install	Page 281
get_node_info	Page 284
hard_logger	Page 286
io3000	Chapter 9, page 173
jf-ccmd_info	Page 343
jf-node_info	Page 342
jf-reserve_info	Page 344
kill_by_name	Page 320
lcd	Page 288
load_eprom	Page 289
log_event	Page 318
mem3000	Chapter 10, page 207
opie	Page 292
pciromldr	Page 296
pdcfl	Chapter 6, page 143
pim_dumper	Page 300
POST	Chapter 3, page 67
set_complex	Page 302
soft_decode	Page 304
sppdsh	Page 322
spp_console	Page 305
tc_init	Page 309

Name	Locations
`tc_ioutil`	Page 311
`tc_show_struct`	Page 312
`ts_config`	Page 21
`ver`	Page 316
`xconfig`	Page 51
`xsecure`	Page 64

B LED codes

This appendix describes core utilities board (SCUB) LED errors. The Attention LED on the core utilities board (SCUB) turns on, and the Attention light bar on the front of the node flashes to indicate the presence of an error code listed Table 98. Additionally, only the highest priority error is displayed. Once remedied, an error that is cleared may expose a lesser priority error.

Power on detected errors

This section describes core utilities board (SCUB) LED errors from highest to lowest priority detected at power on. The Attention LED on the core utilities board (SCUB) turns on, and the Attention light bar on the front of the node flashes to indicate the presence of an error code listed in Table 98. Additionally, only the highest priority error is displayed. Once remedied, an error that is cleared may expose a lesser priority error.

Errors are listed in sequence from the highest to lowest priority.

NOTE Errors from LED hex code 00 through hex code 67 shut the system down, and errors from hex-code 68 through 73 leave the system up.

Table 98 SCUB detects power on error

LED	Fault	Symptoms	Corrective action
00	3.3V error (highest priority)	1. 5V is up 3.3V is not. 2. SSP interface will not function.	Call the Response Center.
01	ASIC Install 0 (MIB)	1. Incorrect rotation or part in one of the processor agent chip (PAC) sockets. 2. Incorrect rotation or part in one of the routing (XBAR) attachment chip (RAC) sockets.	Call the Response Center.
02	ASIC Install 1 (EMB)	1. Incorrect rotation or part in one of the memory access chip (MAC) sockets. 2. Incorrect rotation or part in one of the toroidal access chip (TAC) on memory board (MB).	Call the Response Center.

LED	Fault	Symptoms	Corrective action
03	FPGA not OK	1. Core Utilities Board (SCUB) monitoring utilities chip (MUC) problem. 2. MUC cannot get correct program transfer from EEPROM on power up.	• Cycle the node power using the Key switch. • Call the Response Center.
04	dc OK error (Upper Left)	1. Power supply is reporting failure (dc OK) after keyswitch is turned on, but prior to SCUB power on sequence. 2. This is the first of two or more supplies reporting failure.	Call the Response Center.
05	dc OK error (Upper Right)	1. Power supply is reporting failure (dc OK) after keyswitch is turned on, but prior to SCUB power on sequence. 2. This is the first of two or more supplies reporting failure.	Call the Response Center.
06	dc OK error (Lower Left)	1. Power supply is reporting failure (dc OK) after keyswitch is turned on, but prior to SCUB power on sequence. 2. This is the first of two or more supplies reporting failure.	Call the Response Center.
07	dc OK error (Lower Right)	1. Power supply is reporting failure (dc OK) after keyswitch is turned on, but prior to SCUB power on sequence. 2. This is the first of two or more supplies reporting failure.	Call the Response Center.

LED	Fault	Symptoms	Corrective action
08-11	48V error NPSUL failure PWRUP=0-9	1. Error occurs when 48 volt distribution falls below 42 volts during power up state displayed. Power up state indicates which loads are being turned on. 2. Excessive load on 48 volts due to an inadequate number of functioning 48 volt supplies or overload condition on 48V bus. 3. Possible node power supply (NPS) upper left failure.	Call the Response Center.
12-1B	48V error NPSUR failure PWRUP=0-9	1. Error occurs when 48 volt distribution falls below 42 volts during power up state displayed. Power up state indicates which loads are being turned on. 2. Excessive load on 48 volts due to an inadequate number of functioning 48 volt supplies or overload condition on 48V bus. 3. Possible node power supply (NPS) upper right failure.	Call the Response Center.

LED	Fault	Symptoms	Corrective action
1C-25	48V error NPSLL failure PWRUP=0-9	1. Error occurs when 48 volt distribution falls below 42 volts during power up state displayed. Power up state indicates which loads are being turned on. 2. Excessive load on 48 volts due to an inadequate number of functioning 48 volt supplies or overload condition on 48V bus. 3. Possible node power supply (NPS) lower left failure.	Call the Response Center.
26-2F	48V error NPSLR failure PWRUP=0-9	1. Error occurs when 48 volt distribution falls below 42 volts during power up state displayed. Power up state indicates which loads are being turned on. 2. Excessive load on 48 volts due to an inadequate number of functioning 48 volt supplies or overload condition on 48V bus. 3. Possible node power supply (NPS) lower right failure.	Call the Response Center.

LED	Fault	Symptoms	Corrective action
30-39	48V error (maintenance) no supply failure reported PWRUP=0-9	1. Error occurs when 48 volt distribution falls below 42 volts during power up state displayed. Power up state indicates which loads are being turned on. 2. Excessive load on 48 volts due to an inadequate number of functioning 48 volt supplies or overload condition on 48V bus. 3. Possible node power supply (NPS) failure.	Call the Response Center.
3A	48V Yo Yo error	1. Core utilities board (SCUB) lost and then regained 48V power without the machine being turned off or ac power failure. 2. Core utilities board (SCUB) will display this error and not power on the system.	• Cycle dc power to the node using the keyswitch to attempt to clear the Yo Yo bit. • Call the Response Center.
3B	MIB power fail (MIBPB)	1. VDD (3.3V) error on MidPlane power board (MIBPB). 2. Midplane power fails and entire node will power down. 3. Core utilities board (SCUB) still active.	• Cycle dc power to the node using the keyswitch to attempt to clear the error. • Call the Response Center.
3C	Clock fail	• Core utilities board (SCUB) monitors clock on MidPlane (MIB).	Call the Response Center.

SCUB detected memory power fail

This describes covers memory errors detected by the monitoring utilities chip (MUC) on the core utilities board after power-on.

Table 99 SCUB detects memory power fail

LED	Fault	Symptoms	Corrective action
40	MB0L Power Fail	1. 3.3V dropped below acceptable level. 2. Core utilities board (SCUB) detected a power loss on reported memory board (MB). 3. Core utilities board (SCUB) powers down the system	Call the Response Center.
41	MB1L Power Fail		
42	MB2R Power Fail		
43	MB3R Power Fail		
44	MB4L Power Fail		
45	MB5L Power Fail		
46	MB6R Power Fail		
47	MB7R Power Fail		

SCUB detected processor error

This section describes processor errors detected by the monitoring utilities chip (MUC) on the core utilities board after power-on.

Table 100 SCUB detects processor power fail

LED	Fault	Symptoms	Corrective action
48	PB0L Power Fail	1. 3.3V dropped below acceptable level. 2. Core utilities board (SCUB) detected a power loss on the reported processor board (PB). 3. Core utilities board (SCUB) powers down the system.	Call the Response Center.
49	PB1R Power Fail		
4A	PB2L Power Fail		
4B	PB3R Power Fail		
4C	PB4L Power Fail		
4D	PB5R Power Fail		
4E	PB6L Power Fail		
4F	PB7R Power Fail		
50	PB0R Power Fail		
51	PB1L Power Fail		
52	PB2R Power Fail		
53	PB3L Power Fail		
54	PB4R Power Fail		
55	PB5L Power Fail		
56	PB6R Power Fail		
57	PB7L Power Fail		

SCUB detected I/O error

This section describes I/O errors detected by the monitoring utilities chip (MUC) on the core utilities board after power-on.

Table 101 SCUB detects I/O (IOB) power fail

LED	Fault	Symptoms	Corrective action
58	Left Front I/O Board failure	1. 3.3V or 5V dropped below acceptable level (+12V and -12V not monitored). 2. Core utilities board (SCUB) detected a power loss on reported I/O board (IOB). 3. Core utilities board (SCUB) powers down the system.	Call the Response Center.
59	Left Rear I/O Board failure		
5A	Right Front I/O Board failure		
5B	Right Rear I/O Board failure		

SCUB detected fan error

This section describes fan errors detected by the monitoring utilities chip (MUC) on the core utilities board after power-on.

NOTE Fan positions are referred to as viewed from the rear of the server.

Table 102 SCUB detects fan power fail

LED	Fault	Symptoms	Corrective action
5C	Fan failure Upper Right	Sensor in the reported fan (as viewed from rear of system) determines fan failure.	Call the Response Center.
5D	Fan failure Upper Middle		
5E	Fan failure Upper Left		
5F	Fan failure Lower Right		
60	Fan failure Lower Middle		
61	Fan failure Lower Left		

SCUB detected ambient air errors

This section describes air errors detected by the monitoring utilities chip (MUC) on the core utilities board after power-on.

Table 103 SCUB detects ambient air error

LED	Fault	Symptoms	Corrective action
62	Ambient hot	1. Ambient air too hot. 2. Core utilities board (SCUB) powers down system. 3. Should have received “ambient air too warm” error 69 prior to this error.	• Check site temperature. • Call the Response Center.
63	OVERTEMP MIB	1. MidPlane (MIB) too hot. 2. Core utilities board (SCUB) sensed overtemp on MidPlane power board (MIBPB) and powers down the system.	• Check that air flow is not blocked. • Check fans. • Call the Response Center.
64	QUADRL 0	1. Board overheated in Quadrant 0. 2. Core utilities board (SCUB) sensed overtemp in Quadrant 0 and powers down the system.	Call the Response Center.
65	QUADRU 1	1. Board overheated in Quadrant 1. 2. Core utilities board (SCUB) sensed overtemp in Quadrant 1 and powers down the system.	Call the Response Center.
66	QUADLL 2	1. Board overheated in Quadrant 2. 2. Core utilities board (SCUB) sensed overtemp in Quadrant 2 and powers down the system.	Call the Response Center.
67	QUADLU 3	1. Board overheated in Quadrant 3. 2. Core utilities board (SCUB) sensed overtemp in Quadrant 3 and powers down the system.	Call the Response Center.

SCUB detected hard error

This section describes hard errors detected by the monitoring utilities chip (MUC) on the core utilities board after power-on.

Table 104 **Hard error**

LED	Fault	Symptoms	Corrective action
68	Hard error (RAC) (PAC) (MAC) (TAC) (SAGA)	1. Hard error lines to core utilities board (SCUB) reported ASIC problem. 2. Bit and hard error bus determine which ASIC to check	• Read `/spp/data/hard_list`. • Call the Response Center.

SCUB detected intake ambient air error

This section describes air intake errors detected by the monitoring utilities chip (MUC) on the core utilities board after power-on.

Table 105 Ambient air (intake) error

LED	Fault	Symptoms	Corrective action
69	Ambient air too warm is an environmental warning	Intake air through SCUB too warm.	• Check site temperature and correct. • If the fault reoccurs when room temperature is within specification. call the Response Center

SCUB detected dc error

This section describes dc errors detected by the monitoring utilities chip (MUC) on the core utilities board after power-on.

Table 106 **dc error**

LED	Fault	Symptoms	Corrective action
70	NPSUL failure (warning)	1. Node power supply (Viewed from Node front) failure reported. 2. Low-priority error for redundant power configurations.	Call the Response Center
71	NPSUR failure (warning)		
72	NPSLL failure (warning)		
73	NPSLR failure (warning)		

Displaying the SCUB LED values using pce

Use the `sppdsh` command `pce` to display the value of the LEDs on the SCUB.

Step 1. Bring up the `sppdsh` prompt at a sppuser window by entering:

```
$ sppdsh
```

Step 2. Use the `pce` command to display the LED values for all nodes, enter:

```
sppdsh: pce all
```

```
Node     IP address Clocks      LEDS   @C U SHPT Supply1 Supply2 Supply3 Supply4
------------------- ------ --------- ---- ------ ------- ------- ------- -------
  0   15.99.111.116 Normal   0x00      25 1 0000 Nominal Nominal Nominal Nominal
  2   15.99.111.117 Normal   0x00      25 1 0000 Nominal Nominal Nominal Nominal
```

For more information about the `pce` command see the sppdsh man page.

Step 3. Decode the LED values using Appendix A, “LED codes” .

Identifying a node with the blink command

The blink command is used to physically identify a node. This command forces the node attention light bar to blink or turns off blinking, provided an error does not exist on the node.

Step 1. Bring up the `sppdsh` prompt at a sppuser window by entering:

```
$ sppdsh
```

Step 2. Use the `blink` command to cause the attention light bar to blink on a specific node by entering the `blink` command followed by the node number. For example:

```
sppdsh: blink 0
```

For more information about the `blink` command see the sppdsh man page.

Step 3. After you have physically identified the node cause the attention light bar to return to a steady state by entering:

```
sppdsh: blink 0
```

C Memory configurations

In the V2500/V2600 server, Excalibur Pluggable Memory Boards (EPMBs) are installed in 16 DIMM connectors on the EWMBs.

A V2500/V2600 memory board is organized by quadrants, rows, and buses. Each memory board has four quadrants, four rows and eight buses.

The following terms are used to describe a V2500/V2600 memory board, as shown in Figure 74:

Slot	The physical location into which DIMMs are installed. There are 16 DIMM slots, each with a unique designator which denotes the slot's quadrant and bus.
Quadrant	A group of four DIMM slots staggered across the memory board.
Buses	Eight buses span the four rows. Each DIMM in a quadrant is on a different bus.
Rows	Each DIMM has SDRAMs on each side and represents two rows. For instance, the first DIMM installed in the system would represent row 0 bus 0 and row 1 bus 0. All DIMMs have the same SDRAMs on both sides. Therefore, rows 0 and 1 will have the same SDRAM size. Rows 2 and 3 will have the same SDRAM size. Bus interleaving can be configured to either 4 way or 8 way bus interleaving. 8 way provides the best performance. To achieve 8 way bus interleaving, all buses on a row must be populated with DIMMs having the same SDRAM size.

Table 107 shows the correlation between a DIMM slot and a row bus intersection. The first DIMM to be installed in a memory board, Q0B0, occupies row 0 bus 0 and row 1 bus 0 in quadrant 0.

Table 107 DIMM row/bus table

Rows	Buses							
	0	1	2	3	4	5	6	7
0	Q0B0	Q0B1	Q0B2	Q0B3	Q1B4	Q1B5	Q1B6	Q1B7
1								
2	Q2B0	Q2B1	Q2B2	Q2B3	Q3B4	Q3B5	Q3B6	Q3B7
3								

V2500/V2600 DIMM quadrant designations

Memory boards can be populated in increments of four DIMMs called quadrants.

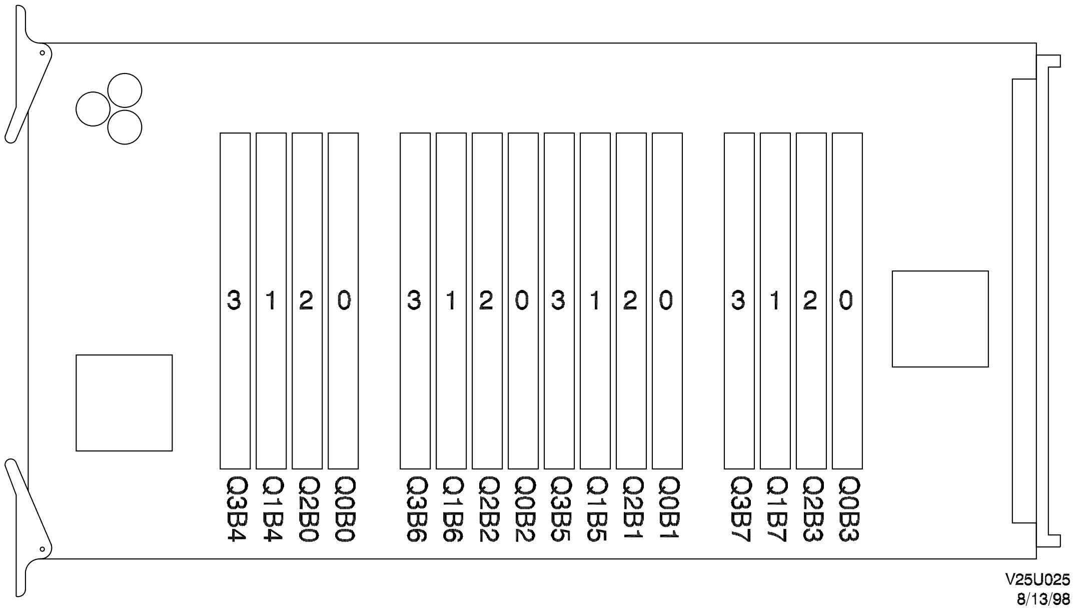
- Four DIMMS provides 1/4 population
- Eight DIMMS provides 1/2 population
- Twelve DIMMS provides 3/4 population
- Sixteen DIMMS provides full population

Table 108 shows the rows and buses associated with each quadrant ID and Figure 74 shows how these are laid out on the memory board.

Table 108 Quadrant assignments

Rows	**Buses**							
	0	1	2	3	4	5	6	7
0	Quadrant 0				Quadrant 1			
1								
2	Quadrant 2				Quadrant 3			
3								

Figure 74 V2500/V2600 DIMM locations



Example:

Q2B3: Quadrant 2, Bank 3

V2500/V2600 DIMM configuration rules

Use the following rules to plan the memory board DIMM configuration:

- All memory boards must be populated identically.
- Single node memory boards may be populated in 1/4, 1/2, 3/4, or full increments.
- Multi node memory boards may be populated in only 1/4, 1/2, or full increments.
- All DIMMs within a quadrant must be of the same size: 32 Mbyte, 128 Mbyte or 256 Mbyte.
- DIMMs in quadrant 0 can be of a different size than DIMMs in quadrant 2 or 3 without degrading performance.
- DIMMs in quadrant 1 can be of a different size than DIMMs in quadrant 2 or 3 without degrading performance.
- DIMMS in quadrant 0 and 1 should be the same size for maximum performance.
- DIMMS in quadrant 2 and 3 should be the same size for maximum performance.
- DIMMs in quadrant 0 can be of a different size than DIMMs in quadrant 1. To allow this memory to be fully utilized, the bus interleave span will be reduced to 4 way bus interleaving. This will degrade performance.
- DIMMs in quadrant 2 can be of a different size than DIMMs in quadrant 3. To allow this memory to be fully utilized, the bus interleave span will be reduced to 4 way bus interleaving. This will degrade performance.
- Mixing of 32-Mbyte DIMMS and 256-Mbyte DIMMs is not supported.
- All quadrants on a given memory board do not have to be populated with DIMMs.

V2500/V2600 memory board configuration rules

The V2500/V2600 system supports up to eight memory boards. Valid configurations of memory boards include two, four, and eight. (A six memory board configuration is not supported.) The first two memory boards, as shown in Table 109 on page 370, are located in slots MB0L and MB1l.

Table 109 Memory board configurations

Order	Slot locations
Minimum system configuration	MB0L MB1L
Four memory boards	MB6R MB7R
Eight memory boards	MB2R MB3R MB4L MB5L

Index